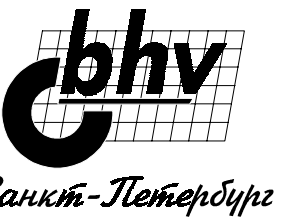
Марина Дмитриева

САМОУЧИТЕЛЬ
JavaScript



Дюссельдорф ◆ Киев ◆ Москва ◆ Санкт-Петербург

УДК 681.3.06

Рассматриваются основы программирования на языке JavaScript, базовые объекты и методы, элементы форм, размещаемых на Web-страницах, методы решения задач. Приводятся сценарии и тексты HTML-кода. Изучаются способы представления знаний, понятие логического следствия, получение новых знаний из уже доказанных. Книга содержит примеры решения задач из различных областей: обработка символьной информации, численные расчеты, работа с изображениями, создание меню, обеспечение навигации по Web-документам.

Для программистов и Web-разработчиков

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Наталья Таркова*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Натальи Смирновой*
Корректор	*Наталия Першакова*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

Дмитриева М. В.
Самоучитель JavaScript — СПб.: БХВ-Петербург, 2001. — 512 с.: ил.

ISBN 5-94157-122-4

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 13.08.01.
Формат 70×100 1/16. Печать офсетная. Усл. печ. л. 41,28.
Тираж 5000 экз. Заказ №
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.
Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.
Отпечатано с диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.

Содержание

Введение

Предлагаемая книга является самоучителем по одному из разделов в области технологий разработки Web-приложений. При создании интерактивных документов в качестве языка сценариев используется язык JavaScript. В нем удачно сочетаются основные понятия современного программирования, практичность и наличие средств поддержки идей объектно-ориентированного программирования. При написании книги ставилась задача предоставить сведения для получения элементарных практических навыков по составлению сценариев, на простых и содержательных примерах продемонстрировать основные идеи и методы, принятые в современном программировании.

Язык сценариев JavaScript предназначен для создания интерактивных HTML-документов. Обычно ему отводится роль поддержки диалога с пользователем, применения для создания сценариев, обеспечивающих привлекательный вид Web-страниц. Но JavaScript прекрасно подходит и для обучения основам программирования. Центральным аспектом языка является понятие объекта, поэтому уже при начальном изучении основ программирования учащийся вводится в круг представлений о современных технологиях. Для студентов гуманитарной направленности знание языка в предложенном объеме является достаточным для использования его в профессиональной деятельности при подготовке Web-документов.

Язык JavaScript позволяет обрабатывать исходные данные, представленные с помощью различных элементов управления, создавать тестирующие программы, осуществлять контроль вводимых данных и многое другое. Можно создавать как итеративные, так и рекурсивные сценарии, использовать процедурный тип данных для решения многих классов задач, организовывать работу со стандартными объектами, обрабатывать текстовую информацию и решать другие задачи.

Каждый из теоретических разделов книги содержит большое количество примеров решения задач из различных областей: обработка символьной информации, численные расчеты, работа с изображениями, создание меню, обеспечение навигации по Web-документам и др. В частности, рассматриваются алгоритмы, которые обсуждаются в школьном курсе информатики, представлены записи этих алгоритмов на языке JavaScript.

В конце каждой главы приводятся упражнения, которые предлагается выполнить самостоятельно для закрепления рассмотренного материала. Материал книги построен таким образом, что новые понятия объясняются в тот момент, когда все ранее описанные средства уже образуют некоторую сис-

тему. Таким образом, в зависимости от наличия времени допустимо изучение материала в объеме нескольких тем.

Первая часть книги ориентирована на читателей, не знакомых с программированием, но желающих получить простые навыки, позволяющие в одних случаях сделать свой Web-документ более привлекательным. В других случаях требуется написать сценарий для простого анализа данных, например, для обработки анкеты. Для таких читателей важно уметь создавать меню, помещать в документ изображения, извлекать расположенные на Web-странице данные, которые могут представляться с помощью переключателей, флажков, списков, текстовых полей и т. д. Вычисления в аналогичных задачах несложные, поэтому и реализованные сценарии будут простыми. В этой части подробно описываются создаваемые документы и используемые сценарии. Такая степень детализации может быть излишней для следующей предполагаемой категории читателей.

Вторая часть книги предназначена для читателей, желающих ознакомиться с принципами современного программирования, научиться создавать более сложные сценарии. Спор о том, какой язык программирования выбрать в качестве базового, так окончательно не решен и, видимо, не будет решен никогда. Общепризнанным языком при обучении основам программирования является язык Паскаль. Так как любой язык — это способ записи алгоритмов, то целесообразно использовать в качестве базового языка JavaScript, по крайней мере, для тех категорий пользователей, которым не потребуется работать со сложными структурами данных типа деревьев, графов и т. д. Основное внимание обращается на методы решения различных задач, обсуждаются итерационные алгоритмы, алгоритмы обработки текстовой информации, классические алгоритмы работы с массивами. Рассматриваются задачи, при решении которых осуществляется работа с датами, применяются численные методы и др. Во второй части рассматриваются такие объекты JavaScript, как `String`, `Data`, `Array` и те основные методы, которые предусмотрены в языке сценариев для работы со стандартными объектами. В книге обсуждаются решения задач, которые рассматриваются, как правило, в курсах, посвященных основам программирования.

И, наконец, третья категория читателей — изучившие основы программирования и умеющие писать программы на каком-либо из распространенных процедурных языков, может обратиться к последней части книги. Таких читателей, как правило, интересуют лишь отдельные аспекты использования JavaScript. Например, процедурный тип данных и его реализация в языке JavaScript, методы работы со строками, которых нет в других языках программирования, особенности создания массивов и методы, характерные лишь для языка JavaScript, и т. п. Для них в третьей части рассматриваются рекурсивные методы решения задач, приводятся примеры сценариев, использующих рекурсивные функции, обсуждается метод исчерпывающего перебора и его реализация при решении конкретных задач, а также рассмат-

риваются задачи, при решении которых используется метод рекурсивного спуска.

Особо хочется обратить внимание читателей на последние две главы. Они посвящены решению задач, традиционно относящихся к искусственному интеллекту, — одной из активно развивающихся областей информатики. Способы представления знаний, понятие логического следствия, получение новых знаний из уже доказанных — неполный перечень тем, которые рассматриваются в этих главах. В книге приводится необходимый для понимания проблем теоретический материал. Р. Смаллиан написал ряд занимательных книг по математической логике [15]. Для иллюстрации теоретического материала и в качестве упражнений для самостоятельного решения используются, в основном, задачи из этих увлекательных книг.

Отдельная глава посвящена методам поиска доказательства. Рассмотрены различные стратегии поиска, сформулированы задания на разработку сценариев поиска решений. JavaScript располагает мощными средствами для работы с формулами, воспользоваться которыми рекомендуется при решении аналогичных задач.

При изучении языка сценариев JavaScript предполагается, что читатель знаком с основами языка HTML, хотя не исключается возможность "параллельного" изучения HTML и JavaScript. В книге, как правило, представлены не только функции JavaScript, решающие задачу, но полностью HTML-код документа. Приведенные примеры служат для закрепления навыков создания HTML-страниц, поэтому полезна не только функция, но весь документ. Кроме того, во многих задачах требуется создание документа определенной заданной структуры. Более того, многие сценарии на языке JavaScript призваны "оживить" HTML, т. е. цель создания сценария заключается в том, чтобы продемонстрировать, как будет меняться вид страницы при изменении значений параметров HTML-тегов. Вместе собранные и должным образом оформленные такого рода сценарии являются примером разработанного Web-приложения.

Для некоторых задач приводится несколько вариантов решения, каждый из них отличается набором тех средств, которые к данному моменту рассмотрены, и которые читатель может использовать при построении сценария.

Для закрепления пройденного материала рекомендуется выполнять упражнения, предложенные в конце каждой главы и подобранные таким образом, чтобы не вызывать затруднения у тех, кто внимательно изучил и понял материал очередной главы. Задачи, составляющие упражнения, как правило, имеют одинаковую сложность и могут использоваться в качестве индивидуальных заданий при обучении студентов в компьютерном классе.

При работе с книгой предполагается, что читатель знаком именно с основами HTML. Если читатель знает каскадные таблицы стилей, то некоторые

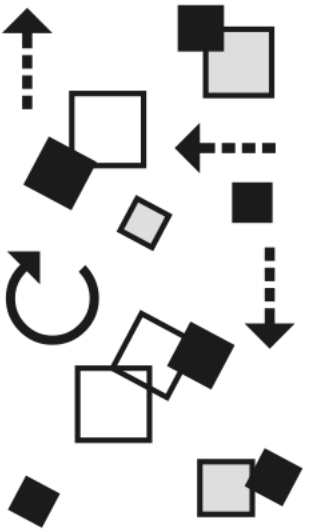
задачи можно решить, используя стили, иным способом, чем тот, что предложен в книге. Такие сценарии полезно написать в качестве упражнения.

Самоучитель разработан на основе опыта преподавания курса информатики для студентов математических и гуманитарных специальностей, для слушателей факультета повышения квалификации.

Я признательна студентам и аспирантам, которые не только прослушали мой курс лекций по технологии создания Web-публикаций, но и выполнили многочисленные индивидуальные задания и курсовые работы.

Я благодарна Сенину Владимиру Даниловичу и Керову Леониду Александровичу за предоставленную мне возможность работать по интересующей тематике. Хочу поблагодарить за сотрудничество Позднякова Сергея Николаевича, любезно согласившегося прочесть рукопись. Выражаю глубокую благодарность Лаврову Святославу Сергеевичу, впервые прочитавшему в университете курс лекций по искусственному интеллекту. Хочу выразить благодарность Рыбакову Евгению Евгеньевичу, после общения с которым возник замысел создания книги. Я признательна Дмитриеву Юрию Игоревичу за неоценимую помощь при работе над книгой и подготовке рукописи к публикации.

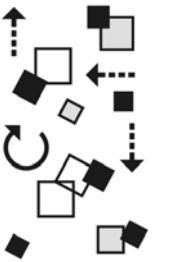
Надеюсь, что книга будет полезна школьникам, студентам младших курсов, слушателям курсов повышения квалификации и всем, кто интересуется вопросами создания и использования Web-приложений.



ЧАСТЬ I

Язык JavaScript для начинающих

Глава 1



Основные положения

Язык сценариев JavaScript

Язык программирования JavaScript разработан фирмой Netscape в сотрудничестве с Sun Microsystems и предназначен для создания интерактивных HTML-документов. Основные области использования JavaScript таковы:

- создание динамических страниц, т. е. страниц, содержимое которых может меняться после загрузки документа;
- проверка правильности заполнения пользователем форм до пересылки их на сервер;
- решение "локальных" задач с помощью сценариев

и некоторые другие сферы.

JavaScript позволяет создавать приложения, выполняемые как на стороне клиента, так и на стороне сервера. При разработке приложений обоих типов используется так называемое *ядро*, в котором содержатся определения стандартных объектов. Клиентские приложения выполняются браузером на машине пользователя.

Программа (сценарий) на языке JavaScript обрабатывается встроенным в браузер интерпретатором. Надо стремиться к тому, чтобы написанные сценарии корректно выполнялись в любом браузере. На первоначальном этапе обучения добиться удовлетворения этого требования сложно. Предлагаемые в книге сценарии отлаживались в Internet Explorer версии 4.01 и выше.

Программа (сценарий) на языке JavaScript представляет собой последовательность операторов. Если несколько операторов располагаются на одной строке, то между ними следует поставить знак "точка с запятой" (`;`). Если каждый оператор размещается на одной строке, то разделитель можно не писать. Один оператор может располагаться на нескольких строках.

Согласно принципам структурного программирования, программу рекомендуется записывать таким образом, чтобы в ней была отражена блочная структура. Это облегчает исследование программы и поиск ошибок.

В программах на JavaScript можно использовать комментарии. Для того чтобы задать комментарий, располагающийся на одной строке, достаточно перед его

текстом поставить две косые черты (//). Если же поясняющий текст занимает несколько строк, то его следует заключать между символами /* и */.

В JavaScript строчные и прописные буквы алфавита считаются разными символами.

Литералы

Простейшие данные, с которыми может оперировать программа, называются *литералами*. Литералы не могут изменяться. Литералы целого типа могут быть заданы в десятичном (по основанию 10), шестнадцатеричном (по основанию 16) или восьмеричном (по основанию 8) представлении.

Литерал целого типа в десятичном представлении записывается как последовательность десятичных цифр со знаком или без него, например, 15, 123, −156, +3567.

Шестнадцатеричные числа включают цифры 0—9 и буквы *a*, *b*, *c*, *d*, *e*, *f*. Шестнадцатеричные числа записываются с символами 0x перед числом, например, 0x25, 0xa1,0xff.

Восьмеричное число включает только цифры 0—7 и записывается, начиная с нуля, например, 03, 0543, 011.

Запись вещественного литерала отличается от записи вещественного числа в математике тем, что вместо запятой, отделяющей целую часть от дробной, указывается точка, например, 123.34, −22.56. Кроме того, для записи вещественных чисел можно использовать так называемую экспоненциальную форму, например, число 0,000000273, которое можно представить в виде 2.73×10^{-7}, в языке JavaScript записывается так: 2.73e−7. В такой записи знак умножения и число 10 заменяются символом e. В записи вещественного литерала должна присутствовать, по крайней мере, одна цифра и десятичная точка или символ экспоненты (e или E).

Кроме целых и вещественных значений в языке JavaScript могут встречаться так называемые логические значения. Существуют только два логических значения: истина и ложь. Первое представляется литералом true, второе — false. В некоторых реализациях JavaScript может быть использована единица в качестве true, и ноль в качестве false.

Строковый литерал представляется последовательностью символов, заключенной в одинарные или двойные кавычки. Примером строкового литерала может быть строка "результат" или 'результат'. Строковый литерал, представляющий пустую строку, обозначается как "" или ''.

Переменные

Переменные используются для хранения данных. Переменные в сценарии представляются с помощью идентификаторов. Идентификатор должен начинаться с буквы латинского алфавита, либо с символа подчеркивания. Далее может указываться последовательность, содержащая буквы латинского алфавита, цифры или знак подчеркивания, например, `test1`, `_my_test`, `test_1`. Тип переменной зависит от хранимых в ней данных, при изменении типа данных меняется тип переменной. Определить переменную можно с помощью оператора `var`, например:

```
var test1
```

В данном случае тип переменной `test1` не определен и станет известен только после присвоения переменной некоторого значения.

Оператор `var` можно использовать и для инициализации переменной, например, конструкцией

```
var test2=276
```

определяется переменная `test2` и ей присваивается значение `276`.

Значение переменной изменяется в результате выполнения оператора присваивания. Оператор присваивания может быть использован в любом месте программы и способен изменить не только значение, но и тип переменной. Оператор присваивания выглядит так

```
a=b
```

где `a` — переменная, которой мы хотим задать некоторое значение; `b` — выражение, определяющее новое значение переменной. Пусть в сценарии описаны следующие переменные

```
var n=3725
var x=2.75
var p=true
var s="Выполнение завершено"
```

Переменные `n` и `x` имеют тип `number`, тип переменной `p` — логический, переменная `s` имеет тип `string`. В JavaScript определен тип `function` для всех стандартных функций и функций, определяемых пользователем. Объекты JavaScript имеют тип данных `object`. Переменные типа `object` часто называют просто объектами, они могут хранить объекты.

Переменные, описанные в сценарии как в части `<HEAD>`, так и в части `<BODY>`, имеют одну и ту же область действия, доступны любому сценарию текущего документа. Такие переменные называются *глобальными* в отличие от *локальных* переменных, определенных в теле функции.

Выражения

Выражения строятся из литералов, переменных, знаков операций, скобок. В результате вычисления выражения получается единственное значение, которое может быть либо числом (целым или вещественным), либо строкой, либо логическим значением. Используемые в выражении переменные должны быть инициализированы. Если при вычислении выражения встречается неопределенная или неинициализированная переменная, то фиксируется ошибка. В JavaScript существует литерал `null` для обозначения неопределенного значения. Если переменной присвоено значение `null`, то она считается инициализированной.

Выражения формируются из операндов и обозначений операций. Например, в формуле `a*b` операндами являются `a` и `b`, обозначением операции — знак `*`.

Операции делятся на *унарные (одноместные)* или *бинарные (двуместные)*. Выражение записывается либо в виде ⊕A, если ⊕ — обозначение унарной операции, либо A⊕B, если ⊕ — обозначение бинарной операции. Вычисление выражения ⊕A сводится к вычислению операнда A и применению операции ⊕ к значению операнда. Вычисление выражения вида A⊕B состоит из следующих шагов:

1. Вычисляются A и B.
2. Операция ⊕ применяется к значению операндов, полученных на шаге 1.

В зависимости от типа вычисленного значения выражения можно разделить на арифметические, логические и строковые. Арифметические выражения получаются при выполнении операций, перечисленных в табл. 1.1.

Таблица 1.1. *Арифметические операции*

Операция	Название
+	Сложение
-	Вычитание
*	Умножение
/	Деление
%	Остаток от деления целых чисел
++	Увеличение значения операнда на единицу
--	Уменьшение значения операнда на единицу

Операторы в выражении вычисляются слева направо в соответствии с приоритетами арифметических операций. При необходимости с помощью скобок можно изменить порядок выполнения операций. В языке JavaScript определены операторы, в которых производятся арифметические действия над

левым и правым операндом и результат присваивается переменной, заданной левым операндом. Операции так называемой сокращенной формы присваивания представлены в табл. 1.2.

Таблица 1.2. *Сокращенные формы оператора присваивания*

Оператор	Эквивалентный оператор присваивания
`X += Y`	`X = X+Y`
`X -= Y`	`X = X-Y`
`X *= Y`	`X = X*Y`
`X /= Y`	`X = X/Y`
`X %= Y`	`X = X%Y`

Операции отношения применимы к операндам любого типа. Результат операции — логическое значение `true`, если сравнение верно, и `false` — в противном случае. Перечислим операции сравнения:

- `<` (меньше);
- `<=` (меньше или равно);
- `==` (равно);
- `!=` (не равно);
- `>=` (больше или равно);
- `>` (больше).

Операция `!` (логическое НЕ) применяется к операндам логического типа, если значение операнда `a` равно `true`, то значение выражения `!a` — `false`, если значение операнда `a` равно `false`, то значение выражения `!a` — `true`. Результат применения логических операций `&&` (логическое И) и `||` (логическое ИЛИ) приведен в табл. 1.3.

Таблица 1.3. *Логические операции*

A	B	A&&B	A||B
true	true	true	true
true	false	false	true
false	true	false	true
false	false	false	False

Значение выражения `A&&B` истинно, если истинны оба операнда, и ложно в противном случае. Значение выражения `A||B` истинно, если значение хотя бы одного из операндов истинно, и ложно в противном случае.

Над строковыми значениями определена операция *конкатенация (соединение) строк*. Обозначается операция знаком плюс. Результатом выполнения этой операции является строка, состоящая из строковых значений операндов, например, в результате выполнения оператора присваивания

```
st = "текущее "+"состояние"
```

переменная `st` получит значение `"текущее состояние"`.

Рассмотрим еще один пример. Пусть выполнено

```
st1 = "текущий "
st2 = "момент"
```

В результате выполнения

```
st1 += st2
```

переменная `st1` получит значение `"текущий момент"`.

Приоритет операций определяет порядок, в котором выполняются операции в выражении. В табл. 1.4 перечислены рассмотренные операции в порядке убывания приоритетов.

Таблица 1.4. *Таблица приоритетов операций*

Название	Обозначение
Инкремент	`++`
Декремент	`--`
Отрицание	`!`
Унарный минус	`-`
Умножение	`*`
Деление, остаток от деления	`/, %`
Сложение	`+`
Вычитание	`-`
Сравнение	`<, >, <=, >=`
Равенство	`==`
Неравенство	`!=`
Логическое И	`&&`
Логическое ИЛИ	`||`
Присваивание	`=, +=, -=, *=, /=, %=, !=`

Упражнения

1. Напишите выражение, истинное тогда и только тогда, когда:
   - значение целой переменной `M` делится нацело на значение целой переменной `K`;
   - значения вещественных переменных `A`, `B` и `C` образуют неубывающую последовательность;
   - значение переменной `X` является наибольшим из трех попарно различных значений `X`, `Y`, `Z`;
   - ни одна из логических переменных `A`, `B`, `C` не истинна;
   - по крайней мере одна из логических переменных `A`, `B`, `C` истинна.
2. Начертите и заштрихуйте область, такую, что заданное выражение, в котором значения `X` и `Y` трактуются как координаты точки на плоскости, принимает значение `true`:
   - `(X*Y>0)`;
   - `Y+X<5 && X*X+Y*Y>1`;
   - `Y<X*X+2 || Y>6`.
3. Напишите формулу, истинную тогда и только тогда, когда точка на плоскости с координатами `X` и `Y` попадает в заштрихованную область (рис. 1.1—1.4).

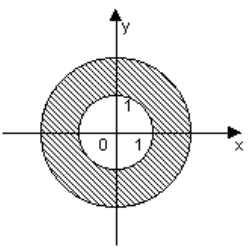


Рис. 1.1. Точка попадает в область, образованную исключением двух кругов

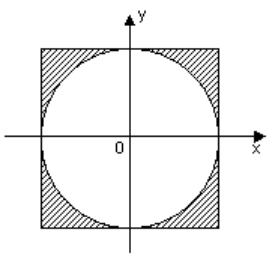


Рис. 1.2. Точка попадает в область, образованную исключением квадрата и круга

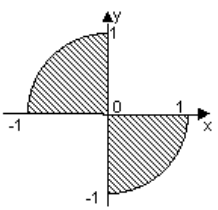


Рис. 1.3. Точка попадает в область, образованную двумя секторами

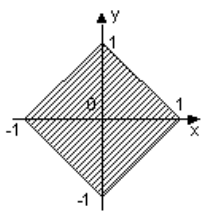


Рис. 1.4. Точка попадает в область, образованную ромбом

Сценарии в HTML-документе

Сценарии, написанные на языке JavaScript, могут располагаться непосредственно в HTML-документе между тегами <script> и </script>.

Одним из параметров тега <script> является language, который определяет используемый язык сценариев. Для языка JavaScript значение параметра равно "JavaScript". Если применяется язык сценариев VBScript, то значение параметра должно быть равным "VBScript". В случае использования языка JavaScript параметр language можно опускать, т. к. этот язык выбирается браузером по умолчанию.

Обычно браузеры, не поддерживающие какие-либо теги HTML, эти теги просто игнорируют. Попытка браузера проанализировать содержимое не поддерживаемых тегов может привести к неверному отображению страницы. Чтобы избежать такой ситуации, рекомендуется помещать операторы языка JavaScript в теги комментария <!-- ... -->. Для правильной работы интерпретатора перед закрывающим тегом комментария --> следует поставить символы //.

Итак, для размещения сценария в HTML-документе следует написать следующее:

```
<script language="JavaScript">
<!--
  Операторы языка JavaScript
//-->
</script>
```

Документ может содержать несколько тегов <script>. Все они последовательно обрабатываются интерпретатором JavaScript. В следующем примере в раздел <BODY> (в тело) HTML-документа вставлены операторы языка JavaScript.

Вычисление площади треугольника

Необходимо написать сценарий, определяющий площадь прямоугольного треугольника по заданным катетам. Сценарий разместим в разделе <BODY> HTML-документа (листинг 1.1).

Листинг 1.1. Первый сценарий в документе

```
<HTML>
  <HEAD>
    <TITLE>Первый сценарий в документе</TITLE>
  </HEAD>
```

```
  <BODY>
    <P>Страница, содержащая сценарий.</P>
    <script>
    <!--
      var a=8; h=10
      document.write("Площадь прямоугольного треугольника равна ",
a*h/2,".")
    //-->
    </script>
    <P>Конец формирования страницы, содержащей сценарий</P>
  </BODY>
</HTML>
```

В сценарии описываются и инициализируются две переменные, затем значение выражения записывается в документ. Для формирования вывода в HTML-страницу используется метод `write` объекта `document`. Строки, записываемые в документ, могут включать в себя теги HTML и выражения JavaScript.

Тег `<noscript>` определяет HTML-код, отображаемый на экране в случае, если JavaScript не поддерживается браузером или поддержка отключена. Этот тег следует после кода, заключенного в теги `<script>` и `</script>`. Если поддержка включена, то тег `<noscript>` игнорируется.

В дальнейших примерах будем считать, что поддержка JavaScript включена.

Функции: описание и использование

При создании программы разумно выделить в ней логически независимые части (так называемые *подпрограммы*). Каждую часть при необходимости можно разбить на отдельные подпрограммы и т. д. Разбиение программы на подпрограммы облегчает процесс отладки, т. к. позволяет отлаживать каждую подпрограмму отдельно. Имеет смысл распределить работу по созданию сложной программы между отдельными программистами. Некоторые подпрограммы можно использовать для решения разных задач.

Один раз созданную и отлаженную программу можно использовать произвольное число раз.

Во многих языках программирования понятие подпрограммы реализуется с помощью конструкций процедур, функций, модулей и т. п.

Основным элементом языка JavaScript является *функция*. Описание функции имеет вид

```
function F (V) {S}
```

где F — идентификатор функции, задающий имя, по которому можно обращаться к функции; V — список параметров функции, разделяемых запятыми; S — тело функции, в нем задаются действия, которые нужно выполнить, чтобы получить результат. Необязательный оператор return определяет возвращаемое функцией значение.

Описание функции не может быть вложено в описание другой функции. Параметры функции внутри ее тела играют ту же роль, что и обычные переменные, но начальные значения этим параметрам присваиваются при обращении к функции. Если описание функции имеет вид

```
function F (v1, v2, ..., vn) {S}
```

то вызов функции должен иметь вид

```
F (e1, e2, ..., en)
```

где e1, e2, ..., en — выражения, задающие *фактические* значения параметров. Параметры v1, v2, ..., vn, указанные в описании функции, называются *формальными* параметрами, чтобы подчеркнуть тот факт, что они получают смысл только после задания в вызове функции фактических параметров e1, e2, ..., en, с которыми функция затем и работает. Если в функции параметры отсутствуют, то описание функции имеет вид

```
function F (){S}
```

Наличие скобок в операторе вызова функции обязательно, т. е. вызов функции в этом случае должен иметь вид:

```
F()
```

Обычно все определения и функции задаются в разделе <HEAD> документа. Это обеспечивает интерпретацию и сохранение в памяти всех функций при загрузке документа в браузер.

В следующем примере в разделе заголовка описана функция care, вычисляющая площадь прямоугольного треугольника по заданным катетам.

Сценарий с функцией

Необходимо написать сценарий, определяющий площадь треугольника по заданному основанию и высоте. Для вычисления площади воспользуемся функцией, описанной в разделе <HEAD> HTML-документа (листинг 1.2).

Листинг 1.2. Использование сценария с функцией

```
<HTML>
  <HEAD>
    <TITLE>Использование сценария с функцией</TITLE>
    <script language="JavaScript">
```

```
    <!-- //
      function care (a, h)
      { return a*h/2 }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Начало отображения страницы со сценарием и функцией.</P>
    <script>
    <!--
      var a1=4; h1=16
      var s=care (a1,h1)
      document.write("При вызове функции получено значение ", s,".");
    //-->
    </script>
    <P>Конец формирования страницы.
  </BODY>
</HTML>
```

Тело функции состоит лишь из оператора `return`, который определяет возвращаемое функцией значение. Вызов функции осуществляется в теле документа при выполнении оператора присваивания: `s=care(a1,h1)`. Формальным параметрам `a` и `h` присваивается значение фактических параметров `a1` и `h1`, и выполняется тело функции. Полученное значение помещается в документ с помощью метода `write`.

Обработчики событий

В предыдущих примерах пользователю не предоставлялась возможность вводить значения, и в зависимости от них получать результат, например, при выполнении функции.

Интерактивные документы можно создавать, используя формы. Предположим, что мы хотим создать форму, в которой поля **Основание** и **Высота** служат для ввода соответствующих значений. Кроме того, в форме создадим кнопку **Вычислить**. При щелчке мышью по этой кнопке мы хотим получить значение площади треугольника.

Действие пользователя (например, щелчок кнопкой мыши) вызывает событие. События в основном связаны с действиями, производимыми пользователем с элементами форм HTML. Обычно перехват и обработка события задается в параметрах элементов форм. Имя параметра обработки события

начинается с приставки `on`, за которой следует имя самого события. Например, параметр обработки события `Click` будет выглядеть как `onClick`.

Значением параметра обработки события могут быть операторы языка JavaScript. В качестве значения параметра обработки события можно задать вызов функции, которая должна выполняться при возникновении события, определяемого параметром обработки события. В этом случае может быть использована следующая форма:

```
<FORM name="form1">
  Основание: <input type="text" size=5 name="st1"><hr>
  Высота:    <input type="text" size=5 name="st2"><hr>
  <input type="button" value=Вычислить
  onClick="care(document.form1.st1.value,document.form1.st2.value)">
</FORM>
```

Обработка значений из формы

Напишем функцию, вычисляющую площадь треугольника по заданному основанию и высоте. Создадим форму для ввода исходных данных.

HTML-код представлен в листинге 1.3.

Листинг 1.3. Обработка значений из формы

```
<HTML>
  <HEAD>
    <TITLE>Обработка значений из формы</TITLE>
    <script language="JavaScript">
    <!-- //
      function care (a, h)
      { var s= (a* h) / 2;
      document.write ("Площадь прямоугольного треугольника равна ",s);
      return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Пример сценария со значениями из формы</P>
    <FORM name="form1">
      Основание: <input type="text" size=5 name="st1"><hr>
      Высота:    <input type="text" size=5 name="st2"><hr>
```

```
    <input type="button" value=Вычислить
    onClick="care(document.form1.st1.value,document.form1.st2.value)">
   </FORM>
  </BODY>
</HTML>
```

Рассмотрим подробнее значение параметра обработки события, представляющего собой вызов функции `care`.

При интерпретации HTML-страницы браузером создаются объекты JavaScript. Взаимосвязь объектов между собой представляет иерархическую структуру. На самом верхнем уровне иерархии находится объект `windows`, представляющий окно браузера. Объект `windows` является *предком* или *родителем* всех остальных объектов. Каждая страница кроме объекта `windows` имеет объект `document`. Свойства объекта `document` определяются содержимым самого документа: цвет фона, цвет шрифта и т. д. В последнем примере на странице расположена форма. Форма (`form`) является *потомком* объекта `document`, а все элементы формы выступают потомками объекта `form`. Ссылка на объект может быть осуществлена по имени, заданному параметром `name` тега `<HTML>`. Для получения значения основания треугольника, введенного в первом поле формы, должна быть выполнена конструкция

```
document.form1.st1.value
```

При ссылке на формы и их элементы можно не указывать объект `document`. В рассмотренном примере получить значение первого поля ввода можно и следующим образом

```
form1.st1.value
```

Итак, когда в функцию передаются данные простых типов, например, чисел, как в рассмотренном случае, передача параметров осуществляется *по значению*. Формальному параметру `a` присваивается значение фактического параметра `form1.st1.value`, а формальному параметру `b` значение `form1.st2.value`. После этого выполняется тело функции.

Ситуация изменится, когда фактическим параметром функции станет объект. В этом случае говорят, что передача параметра осуществляется *по ссылке* или *по наименованию*. Пусть тело документа описано следующим образом:

```
<BODY>
  <P>Вычисление площади прямоугольного треугольника</P>
  <FORM name="form1">
    Основание: <input type="text" size=7 name="st1"><hr>
    Высота:    <input type="text" size=7 name="st2"><hr>
    <input type="button" value=Вычислить
```

```
          onClick="care1(form1.st1,form1.st2)">
  </FORM>
</BODY>
```

В качестве фактических параметров в вызове функции `care1` выступают имена текстовых полей формы. При вычислении площади используются значения, введенные пользователем. Поэтому описание функции `care1` должно быть таким:

```
function care1(a, h)
  {var s=(a.value * h.value)/2;
  document.write("Площадь равна ",s);
  return s}
```

Значение основания треугольника получается с помощью конструкции `a.value`, а высоты `h.value`. Приведем описание документа со сценарием полностью.

Передача параметров по ссылке

Напишем сценарий, определяющий площадь треугольника по заданному основанию и высоте. В качестве фактических параметров функции будем использовать имена текстовых полей формы.

HTML-код представлен в листинге 1.4.

Листинг 1.4. Передача параметров по ссылке

```
<HTML>
  <HEAD>
    <TITLE>Имена полей формы в качестве параметров</TITLE>
    <script language="JavaScript">
    <!-- //
      function care1 (a, h )
      { var s= (a.value* h.value) / 2;
      document.write ("Площадь равна ",s);
      return s}
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади треугольника</P>
    <FORM name="form1">
      Основание: <input type="text" size=7 name="st1"><hr>
      Высота:    <input type="text" size=7 name="st2"><hr>
```

```
      <input type="button" value=Вычислить
        onClick="care1(form1.st1,form1.st2)">
    </FORM>
  </BODY>
</HTML>
```

В следующем примере функции `care2` передается лишь один параметр — имя формы, а внутри функции определяются конкретные переданные значения.

Использование имени формы в качестве параметра функции

Напишем сценарий, определяющий площадь треугольника по заданному основанию и высоте. В качестве фактических параметров функции следует использовать имя формы.

Функция `care2` имеет один параметр `obj`, являющийся именем формы, в которой с помощью текстовых полей задаются пользователем значения. Для того чтобы использовать в вычислениях значения, заданные с помощью формы, требуется применить конструкцию `obj.st1.value`, т. е. указать имя поля формы, а затем выбрать значение (листинг 1.5).

Листинг 1.5. Параметр функции — имя формы

```
<HTML>
  <HEAD>
    <TITLE>Параметр функции - имя формы</TITLE>
    <script language="JavaScript">
    <!-- //
      function care2 (obj )
        {var a=obj.st1.value
        var h=obj.st2.value
        var s= (a* h) / 2;
        document.write ("Площадь равна ",s);
        return s }
      //-->
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади треугольника</P>
```

```
    <FORM name="form1">
      Основание: <input type="text" size=7 name="st1"><hr>
      Высота:    <input type="text" size=7 name="st2"><hr>
      <input type="button" value=Вычислить
        onClick="care2(form1)">
    </FORM>
  </BODY>
</HTML>
```

В предыдущих примерах вычислялось значение, и для его вывода применялся метод `write` объекта `document`. Определим в форме поле **Площадь**, в которое будем помещать вычисленное значение. Пусть в теле документа описана форма со следующими полями:

```
<FORM name="form1">
  Основание: <input type="text" size=7 name="st1"><hr>
  Высота:    <input type="text" size=7 name="st2"><hr>
  <input type="button" value=Вычислить
    onClick="care3(form1)">
  Площадь   <input type="text" size=7 name="res"><hr>
</FORM>
```

Опишем функцию `care3` следующим образом:

```
function care3 (obj)
  {var a=obj.st1.value
  var h=obj.st2.value
  var s= (a* h) / 2;
  obj.res.value=s}
```

В результате щелчка по кнопке **Вычислить** в поле с именем `res` будет помещено требуемое значение. В приведенных примерах значением параметра обработки события было имя функции, которая вызывалась, когда происходило событие. В общем случае значением параметра обработки события могут быть и другие операторы языка JavaScript.

В следующем примере мы не будем описывать функцию для вычисления площади треугольника. В результате щелчка мышью по кнопке **Вычислить** в поле `res` будет помещено вычисленное значение. Значением параметра обработки события в этом случае выступает оператор присваивания.

Оператор присваивания

Создадим программу, позволяющую получать площадь треугольника по заданному основанию и высоте (листинг 1.6).

Листинг 1.6. Использование оператора присваивания для вычисления значения параметра обработки события

```
<HTML>
  <HEAD>
  <TITLE>Оператор присваивания в качестве значения параметра
         обработки события</TITLE>
  <BODY>
    <P>Вычисление площади треугольника по основанию и высоте</P>
    <FORM name="form1">
      Основание: <input type="text" size=7 name="st1"><hr>
      Высота:    <input type="text" size=7 name="st2"><hr>
      <input type="button" value=Вычислить
       onClick="form1.res.value=(form1.st1.value* form1.st2.value)/2">
      Площадь   <input type="text" size=7 name="res"><hr>
    </FORM>
  </BODY>
</HTML>
```

Вычисление среднего дохода

Вводится информация о доходах за каждый месяц первого полугодия. Требуется написать сценарий определения среднего дохода в месяц за рассматриваемый период.

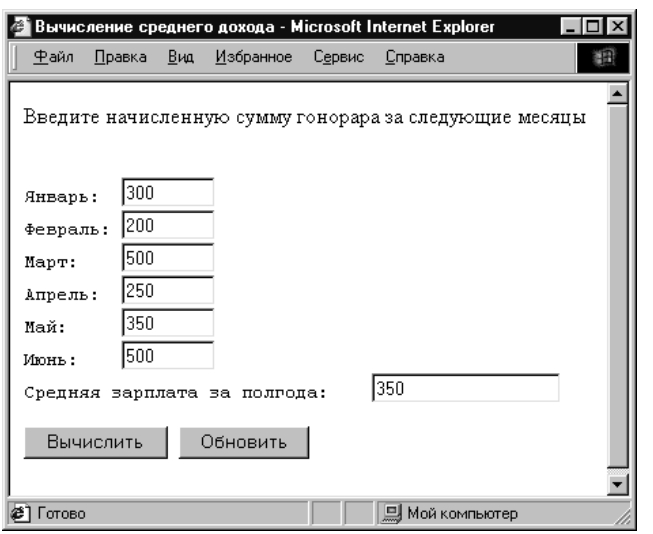


Рис. 1.5. Вычисление среднего дохода

На рис. 1.5 изображен документ после заполнения полей формы и вычисления результата.

Приведем документ со сценарием вычисления среднего дохода за рассматриваемый период (листинг 1.7).

Листинг 1.7. Вычисление среднего дохода

```
<HTML>
  <HEAD>
    <TITLE>Вычисление среднего дохода</title>
    <script language= "JavaScript">
    <!-- //
    function val(obj)
     {var a1= 1*obj.num1.value;
      var a2=1* obj.num2.value;
      var a3=1* obj.num3.value;
      var a4=1*obj.num4.value;
      var a5=1* obj.num5.value;
      var a6=1* obj.num6.value;
      var s =(a1+a2+a3+a4+a5+a6)/6
      obj.res.value = s
     }
    //-->
    </script>
  </HEAD>
  <BODY>
    Введите начисленную сумму гонорара за следующие месяцы
<pre>
<form name="form1">
Январь:  <input type="text" size=8 name="num1">
Февраль: <input type="text" size=8 name="num2">
Март:    <input type="text" size=8 name="num3">
Апрель:  <input type="text" size=8 name="num4">
Май:     <input type="text" size=8 name="num5">
Июнь:    <input type="text" size=8 name="num6">
Средняя зарплата за полгода:    <input type="text" size=18
name="res"><br>
<input type="button" value=Вычислить onClick="val(form1)"> <input
type="reset" value="Обновить">
</form>
```

```
</pre>
  </BODY>
</HTML>
```

В рассмотренных примерах параметр обработки события (`onClick`), задающий действия, выполняемые при обработке события, был связан с элементом типа "кнопка" (`button`). Событие, вызывающее обработку элементов форм — это щелчок мышью по кнопке **Вычислить**. В операторе задания переменной

```
var a1= 1*obj.num1.value
```

умножение на 1 выполняется для того, чтобы значением `a1` стало число, а не строка, т. к. в дальнейшем требуется осуществлять сложение чисел, а не строк.

Рассмотрим дополнительно некоторые другие события, применяемые к объекту "текстовое поле" (`text`).

Площадь квадрата

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда изменилось значение его стороны.

Зададим форму, в которой определены два текстовых поля: одно для длины стороны квадрата, другое для вычисленной площади. Кнопка **Обновить** очищает поля формы. Площадь квадрата вычисляется при возникновении события `Change`, которое происходит в тот момент, когда значение элемента формы с именем `num1` изменилось, и элемент потерял фокус.

HTML-код представлен в листинге 1.8.

Листинг 1.8. Реакция на событие `Change`

```
<HTML>
  <HEAD>
    <TITLE>Обработка события Change — изменение значения элемента</TITLE>
    <script>
      function srec(obj)
       { obj.res.value=obj.num1.value* obj.num1.value}
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади квадрата</P>
    <FORM name="form1">
```

```
      Сторона: <input type="text" size=7 name="num1"
        onChange="srec(form1)"><hr>
      Площадь:  <input type="text" size=7 name="res"><hr>
      <input type="reset" value=Обновить>
    </FORM>
  </BODY>
</HTML>
```

Обработка события *Focus*

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда пользователь переходит к элементу формы с помощью клавиши <Tab> или щелчка мыши.

При решении этой задачи вызов функции произойдет как реакция на событие `Focus`, параметр обработки события `onFocus`.

HTML-код представлен в листинге 1.9.

Листинг 1.9. Обработка события `Focus` — объект формы получает фокус

```
<HTML>
  <HEAD>
    <TITLE>Обработка события Focus — объект формы получает фокус</TITLE>
    <script>
      function srec()
       {form1.res.value=form1.num1.value* form1.num1.value}
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади квадрата</P>
    <FORM name="form1">
      Сторона:  <input type="text" size=7 name="num1"
                    value=6 onFocus="srec()"><hr>
      Площадь:  <input type="text" size=7 name="res"><hr>
      <input type="reset" value=Обновить>
    </FORM>
  </BODY>
</HTML>
```

При щелчке мышью по текстовому полю с именем `num1`, в поле с именем `res` помещается значение 36. Можно задать другое значение стороны квад-

рата и в тот момент, когда элемент формы num1 получит фокус, значение площади будет помещено в соответствующее поле.

Обработка события *Blur*

Событие "потеря фокуса" (Blur) происходит в тот момент, когда элемент формы теряет фокус. В следующем варианте решения задачи вычисления происходят в тот момент, когда поле формы, содержащее данные, потеряло фокус.

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда элемент формы теряет фокус.

HTML-код представлен в листинге 1.10.

Листинг 1.10. Обработка события Blur — потеря объектом фокуса

```
<HTML>
  <HEAD>
    <TITLE>Обработка события Blur - потеря объектом фокуса</TITLE>
    <script>
      function srec()
      {form1.res.value=form1.num1.value* form1.num1.value}
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади квадрата</P>
    <FORM name="form1">
      Сторона: <input type="text" size=7 name="num1" value=8
               onBlur="srec()"><hr>
      Площадь:  <input type="text" size=7 name="res"><hr>
      <input type="reset" value = Обновить>
    </FORM>
  </BODY>
</HTML>
```

В результате потери фокуса полем num1 происходит вычисление значения площади и помещение вычисленного значения в поле res. Первоначально будет вычислен результат, равный 64. Можно ввести нужное значение и при переходе к любому другому элементу, в поле res отобразится вычисленное значение.

Обработка события *Select*

Событие `Select` вызывается выбором части или всего текста в текстовом поле. Например, щелкнув дважды мышью по полю `num1`, мы выделим поле, наступит событие `Select`, обработка которого приведет к вычислению требуемого значения.

Напишем сценарий, определяющий площадь квадрата по заданной стороне. Площадь должна вычисляться в тот момент, когда выбирается часть или весь текст в текстовом поле.

HTML-код представлен в листинге 1.11.

Листинг 1.11. Обработка события `Select` — выбор поля ввода элемента формы

```
<HTML>
  <HEAD>
    <TITLE>Обработка события Select — выбор поля
          ввода элемента формы</TITLE>
    <script>
      function srec()
      {form1.res.value=form1.num1.value* form1.num1.value}
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление площади квадрата</P>
    <FORM name="form1">
      Сторона: <input type="text" size=7 name="num1" value=9
                    onSelect="srec()"><hr>
      Площадь:  <input type="text" size=7 name="res"><hr>
      <input type="reset" value=Обновить>
    </FORM>
  </BODY>
</HTML>
```

Перестановка двух изображений

В документе заданы два изображения. Требуется написать сценарий, который осуществляет перестановку заданных изображений.

Вид документа приведен на рис. 1.6. При нажатии на кнопку **Обменять** изображения меняются местами. Функция `chpict()` вызывается как реакция на

событие "щелчок по кнопке" **Обменять**. Изображения задаются в документе с помощью тегов. Значение параметра `src` определяет имя файла, в котором хранится изображение. Переменная `l` служит для того, чтобы запомнить имя файла, который связан с первым изображением. Оператор присваивания `d.pm1.src=d.pm2.src` обеспечит загрузку второго изображения вместо первого. И, наконец, со вторым тегом изображения будет связан тот файл, который первоначально относился к первому изображению. В результате описанных действий изображения в документе будут переставлены. HTML-код документа, осуществляющего обмен изображений, имеет вид, приведенный в листинге 1.12.

Рис. 1.6. Обмен двух изображений

Листинг 1.12. Обмен двух изображений

```
<HTML>
  <HEAD>
    <TITLE>Обмен двух изображений</TITLE>
    <script language="JavaScript">
    <!-- //
      function chpict ()
      { var d=document
        var l=d.pm1.src
        d.pm1.src=d.pm2.src
        d.pm2.src=l
      }
```

```
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=F8F8FF>
    <H3>Обмен двух изображений</H3>
    <IMG src="m1.gif" name=pm1 width=100>
    <IMG src="m2.gif" name=pm2 width=100>
    <FORM name="form1">
      <input type="button" value="Обменять" onClick="chpict()">
    </FORM>
  </BODY>
</HTML>
```

Итак, для любого события важно знать, когда оно наступает, и к каким объектам применяется. В предыдущих примерах мы привели варианты обработки событий, применяемых к текстовому полю `(text)`.

Вертикальное графическое меню

Создадим документ, реализующий вертикальное графическое меню. При наведении курсора мыши над пунктом меню должна меняться цветовая палитра, соответствующая выделенному пункту меню (рис. 1.7).

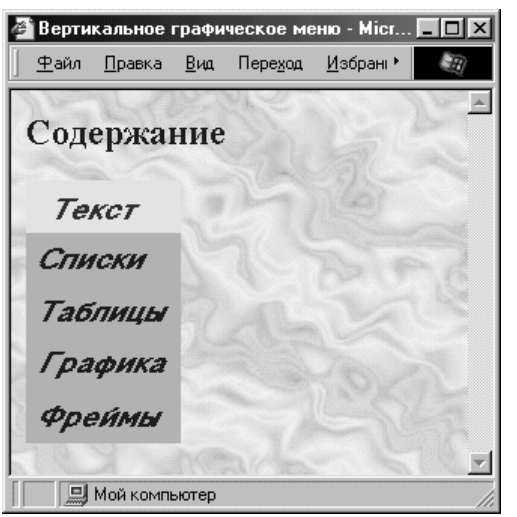


Рис. 1.7. Вертикальное графическое меню

Каждому пункту меню соответствует два изображения:

- первое изображение, когда пункт меню не выбран;
- второе — при выбранном пункте меню, цветовая палитра рисунка изменена.

Графические изображения, соответствующие ситуации, когда пункты меню не выбраны, хранятся в файлах с именами pch1.gif, pch2.gif, pch3.gif, pch4.gif, pch5.gif. Соответствующие им графические изображения с измененной палитрой хранятся в файлах с именами wpch1.gif, wpch2.gif, wpch3.gif, wpch4.gif, wpch5.

При перемещении курсора над пунктом меню возникает событие `onmouseover`. В этом случае при решении задачи требуется изменить графическое изображение, соответствующее выбранному пункту меню, что осуществляется в результате выполнения конструкции

```
onmouseover="document.pm1.src='wpch1.gif'"
```

Если курсор мыши выходит за пределы области связи, то возникает событие `onmouseout`, в результате обработки которого пункт меню должен принять первоначальный вид. Это достигается оператором присваивания

```
onmouseout="document.pm1.src='pch1.gif'"
```

HTML-код документа, реализующего графическое вертикальное меню, может иметь следующий вид, представленный в листинге 1.13, а.

Листинг 1.13, а. Вертикальное графическое меню (вариант 1)

```
<HTML>
  <HEAD>
    <TITLE>Вертикальное графическое меню</TITLE>
  </HEAD>
  <BODY background="fon1.jpg">
    <h2><font color="#0000FF">Содержание</font></h2>
    <A href="tch1.htm" target="Main"
       onmouseover="document.pm1.src='wpch1.gif'"
       onmouseout="document.pm1.src='pch1.gif'" >
       <IMG src="pch1.gif" name="pm1" alt="форматирование текста"
           border="0" width="103" height="35"></A><br>
    <A href="tch2.htm" target="Main"
       onmouseover="document.pm2.src='wpch2.gif'"
       onmouseout="document.pm2.src='pch2.gif'" >
       <IMG src="pch2.gif"  name="pm2" alt="создание списков" border="0"
            width="103" height="35"></A><br>
    <A href="tch3.htm" target="Main"
```

```
      onmouseover="document.pm3.src='wpch3.gif'"
      onmouseout="document.pm3.src='pch3.gif'" >
      <IMG src="pch3.gif" name="pm3" alt="построение таблиц" border="0"
           width="103" height="35"></A><br>
   <A href="tch4.htm" target="Main"
      onmouseover="document.pm4.src='wpch4.gif'"
      onmouseout="document.pm4.src='pch4.gif'" >
      <IMG src="pch4.gif"  name="pm4" alt="использование графики"
           border="0" width="103" height="35"></A><br>
   <A href="tch5.htm"
      onmouseover="document.pm5.src='wpch5.gif'"
      onmouseout="document.pm5.src='pch5.gif'" >
      <IMG src="pch5.gif" name="pm5" alt="создание фреймовой структуры"
           border="0" width="103" height="35"></A> <br>
  </BODY>
</HTML>
```

Событие `onmouseover` возникает и при перемещении курсора мыши над изображением. Событие `onmouseout` наступает при выходе курсора за пределы области изображения. Поэтому обработку события можно помещать не в тег `<A>`, как было сделано в предыдущем примере, а в тег `<IMG>`, как это сделано в листинге 1.13, б.

Листинг 1.13, б. Вертикальное графическое меню (вариант 2)

```
<HTML>
  <HEAD>
    <TITLE>Вертикальное графическое меню</TITLE>
  </HEAD>
  <BODY background="fon1.jpg">
    <H2><FONT color="#0000FF">Содержание</FONT></H2>
    <A href="tch1.htm" target="Main">
      <img src="pch1.gif" name="pm1" alt="форматирование текста"
           border="0" width="103" height="35"
           onmouseover="document.pm1.src='wpch1.gif'"
           onmouseout="document.pm1.src='pch1.gif'"></A><br>
    <A href="tch2.htm" target="Main">
      <img src="pch2.gif"  name="pm2" alt="создание списков" border="0"
           width="103" height="35"
           onmouseover="document.pm2.src='wpch2.gif'"
           onmouseout="document.pm2.src='pch2.gif'"></A><br>
```

```
    <A href="tch3.htm" target="Main">
      <img src="pch3.gif" name="pm3" alt="построение таблиц" border="0"
           width="103" height="35"
           onmouseover="document.pm3.src='wpch3.gif'"
           onmouseout="document.pm3.src='pch3.gif'"></A><br>
    <A href="tch4.htm" target="Main" >
      <IMG src="pch4.gif"  name="pm4" alt="использование графики"
           border="0"  width="103" height="35"
           onmouseover="document.pm4.src='wpch4.gif'"
           onmouseout="document.pm4.src='pch4.gif'"></A><br>
    <A href="tch5.htm" target="Main">
      <img src="pch5.gif" name="pm5" alt="создание фреймовой структуры"
           border="0" width="103" height="35"
           onmouseover="document.pm5.src='wpch5.gif'"
           onmouseout="document.pm5.src='pch5.gif'"></A> <br>
  </BODY>
</HTML>
```

Вид документа в обоих случаях будет одинаков.

Расписание занятий

Напишем сценарий, в результате работы которого при попадании курсора мыши на элемент списка, в текстовом поле появляется соответствующее сообщение.

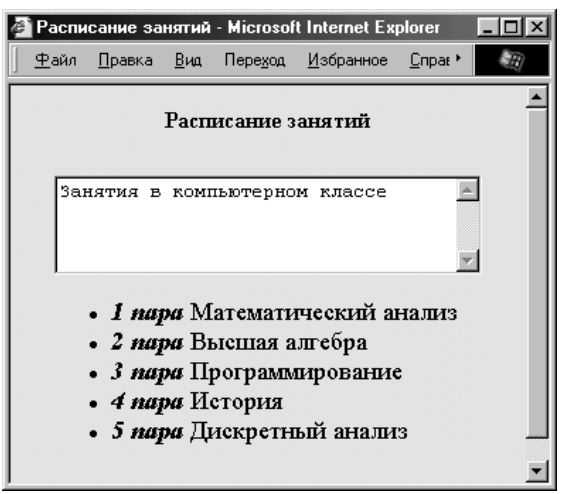


Рис. 1.8. Расписание занятий

Пусть список представляет собой расписание занятий в определенный день, а текст сообщения содержит комментарий к каждому занятию. Вид документа приведен на рис. 1.8.

Расписание занятий задается списком. В теге <LI>, определяющем элемент списка, предусмотрена реакция на два события, связанных с попаданием курсора мыши на элемент списка и выход курсора за пределы элемента списка. При попадании в текстовое поле помещается значение строки. Размещаемая в текстовом поле строка определяется параметром функции. HTML-код документа имеет вид, представленный в листинге 1.14.

Листинг 1.14. Расписание занятий

```
<HTML>
  <HEAD>
    <TITLE>Расписание занятий </TITLE>
    <script>
    <!--
      var s1="Контрольная работа"
      var s2='Изменилась аудитория 2246. 2 этаж'
      var s3='Занятия в компьютерном классе'
      var s4='Подготовка реферата'
      var s5='Занятия переносятся на следующую неделю'
      function sch(s)
       {document.form1.m1.value=s}
      function delet()
       {document.form1.m1.value=''}
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFDCDC">
    <Center>
    <H4>Расписание занятий</H4>
    <TABLE border=0 cellspacing=5 cellpadding=5>
      <TR valign=top>
        <TD align=center>
        <FORM name ="form1">
          <textarea name="m1" cols=35 rows=4></textarea></TD>
        </FORM>
      </TR>
      <TR valign=middle><TD>
```

```
        <UL><FONT size=4>
          <LI onmouseover="sch (s1)" onmouseout="delet()">
            <b><i>1 пара </i></b> Математический анализ
          <LI onmouseover="sch (s2)" onmouseout="delet()">
            <b><i>2 пара </i></b>Высшая алгебра
          <LI onmouseover="sch (s3)" onmouseout="delet()">
            <b><i>3 пара </i></b> Программирование
          <LI onmouseover="sch (s4)" onmouseout="delet()">
            <b><i>4 пара </i></b> История
          <LI onmouseover="sch (s5)" onmouseout="delet()">
            <b><i>5 пара </i></b> Дискретный анализ
          </FONT>
        </UL>
      </TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

В табл. 1.5 представлены события и элементы документов HTML, в которых эти события могут происходить.

Таблица 1.5. *События и объекты*

Событие	Объекты	Когда происходит событие
Abort	image	Отказ от загрузки изображения
Blur	windows, элементы формы	Потеря объектом фокуса
Change	text, textarea, select	Изменение значения элемента
Click	button, radio, checkbox, submit, reset, link	Щелчок на элементе или связи
DragDrop	windows	Перетаскивается мышью объект в окно браузера
Error	image, windows	Ошибка при загрузке документа или изображения
Focus	windows, элементы формы	Окно или элемент формы получает фокус
KeyDown	document, image, link, textarea	Нажатие клавиши клавиатуры
KeyPress	document, image, link, textarea	Удержание нажатой клавиши клавиатуры

Таблица 1.5 *(окончание)*

Событие	Объекты	Когда происходит событие
KeyUp	document, image, link, textarea	Отпускаются клавиши клавиатуры
Load	Тело документа	Загружается документ в браузер
MouseDown	document, button, link	Нажатие кнопки мыши
MouseOut	area, link	Перемещение курсора из области изображения или связи
MouseOver	link	Перемещение курсора над связью
MouseUp	document, button, link	Отпускается кнопка мыши
Move	windows	Пользователь или сценарий перемещает окно
Reset	form	Нажатие кнопки **Reset** формы
Resize	windows	Пользователь или сценарий изменяет размеры окна
Select	text, textarea	Выбирается поле ввода элемента формы
Submit	form	Нажатие кнопки **Submit** формы
Unload	Тело документа	Пользователь закрывает документ

Во многих рассмотренных примерах значением параметра обработки события был вызов функции, осуществляющей требуемые действия.

Объект *Math* и его методы

В языке JavaScript определены некоторые стандартные объекты и функции, пользоваться которыми можно без предварительного описания. Одним из стандартных объектов является объект `Math`. В свойствах упомянутого объекта хранятся основные математические константы, а его методы можно использовать для вызова основных математических функций. В табл. 1.6 приведены некоторые методы объекта `Math`.

Таблица 1.6. *Методы объекта* `Math`

Метод объекта	Описание метода
abs	Абсолютное значение
sin, cos, tan	Тригонометрические функции

Таблица 1.6 *(окончание)*

Метод объекта	Описание метода
`log`	Натуральный логарифм
`exp`	Экспонента
`min`	Наименьшее значение двух аргументов
`max`	Наибольшее значение двух аргументов
`pow`	Показательная функция
`sqrt`	Квадратный корень

Вычисление площади и периметра треугольника

Напишем сценарий вычисления площади и периметра треугольника, заданного длинами сторон.

Для того чтобы вычислить площадь треугольника по длинам сторон, можно воспользоваться формулой Герона, в соответствии с которой требуется применить функцию извлечения квадратного корня. Поэтому применим метод `sqrt` объекта `Math`: `Math.sqrt`.

HTML-код представлен в листинге 1.15.

Листинг 1.15. Вычисление площади и периметра треугольника с помощью объекта `Math`

```
<HTML>
  <HEAD>
    <TITLE>Объект Math</TITLE>
    <script language="JavaScript">
    <!-- //
      function care (obj)
       {var a=obj.st1.value
        var b=obj.st2.value
        var c=obj.st3.value
        var  s; p=a*1+b*1+c*1;
        document.writeln ("Периметр треугольника равен ",p,"<br>");
        p=p/2;
        s=Math.sqrt(p*(p-a)*(p-b)*(p-c));
        ocument.write ("Площадь треугольника равна ",s);
       }
```

```
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Пример сценария с математической функцией</P>
    <P>Вычисление площади и периметра треугольника</P>
    <FORM name="form1">
      Сторона 1: <input type="text" size=7 name="st1"><hr>
      Сторона 2: <input type="text" size=7 name="st2"><hr>
      Сторона 3: <input type="text" size=7 name="st3"><hr>
                 <input type="button" value=Вычислить
                        onClick="care(form1)"><hr>
                 <input type="reset" value=Отменить
    </FORM>
  </BODY>
</HTML>
```

Обратите внимание на кнопку **Отменить**, при нажатии на которую очищаются все поля формы.

Вычисление гиперболических функций

Напишите сценарий вычисления гиперболических функций

$$sh(x) = \frac{e^x - e^{-x}}{2},\ ch(x) = \frac{e^x + e^{-x}}{2},\ th(x) = \frac{sh(x)}{ch(x)}.$$

В заголовке документа определим три функции. Каждой из гиперболических функций соответствует кнопка, при нажатии на которую вычисляется значение функции в заданной точке.

HTML-код представлен в листинге 1.16.

Листинг 1.16. Описание и использование гиперболических функций

```
<HTML>
  <HEAD>
    <TITLE>Определение и использование нескольких функций</TITLE>
    <script language="JavaScript">
    <!-- //
      function sh(x)
       {var y=(Math.exp(x) - Math.exp(-x))/2; return y}
```

```
      function ch(x)
       {var y=(Math.exp(x) + Math.exp(-x))/2; return y}
      function th(x)
       {var y=sh(x)/ch(x); return y}
    //-->
    </script>
  </HEAD>
  <BODY>
    Определение и использование гиперболических функций
    <HR>
    <PRE>
<FORM name="form1">
Введите значение x:       <input type="text" size=5 name="arg"><br>
 гиперболический синус:   <input type="text" size=20 name="res1">
 <input type="button" value= "   синус  "
        onClick="form1.res1.value=sh(form1.arg.value)"><br>
 гиперболический косинус: <input type="text" size=20 name="res2">
 <input type="button" value= "косинус"
        onClick="form1.res2.value=ch(form1.arg.value)"><br>
 гиперболический тангенс: <input type="text" size=20 name="res3">
 <input type="button" value= "тангенс"
        onClick="form1.res3.value=th(form1.arg.value)">
</FORM>
</PRE>
  </BODY>
</HTML>
```

При работе с объектами можно использовать оператор `with`, который имеет следующий синтаксис:

```
with (t) {S}
```

где `t` — объект; `S` — последовательность операторов. Оператор `with` задает объект, используемый по умолчанию в последовательности операторов `S`. Все свойства и методы в `S` являются свойствами и методами объекта `t`. Применение этого оператора сокращает текст программы, т. к. избавляет от необходимости указывать иерархию объектов. Этот оператор часто используется для объекта `Math`, тогда обращение к его свойствам и методам можно производить без самого объекта `Math`, например, как в следующем варианте описания функции `sh(x)`:

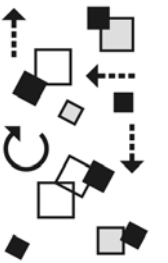
```
function sh(x)
  { var y
```

```
with (Math){y= (exp(x) — exp(-x))/2}
return y}
```

Упражнения

1. Вводится информация о доходах сотрудника за первый квартал текущего года. Требуется определить:
   - общую сумму дохода за квартал;
   - сумму подоходного налога (13%);
   - сумму, подлежащую выдаче на руки.
2. На плоскости заданы координаты трех точек. Напишите сценарий, который вычисляет площадь треугольника.
3. Напишите сценарий, который для точки, заданной координатами на плоскости, определяет расстояние до начала координат.
4. Напишите сценарий, который обменивает местами значения двух введенных переменных.
5. Напишите сценарий, который определяет объем шара и площадь боковой поверхности, если известен радиус.
6. Задан радиус окружности. Определите длину окружности и площадь соответствующего круга.
7. Задана окружность (координатами центра и радиусом) и точка вне окружности. Определите длину касательной из заданной точки к окружности.
8. Определите расстояние между двумя точками на плоскости, заданными своими координатами.

Глава 2



Организация ветвлений в программах

Условный оператор

При составлении программы часто необходимо выполнение различных действий в зависимости от результатов проверки некоторых условий. Для организации ветвлений можно воспользоваться условным оператором, который имеет вид:

```
if B {S1}
else {S2}
```

где `B` — выражение логического типа; `S1` и `S2` — операторы. Выполнение условного оператора осуществляется следующим образом. Вычисляется значение выражения `B`. Если оно истинно, то выполняются операторы `S1`, если ложно — операторы `S2`. Если последовательность операторов `S1` или `S2` состоит лишь из одного оператора, то фигурные скобки можно опустить. Возможна сокращенная форма условного оператора:

```
if B {S}
```

где `B` — выражение логического типа; `S` — последовательность операторов. Выполнение краткого условного оператора осуществляется так: вычисляется значение выражения `B`, если оно истинно, то выполняются операторы `S`.

Максимальное значение

Для трех заданных значений *a*, *b*, *c* необходимо написать сценарий, определяющий максимальное значение.

На рис. 2.1 приведен вид документа после введения данных и выполнения сценария, определяющего максимальное значение из введенных.

Поступим следующим образом. Сначала максимальным значением `m` будем считать значение `a`, далее значение `b` сравним с максимальным. Если окажется, что `b` больше `m`, то максимальным становится `b`. И, наконец, значение `c` сравнивается с максимальным значением из предыдущих значений `a` и `b`. Если `c` больше `m`, то максимальным становится `c`. Оператор присваивания `obj.res.value=m` обеспечивает запись вычисленного максимального

значения в соответствующее поле формы. Функция `Number(s)` преобразует объект `s`, заданный в качестве параметра, в число. Полностью сценарий может быть записан так, как представлено в листинге 2.1.

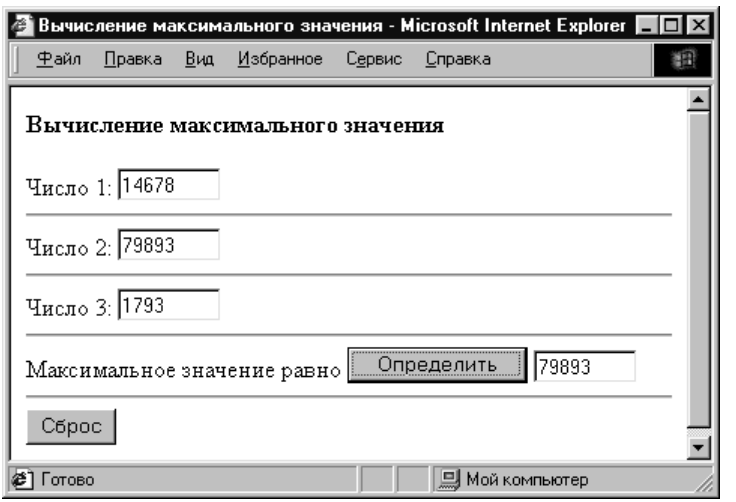


Рис. 2.1. Максимальное значение последовательности из трех элементов

Листинг 2.1. Вычисление максимального значения из трех заданных

```
<HTML>
  <HEAD>
    <TITLE>Вычисление максимального значения</TITLE>
    <script language="JavaScript">
    <!-- //
      function  maxval (obj )
       { var a = Number(obj.num1.value);
         var b = Number(obj.num2.value);
         var c = Number(obj.num3.value);
         var m=a
         if (b > m)   m=b
         if (c > m)   m=c
         obj.res.value=m
       }
    //-->
    </script>
```

```
  </HEAD>
  <BODY>
    <H4>Вычисление максимального значения</H4>
    <FORM name="form1">
      Число 1: <input type="text" size=8 name="num1"><hr>
      Число 2: <input type="text" size=8 name="num2"><hr>
      Число 3: <input type="text" size=8 name="num3"><hr>
      Максимальное значение равно
      <input type="button" value=Определить onClick="maxval(form1)">
      <input type="text"  size=8 name="res"><hr>
      <input type="reset">
    </FORM>
  </BODY>
</HTML>
```

Решим рассмотренную задачу другим способом. Вспомним, что стандартный объект `Math` имеет метод `max`, который определяет наибольшее значение двух аргументов. Опишем функцию `maxval1`, которая определяет максимальное значение из трех заданных значений и использует объект `Math`.

```
function  maxval1 (obj )
 {  var a = Number(obj.num1.value);
    var b = Number(obj.num2.value);
    var c = Number(obj.num3.value);
    obj.res.value=Math.max(Math.max(a,b),c)
}
```

Если бы требовалось определить максимальное из четырех заданных значений `a`, `b`, `c`, `d`, то можно было бы воспользоваться формулой

```
Math.max(Math.max(a,b), Math.max(c,d))
```

Максимальное и минимальное значения

Заданы три значения *a*, *b*, *c*. Требуется определить максимальное и минимальное значения из заданных величин.

Переменную `l` будем использовать для хранения минимального значения, переменную `t` — максимального. В результате выполнения условного оператора

```
if (a > b) {l= b; t = a} else {t = b; l = a}
```

переменная `l` получит минимальное значение из двух рассмотренных `a` и `b`, переменная `t` примет максимальное значение из значений `a` и `b`. Затем значение `c` сравнивается с максимальным значением. Если оказывается, что `c`

больше t, то максимальным становится c. Если же значение c меньше минимального l, то минимальным становится c. Приведем документ со сценарием решения задачи (листинг 2.2).

Листинг 2.2. Максимальное и минимальное из трех заданных значений

```
<HTML>
  <HEAD>
    <TITLE>Вычисление максимального и минимального  значений</TITLE>
    <script language="JavaScript">
    <!-- //
      function  maxminval (obj )
       { var a = Number(obj.num1.value);
         var b = Number(obj.num2.value);
         var c = Number(obj.num3.value);
         var l
         var t
         if (a > b) {l= b; t = a}
           else {t = b; l = a}
         if (b > t) t =b
         if (c < l)  l=c
         obj.resmax.value=t
         obj.resmin.value=l
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Вычисление максимального и минимального значений</H4>
    <FORM name="form1">
      Число 1: <input type="text" size=8 name="num1"><br>
      Число 2: <input type="text" size=8 name="num2"><br>
      Число 3: <input type="text"  size=8 name="num3"><hr>
      <input type="button" value=Вычислить
             onClick="maxminval(form1)"><hr>
      <input type="text"  size=8 name="resmax">максимальное значение<hr>
      <input type="text"  size=8 name="resmin">минимальное значение<hr>
      <input type="reset">
```

```
    </FORM>
  </BODY>
</HTML>
```

Стандартный объект `Math` содержит метод `min`, определяющий минимальное значение из двух заданных величин. Используя стандартный объект `Math` и его методы, опишем функцию `maxminv1`, определяющую максимальное и минимальное из трех заданных значений следующим образом.

```
function  maxminval1 (obj)
 { var a = Number(obj.num1.value);
   var b = Number(obj.num2.value);
   var c = Number(obj.num3.value);
   obj.resmax.value=Math.max(Math.max(a,b),c)
   obj.resmin.value=Math.min(Math.min(a,b),c)
}
```

Сортировка чисел

Заданы четыре значения *a*, *b*, *c*, *d*. Требуется написать сценарий, который сортирует эти числа в порядке возрастания.

Вводимые данные представлены так, как указано на рис. 2.2.

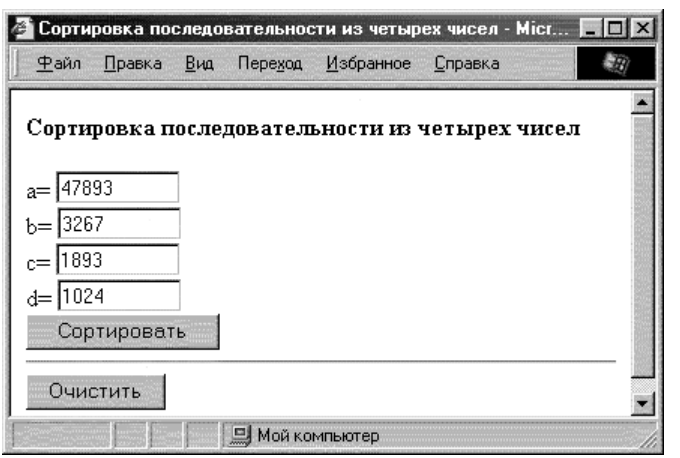


Рис. 2.2. Пример сортировки последовательности из четырех элементов

Сначала сравниваются значения двух переменных `a` и `b`. Если окажется, что `a` больше `b`, то переменные обмениваются значениями. Далее сравнивается значение `a` со значением `c`. Если значение `a` больше `c`, то опять эти переменные обмениваются значениями. И, наконец, сравниваются `a` и `d`, при

необходимости обмениваюся значениями. Таким образом, после описанных действий значение переменной a — минимальное из четырех рассмотренных. Далее значение b сравнивается последовательно со значением c и d, если необходимо, то осуществляется обмен значений. Значение b станет минимальным из рассмотренных b, c, d. Последний шаг — сравнение значений c и d. Полностью программа решения задачи может быть записана так, как представлено в листинге 2.3.

Листинг 2.3. Сортировка последовательности из четырех чисел

```
<HTML>
  <HEAD>
    <TITLE>Сортировка последовательности из четырех чисел</TITLE>
    <script language="JavaScript">
    <!-- //
      function  sortval(obj)
       { var a = Number(obj.num1.value);
         var b = Number(obj.num2.value);
         var c = Number(obj.num3.value);
         var d = Number(obj.num4.value);
         if (a > b) { r =a; a = b; b= r}
         if (a > c) { r =a; a = c; c= r}
         if (a > d) { r =a; a = d; d= r}
         if (b > c) { r =b; b = c; c= r}
         if (b > d) { r =b; b = d; d= r}
         if (c > d) { r = c; c = d;  d= r}
         obj.num1.value=a
         obj.num2.value=b
         obj.num3.value=c
         obj.num4.value=d
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Сортировка последовательности из четырех чисел</H4>
    <FORM name="form1">
      Число 1: <input type="text" size=10 name="num1"><hr>
      Число 2: <input type="text" size=10 name="num2"><hr>
      Число 3: <input type="text"  size=10 name="num3"><hr>
```

```
      Число 4: <input type="text"  size=10 name="num4"><hr>
      <input type="button" value=Сортировать
             onClick="sortval(form1)"><hr>
      <input type="reset"  value=Очистить>
    </FORM>
  </BODY>
</HTML>
```

После нажатия кнопки **Сортировать** будет выполнен сценарий, осуществляющий сортировку чисел по возрастанию, и документ примет вид как на рис. 2.3.

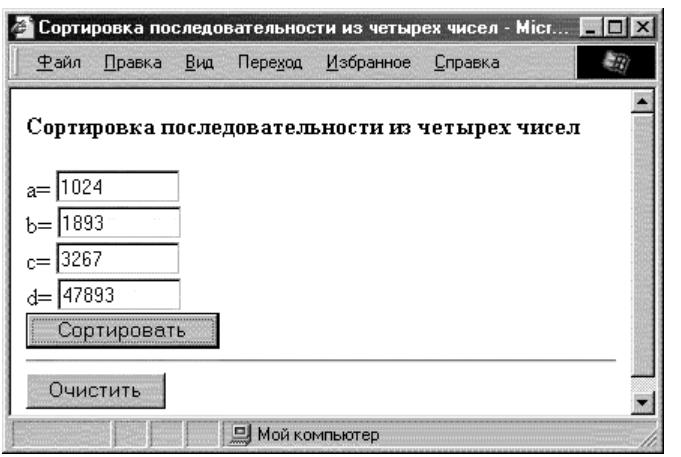


Рис. 2.3. Отсортированная последовательность

Вычисление размера стипендии

Требуется написать сценарий определения размера стипендии в зависимости от оценок, полученных во время сессии. Стипендия начисляется по следующим правилам:

- студенту, сдавшему экзамены на все оценки "отлично", начисляется стипендия в 100 у. е.;
- если студент получил четверки и пятерки (наличие хотя бы одной пятерки обязательно), то ему полагается стипендия в размере 75 у. е.;
- если все оценки четверки, то размер стипендии 50 у. е.;
- в остальных случаях стипендия не назначается.

При решении задачи поступим следующим образом. Вычислим максимальную и минимальную из оценок, полученных на экзамене. Далее будем ис-

следовать эти значения. Если окажется, что минимальная оценка равна 5, то это означает, что студент все экзамены сдал на оценку "отлично". Если максимальная из оценок равна 5, а минимальная 4, то студенту полагается стипендия в размере 75 у. е. Если и минимальная из оценок, и максимальная равны 4, то все экзамены сданы на 4, поэтому размер назначаемой стипендии — 50 у. е. В остальных случаях стипендия не назначается. Документ с введенными оценками и результатом их анализа представлен на рис. 2.4.

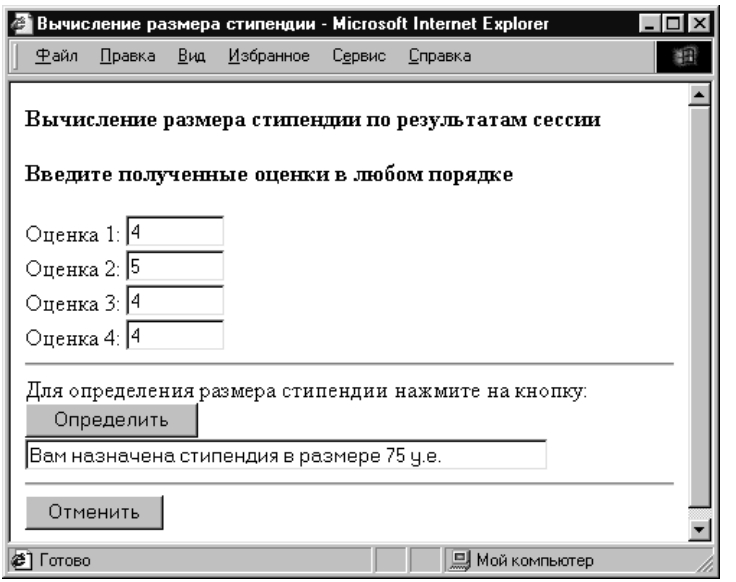


Рис. 2.4. Вычисление размера стипендии

Приведем документ и текст сценария решения задачи (листинг 2.4).

Листинг 2.4. Вычисление размера стипендии

```
<HTML>
  <HEAD>
    <TITLE>Вычисление размера стипендии</TITLE>
    <script language="JavaScript">
    <!-- //
      function  st (obj )
       { var a = Number(obj.num1.value);
         var b = Number(obj.num2.value);
         var c = Number(obj.num3.value);
         var d = Number(obj.num4.value);
```

```
        var l;   var t
        var m = 0
        if (a > b) {l= b; t = a}
          else {t = b; l = a}
        if (b > t) t =b;  if (c > t) t =c;  if (d > t) t =d;
        if (b < l) l=b;   if (c < l)  l=c;  if (d < l)  l=d;
        if (l ==5) {m=100}
        else
          if ((t ==5)&& (l ==4))  {m=75}
          else
            if ((t ==4)&& (l ==4))  {m=50}
        if (m==0)
        obj.stip.value="К сожалению, Вам стипендия не назначена"
        else
         obj.stip.value="Вам назначена стипендия в размере "+m+" у.е."
      }
  //-->
  </script>
 </HEAD>
 <BODY>
  <H4>Вычисление размера стипендии по результатам сессии</H4>
  <H4>Введите полученные оценки в любом порядке</H4>
  <FORM name="form1">
    Оценка 1: <input type="text" size=8 name="num1"><br>
    Оценка 2: <input type="text" size=8 name="num2"><br>
    Оценка 3: <input type="text" size=8 name="num3"><br>
    Оценка 4: <input type="text" size=8 name="num4"><hr>
    Для определения размера стипендии нажмите на кнопку:<br>
    <input type="button" value=Определить onClick="st(form1)"><br>
    <input type="text"  size=50 name="stip"><hr>
    <input type="reset" value=Отменить>
  </FORM>
 </BODY>
</HTML>
```

Как и в предыдущих случаях, функцию `st` можно описать по-другому, используя свойства стандартного объекта `Math`. Приведем описание функции `st1`.

```
function  st1 (obj )
 { var a = Number(obj.num1.value);
```

```
  var b = Number(obj.num2.value);
  var c = Number(obj.num3.value);
  var d = Number(obj.num4.value);
  var l = Math.max(Math.max(a,b),Math.max(c,d))
  var t  = Math.min(Math.min(a,b), Math.min(c,d))
  var m = 0
  if (l ==5) {m=100}
  else
    if ((t ==5)&& (l ==4))  {m=75}
      else
        if ((t ==4)&& (l ==4))  {m=50}
   obj.stip.value=m
}
```

Расположение точки относительно треугольника

Точка на плоскости задается своими координатами. Треугольник на плоскости — координатами вершин. Необходимо написать сценарий, определяющий, лежит ли заданная точка внутри треугольника.

Прежде чем писать программу, сделаем несколько замечаний. Площадь треугольника с вершинами A, B, C обозначим S_{ABC}. При решении задачи будем использовать тот факт, что, если точка P лежит внутри треугольника с вершинами A, B, C или на его стороне, то выполняется равенство (рис. 2.5):

$$S_{ABC} = S_{APB} + S_{BPC} + S_{APC}.$$

Если же точка P лежит вне треугольника (рис. 2.6), то последнее равенство будет нарушено.

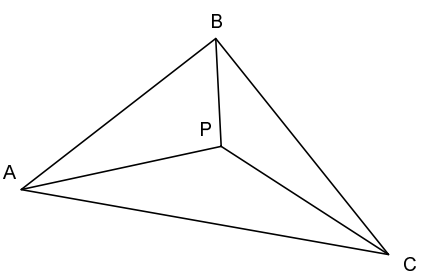


Рис. 2.5. Точка P внутри треугольника

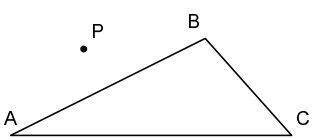


Рис. 2.6. Точка P вне треугольника

Опишем функцию `care`, которая вычисляет площадь треугольника, если заданы координаты его вершин.

```
function  care (ax,ay,bx,by,cx,cy)
  { var   s = Math.abs(ax*(by-cy)+bx*(cy-ay)+cx*(ay-by)); return s}
```

Работая с величинами вещественного типа, необходимо помнить, что вещественные числа представляются в памяти компьютера приближенно, поэтому для них следует избегать проверки равенства $P = Q$, а надо использовать выражение $|P - Q| < eps$, где *eps* некоторое заранее заданное маленькое число. Поэтому в программе после вычисления площадей

```
var s= care (ax,ay,bx,by,cx,cy)
var s1=care (ax,ay,bx,by,x,y)
var s2=care (ax,ay,x,y,cx,cy)
var s3=care (x,y,bx,by,cx,cy)
```

будет проверяться условие

```
Math.abs(s-s1-s2-s3)< 0.001
```

т. к. величины `s`, `s1`, `s2`, `s3` вещественного типа. Приведем документ со сценарием, решающим задачу (листинг 2.5).

Листинг 2.5. Расположение точки относительно треугольника

```
<HTML>
  <HEAD>
    <TITLE>Точка внутри треугольника</TITLE>
    <script language="JavaScript">
    <!-- //
      function  care (ax,ay,bx,by,cx,cy)
       { var   s = Math.abs(ax*(by-cy)+bx*(cy-ay)+cx*(ay-by))
         return s
       }
      function  poinpoly (obj)
       { var x= Number(obj.x.value);   var y= Number(obj.y.value)
         var ax= Number(obj.ax.value); var ay= Number(obj.ay.value)
         var bx= Number(obj.bx.value); var by= Number(obj.by.value)
         var cx= Number(obj.cx.value); var cy= Number(obj.cy.value)
         var s=care (ax,ay,bx,by,cx,cy)
         var s1=care (ax,ay,bx,by,x,y)
         var s2=care (ax,ay,x,y,cx,cy)
         var s3=care (x,y,bx,by,cx,cy)
         if (Math.abs(s-s1-s2-s3) <= 0.001)
           obj.res.value="точка лежит внутри треугольника"
```

```
          else
            obj.res.value="точка  не лежит внутри треугольника"
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Лежит ли точка внутри заданного треугольника?</H4>
    <FORM name="form1">
    <PRE>
Координаты первой вершины треугольника:
x=<input type="text" size=7 name="ax"> y=<input type="text" size=7
name="ay">
Координаты второй вершины треугольника:
x=<input type="text" size=7 name="bx"> y=<input type="text" size=7
name="by">
Координаты третьей вершины треугольника:
x=<input type="text" size=7 name="cx"> y=<input type="text" size=7
name="cy"><hr>
Введите координаты точки:
x=<input type="text" size=7 name="x"> y=<input type="text" size=7
name="y"><hr>
<input type="button" value=Вычислить onClick=" poinpoly(form1)"><hr>
Результат: <input type="text" size=40 name="res"><hr>
<input type="reset" value=Отменить
    </FORM>
  </BODY>
</HTML>
```

Точка внутри области треугольника

Треугольник ABC на плоскости задан координатами вершин. Построим еще один треугольник, соединив середины сторон данного треугольника отрезками (рис. 2.7). Напишите сценарий, проверяющий, принадлежит ли заданная точка одному из заштрихованных треугольников.

Будем использовать функцию

```
poinpoly(ax,ay,bx,by,cx,cy,x,y)
```

которая проверяет, лежит ли точка с координатами (x, y) внутри треугольника, задаваемого координатами вершин. В программе необходимо найти координаты точек D, E, F, зная координаты точек A, B, C. Выяснить, при-

надлежит ли точка заштрихованной области можно, проверив значение логического выражения

```
( poinpoly (ax,ay,dx,dy,fx,fy,x,y) ||
  poinpoly (dx,dy,bx,by,ex,ey,x,y) ||
  poinpoly (fx,fy,ex,ey,cx,cy,x,y))
```

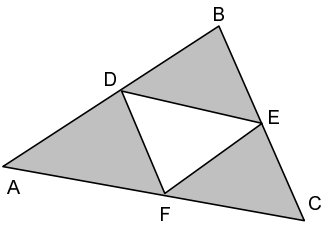


Рис. 2.7. Вложенные треугольники

Если точка принадлежит одному из заштрихованных треугольников, то значение выражения равно `true`; если же точка ни одному из треугольников не принадлежит, то значение выражения — `false`. Однако число вызовов функции `poinpoly (ax,ay,bx,by,cx,cy,x,y)` можно сократить, заметив, что точка принадлежит заштрихованной области, если она лежит внутри треугольника с вершинами A, B, C, и вне треугольника с вершинами E, D, F, или, другими словами, когда истинно выражение

```
( poinpoly (ax,ay,bx,by,cx,cy,x,y) &&
  ! ( poinpoly (fx,fy,dx,dy,ex,ey,x,y))
```

В документе оставлены поля, выдающие дополнительную информацию о принадлежности точки каждому из заштрихованных треугольников. Результат решения приведен в листинге 2.6.

Листинг 2.6. Точка внутри заштрихованной области треугольника

```
<HTML>
  <HEAD>
    <TITLE>Точка внутри заштрихованной области треугольника </TITLE>
    <script language="JavaScript">
    <!-- //
      function  care (ax,ay,bx,by,cx,cy)
       { var   s = Math.abs(ax*(by-cy)+bx*(cy-ay)+cx*(ay-by)); return s }
      function  poinpoly (ax,ay,bx,by,cx,cy,x,y)
       { var s=care (ax,ay,bx,by,cx,cy)
         var s1=care (ax,ay,bx,by,x,y)
```

```
        var s2=care (bx,by,x,y,cx,cy)
        var s3=care (x,y,ax,ay,cx,cy)
        var p=false
        if (Math.abs(s-s1-s2-s3)<0.1) {p= true}
        return p
      }
    function  pp1 (obj)
     { var x  =Number(obj.x.value);   var y=Number(obj.y.value)
       var ax=Number(obj.ax.value); var ay=Number(obj.ay.value)
       var bx=Number(obj.bx.value); var by=Number(obj.by.value)
       var cx=Number(obj.cx.value); var cy=Number(obj.cy.value)
       var dx=  (ax + bx)/2; var dy=  (ay + by)/2
       var ex=  (bx + cx)/2; var ey=  (by + cy)/2
       var fx=  (ax + cx)/2; var fy=   (ay + cy)/2
       obj.test.value  =  poinpoly (ax,ay,bx,by,cx,cy,x,y)
       obj.test1.value=  poinpoly (ax,ay,dx,dy,fx,fy,x,y)
       obj.test2.value=  poinpoly (dx,dy,bx,by,ex,ey,x,y)
       obj.test3.value=  poinpoly (fx,fy,ex,ey,cx,cy,x,y)
       obj.test4.value=  poinpoly (fx,fy,dx,dy,ex,ey,x,y)
       var rest1="точка вне заштрихованной области"
       if( poinpoly (ax,ay,dx,dy,fx,fy,x,y) ||
           poinpoly (dx,dy,bx,by,ex,ey,x,y) ||
           poinpoly (fx,fy,ex,ey,cx,cy,x,y)  )
       rest1="точка принадлежит заштрихованной области"
       obj.res1.value = rest1
       var rest2="точка вне заштрихованной области"
       if (poinpoly(ax,ay,bx,by,cx,cy,x,y) && !
          (poinpoly(fx,fy,dx,dy,ex,ey,x,y)))
       rest2 = "точка принадлежит заштрихованной области"
       obj.res2.value =   rest2
     }
  //-->
  </script>
</HEAD>
<BODY>
  <H4>Определение, лежит ли точка внутри заштрихованной области</H4>
  <FORM name="form1">
    <pre>
```

```
Координаты первой вершины треугольника:
x=<input type="text" size=7 name="ax"> y=<input type="text" size=7
name="ay">
Координаты второй вершины треугольника:
x=<input type="text" size=7 name="bx"> y=<input type="text" size=7
name="by">
Координаты третьей вершины треугольника:
x=<input type="text" size=7 name="cx"> y=<input type="text" size=7
name="cy"><hr>
Введите координаты точки:
x=<input type="text" size=7 name="x"> y=<input type="text" size=7
name="y"><hr>
<input type="button" value=Вычислить onClick=" pp1(form1)"><hr>
Первый вариант решения:  <input type="text" size=50 name="res1">
Второй вариант решения:  <input type="text" size=50 name="res2">
Тесты
Внутри ABC: <input type="text" size=20 name="test">
Внутри ADF: <input type="text" size=20 name="test1">
Внутри DBE: <input type="text" size=20 name="test2">
Внутри FEC: <input type="text" size=20 name="test3">
Внутри DFE: <input type="text" size=20 name="test4">
 <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Циклическая смена изображений

Необходимо написать сценарий, в результате выполнения которого несколько заданных изображений последовательно появляются в документе через равные промежутки времени. При нажатии на кнопку **Остановить** чередование изображений прекращается. Возобновить просмотр рисунков можно, если нажать кнопку **Начать снова** (рис. 2.8).

Событие `load` возникает в тот момент, когда обозреватель заканчивает загрузку окна. Как реакция на событие `load` вызывается функция `succpict()`. Считается, что файлы с изображениями хранятся в той же папке, что и документ со сценарием.

Метод `setTimeout` выполняет действие, задаваемое первым параметром, по истечении указанного в миллисекундах промежутка времени, определенного вторым параметром. В рассмотренном примере в качестве первого параметра задается функция `succpict()`, тем самым обеспечивается повторение вызова функции через каждые две секунды. Просмотр изображений будет пре-

кращен после нажатия кнопки **Остановить**. Если просмотр изображений требуется продолжить, то следует щелкнуть по кнопке **Начать снова**.

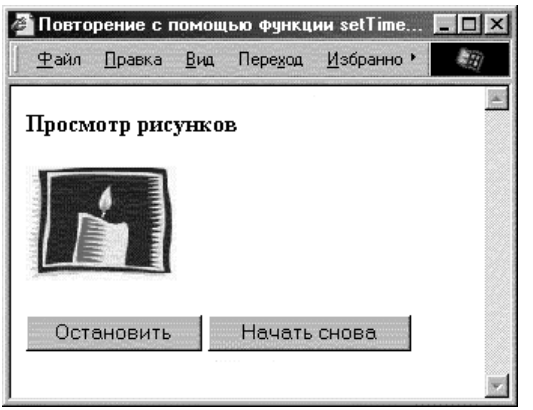


Рис. 2.8. Чередование рисунков

Мы уже неоднократно пользовались тем фактом, что страница содержит объекты, зависящие от ее содержания, и эти объекты являются наследниками объекта `document`. Ссылка на объект может быть осуществлена по имени, заданному параметром `name` тега `<HTML>`. Изображение в документе может быть задано с помощью тега `<IMG>`, например, так:

```
<IMG src="m1.gif" name="mypict">
```

Доступ к объекту `image` в этом случае может быть осуществлен следующим образом:

```
document.mypict
```

Документ с описанным сценарием выглядит так, как представлено в листинге 2.7.

Листинг 2.7. Использование функции `setTimeout`

```
<HTML>
  <HEAD>
    <TITLE>Повторение с помощью функции setTimeout</TITLE>
    <script language="JavaScript">
    <!-- //
      var k=1
      function ref ()
       {k=5}
      function  succpict ()
```

```
      { var d= document
        if (k<=4)
         {if (k==1)
           {d.mypict.src="m1.gif"; k++}
           else
            if (k==2)
              {d.mypict.src="m2.gif"; k++}
            else
              if (k==3)
               {d.mypict.src="m3.gif"; k++}
              else
                if (k==4)
                 {d.mypict.src="m4.gif"; k=1}
          setTimeout("succpict()", 2000)
         }
      }
    //-->
    </script>
  </HEAD>
  <BODY  onLoad="succpict ()">
    <P>Просмотр рисунков</P>
    <IMG src="m1.gif"  name=mypict width=100>
    <FORM name="form1">
      <input type="reset" value=Остановить onClick=ref()>
      <input type="button" value="Начать снова" onClick="k=1;
             succpict()">
    </FORM>
  </BODY>
</HTML>
```

Смена изображений при наведении указателя мыши

Напишем сценарий, во время работы которого смена рисунков происходит при наведении курсора мыши на изображение.

При перемещении пользователем курсора мыши над изображением возникает событие `MouseOver`. Опишем функцию `succpict()`, которая будет реакцией на это событие, и определит, какое изображение следует поместить в документ. Воспользуемся тем фактом, что загружаемые изображения хранятся в файлах с именами m1.gif, ..., m4.gif. Для загрузки *k*-го изображения

формируется имя файла по формуле "m"+k+".gif". Приведем сценарий в листинге 2.8.

Листинг 2.8. Смена изображений при попадании курсора на рисунок

```
<HTML>
  <HEAD>
    <TITLE>Смена рисунков при перемещении курсора мыши
          над изображением</TITLE>
    <script language="JavaScript">
    <!-- //
      var k=1
      function  succpict ()
       { var d= document
         if (k < 4)
           k=k+1
         else k=1
         d.mypict.src="m"+k+".gif"
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Для смены изображения поместите курсор мыши над рисунком.</H4>
    <IMG src="m1.gif" name=mypict width=150 onMouseOver="succpict()">
  </BODY>
</HTML>
```

Эффект визуального удаления изображения

Напишем сценарий, во время работы которого при наведении курсора мыши на изображение оно начинает удаляться от зрителя, уменьшаясь в размерах.

При решении такой задачи воспользуемся свойством `width` объекта `image`. При каждом вызове функции `succpict()` изменяется размер выводимого изображения и этим достигается эффект удаления от зрителя. Функция `setTimeout("succpict()",500)` производит повторный вызов функции `succpict()` через каждые полсекунды. Когда размер изображения уменьшится до заданного, движение прекратится. В начальный момент документ имеет вид как на рис. 2.9.

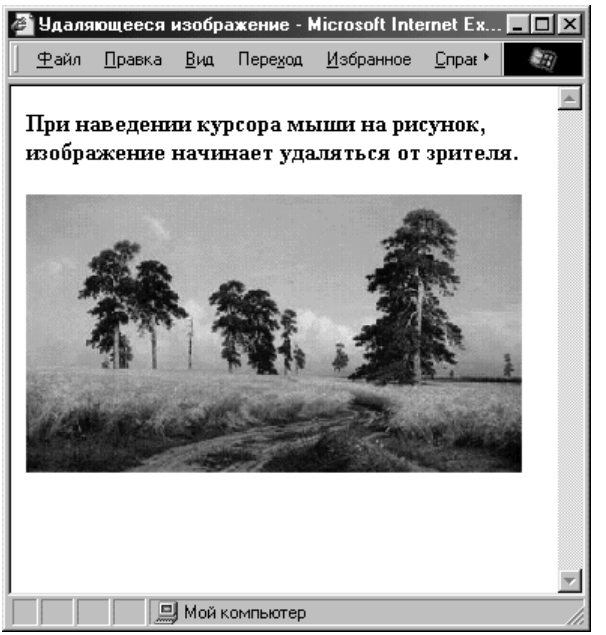


Рис. 2.9. Эффект удаления от зрителя

Для каждого рисунка параметры уменьшения размера изображения и время обновления следует подобрать индивидуально. Сценарий описан в листинге 2.9.

Листинг 2.9. Эффект удаления изображения от зрителя

```
<HTML>
  <HEAD>
    <TITLE>Удаляющееся изображение</TITLE>
    <script language="JavaScript">
    <!-- //
      function  succpict ()
       { var d=document
         var w=d.mypict.width
         if (w>150)
          {d.mypict.width=w-10
           d.mypict.src="msm.jpg"
           setTimeout("succpict()", 500)
          }
       }
    //-->
```

```
    </script>
  </HEAD>
  <BODY>
    <h4>При наведении курсора мыши над рисунком
        изображение начинает удаляться от зрителя.</H4>
    <IMG src="msm.jpg"  name=mypict  onMouseOver="succpict()">
  </BODY>
</HTML>
```

Эффект визуального приближения изображения

Напишем сценарий, при выполнении которого заданное изображение начинает увеличиваться, т. е. моделируется эффект приближения изображения.

Эта задача в некотором смысле является обратной к только что рассмотренной. Будем увеличивать при каждом вызове функции размер изображения до тех пор, пока оно не достигнет заданного размера. Повторные вызовы функции `grpict()` обеспечиваются функцией `setTimeout`, параметры которой следует подобрать в зависимости от изображения. В начальный момент документ имеет вид как на рис. 2.10.

Текст сценария и документ, его содержащий, приведены в листинге 2.10.

Листинг 2.10. Эффект приближения изображения

```
<HTML>
  <HEAD>
    <TITLE>Увеличивающееся изображение</TITLE>
    <script language="JavaScript">
    <!-- //
      function  grpict ()
       { var d= document
         var w= d.mypict.width
         if (w < 300)
         { d.mypict.width=w + 10;
           d.mypict.src="mgr.jpg"
           setTimeout("grpict()", 500)
         }
       }
    //-->
    </script>
  </HEAD>
```

```
  <BODY>
    <P>При наведении курсора мыши над рисунком
       изображение начинает приближаться к зрителю.</P>
    <IMG src="mgr.jpg" name=mypict width=100 onMouseOver="grpict()">
  </BODY>
</HTML>
```

Рис. 2.10. Эффект приближения изображения

Вертикальное графическое меню со стрелкой

Необходимо написать сценарий, реализующий вертикальное графическое меню. При наведении курсора мыши на пункт меню слева от выделенного пункта появляется стрелка, как изображено на рис. 2.11.

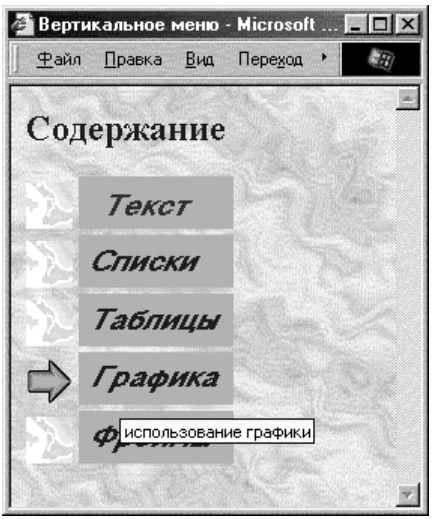


Рис. 2.11. Вертикальное графическое меню со стрелкой

В файле с именем pch1.gif хранится изображение, соответствующее первому пункту меню, в файле с именем pch2.gif — второму пункту меню и т. д. При выборе некоторого пункта меню слева от изображения, соответствующего выбранному пункту, появляется стрелка. Если курсор мыши выходит из области изображения пункта меню, то стрелка исчезает. Изображение стрелки хранится в файле с именем but1.gif, а в файле but0.gif размещается изображение, совпадающее с цветом фона, и использующееся тогда, когда пункт меню не выбран. В документе располагается десять изображений по два на отдельной строке. Пары изображений, которые нельзя размещать на разных строках, заключаются в теги `<nobr>` и `</nobr>`. Первая пара изображений соответствует первому пункту меню и представлена в документе следующим образом:

```
<nobr> <IMG src="but0.gif" width="31" height="31" name="p1">
  <A href="tch1.htm" onmouseover="IMG(p1,true)"
                     onmouseout="IMG(p1,false)">
  <IMG src="pch1.gif" alt="форматирование текста" border="0"
       width="103" height="35"></A>
</nobr> <br>
```

При попадании курсора мыши в область изображения возникает событие `MouseOver`, параметр обработки события `onMouseOver` получает значение `IMG(p1,true)`.

Назначение функции `IMG()` помещать или убирать стрелку слева от изображения, соответствующего выбранному пункту меню. Данная функция описывается следующим образом.

```
function IMG (pict, action)
  { if (action)
     {pict.src="but1.gif" }
    else
     { pict.src="but0.gif" }
  }
```

Какой пункт меню выбран, задается первым параметром, второй параметр определяет, помещать или убирать стрелку с помощью загрузки изображения из файла but0.gif или but1.gif.

Документ со сценарием, реализующим графическое меню, представлен в листинге 2.11.

Листинг 2.11. Вертикальное меню со стрелкой

```
<HTML>
  <HEAD>
    <TITLE>Вертикальное меню со стрелкой</TITLE>
    <script language="JavaScript">
      function IMG  (pict, action)
       {
         if (action)
         { pict.src="but1.gif" }
         else
         { pict.src="but0.gif" }
       }
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <H2><FONT color="#0000FF">Содержание</FONT></H2>
    <nobr> <IMG src="but0.gif" width="31" height="31" name="p1">
      <A href="tch1.htm" onmouseover="IMG(p1,true)"
                         onmouseout="IMG(p1,false)">
      <IMG src="pch1.gif" alt="форматирование текста" border="0"
           width="103" height="35"></A>
    </nobr> <br>
    <nobr><IMG src="but0.gif" width="31" height="31" name="p2">
```

```
      <A href="tch2.htm" onmouseover="IMG(p2, true)"
                         onmouseout="IMG(p2, false)">
      <IMG src="pch2.gif" alt="создание списков" border="0" width="103"
           height="35"></A>
    </nobr> <br>
    <nobr><IMG src="but0.gif" width="31" height="31" name="p3">
      <A href="tch3.htm" onmouseover="IMG(p3, true)"
                         onmouseout="IMG(p3, false)">
      <IMG src="pch3.gif" alt="построение таблиц" border="0" width="103"
           height="35"></A>
    </nobr> <br>
    <nobr><IMG src="but0.gif" width="31" height="31" name="p4">
      <A href="tch4.htm" onmouseover="IMG(p4,true)"
                         onmouseout="IMG(p4,false)">
      <IMG src="pch4.gif" alt="использование графики" border="0"
           width="103" height="35"></A>
    </nobr> <br>
    <nobr><IMG src="but0.gif" width="31" height="31" name="p5">
      <A href="tch5.htm" onmouseover="IMG(p5,true)"
                         onmouseout="IMG(p5,false)">
      <IMG src="pch5.gif" alt="создание фреймовой структуры"
           border="0" width="103" height="35"></A>
    </nobr> <br>
  </BODY>
</HTML>
```

Горизонтальное графическое меню со стрелкой

Напишем сценарий, реализующий горизонтальное графическое меню. При наведении курсора мыши на пункт меню сверху от выделенного пункта появляется стрелка, как изображено на рис. 2.12.

При создании горизонтального графического меню используем таблицу, состоящую из двух строк, по пять ячеек в каждой. Пункты меню в виде графических изображений хранятся в ячейках второй строки. Если выбран некоторый пункт меню, то над ним появляется стрелка. Назначение функции `IMG()` такое же, как и в предыдущем примере. Обратите внимание на вызов указанной функции: вместо логического значения можно использовать 0 или 1.

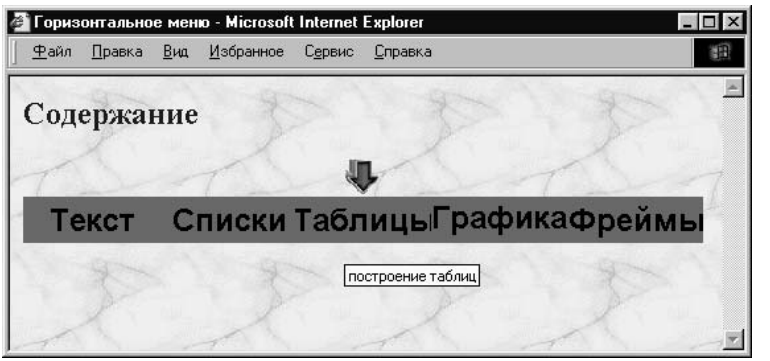


Рис. 2.12. Горизонтальное графическое меню со стрелкой

HTML-код представлен в листинге 2.12.

Листинг 2.12. Горизонтальное меню со стрелкой

```
<HTML>
  <HEAD>
    <TITLE>Горизонтальное меню</TITLE>
    <script language="JavaScript">
      function IMG(pict, action)
       { if (action)
         {pict.src="rbut1.gif"}
         else
         {pict.src="rbut0.gif"}
       }
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <H2><FONT color="#0000FF">Содержание</FONT></H2>
    <TABLE cellspacing=0 cellpadding=0>
      <TR>
        <TD align=center><IMG src="rbut0.gif"
            width="31" height="31" name="p1"></TD>
        <TD align=center><IMG src="rbut0.gif"
            width="31" height="31" name="p2"></TD>
        <TD align=center><IMG src="rbut0.gif"
            width="31" height="31" name="p3"></TD>
        <TD align=center><IMG src="rbut0.gif"
            width="31" height="31" name="p4"></TD>
```

```
        <TD align=center><IMG src="rbut0.gif"
            width="31" height="31" name="p5"></TD>
      </TR>
      <TR>
        <TD><A href="tch1.htm" target="Main" onmouseover="IMG(p1, 1)"
                                             onmouseout="IMG(p1, 0)">
            <IMG src="pch1.gif" alt="форматирование текста"
                 border="0" width="103" height="35"></A></TD>
        <TD><A href="tch2.htm" target="Main" onmouseover="IMG(p2, 1)"
                                             onmouseout="IMG(p2, 0)">
            <IMG src="pch2.gif" alt="создание списков"
                 border="0" width="103" height="35"></A></TD>
        <TD><A href="tch3.htm" target="Main" onmouseover="IMG(p3, 1)"
                                             onmouseout="IMG(p3, 0)">
            <IMG src="pch3.gif" alt="построение таблиц"
                 border="0" width="103" height="35"></A></TD>
        <TD><A href="tch4.htm" target="Main" onmouseover="IMG(p4, 1)"
                                             onmouseout="IMG(p4, 0)">
            <IMG src="pch4.gif" alt="использование графики"
                 border="0" width="103" height="35"></A></TD>
        <TD><A href="tch5.htm" target="Main" onmouseover="IMG(p5, 1)"
                                             onmouseout="IMG(p5, 0)">
            <IMG src="pch5.gif" alt="создание фреймовой структуры"
                 border="0" width="103" height="35"></A></TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

Оператор *switch* и его свойства

В тех случаях, когда при решении задачи требуется выбрать один вариант действия из нескольких возможных, удобно воспользоваться оператором `switch`.

Синтаксис оператора `switch` следующий:

```
switch (B)
 { case L1: S1;
   case L2: S2;
    ... ... ...
```

```
   case Ln: Sn;
   default: S
  }
```

где `B` — выражение; `L1`, `L2`, ..., `Ln` — литералы; `S1`, `S2`, ..., `Sn`; `S` — операторы.

Выполнение переключателя происходит так: вычисляется значение выражения `B`. Если значение `B` равно `L1`, то выполняются операторы `S1`, а затем все остальные операторы либо до первого оператора `break`, либо до конца оператора `switch`. Если значение `B` равно `L2`, то выполняются операторы `S2`, а затем все остальные операторы либо до первого оператора `break`, либо до конца оператора `switch` и т. д. Если же значение `B` не равно ни одному из значений `L1`, `L2`, ..., `Ln`, то выполняются операторы `S`. Часть `default: S` может отсутствовать, тогда переключатель имеет вид

```
switch (B)
 { case L1: S1;
   case L2: S2;
    ... ... ...
   case Ln: Sn;
  }
```

В этом случае, если значение выражения `B` не равно ни одному из значений `L1`, `L2`, ..., `Ln`, то оператор `switch` завершает свою работу, что эквивалентно пустому оператору. На самом деле переключатель удобно записывать в виде:

```
switch (B)
 { case L1: S1; break;
   case L2: S2; break;
    ... ... ...
    case Ln: Sn; break;
    default: S
  }
```

В этом случае оператор `break` обеспечивает завершение работы переключателя после выполнения очередного варианта.

День недели

Напишем программу, которая по номеру дня определяет его название.

Задачу можно решить с использованием условных операторов, в этом случае программа будет ненаглядной. При решении такого рода задач удобно использовать переключатель. Переменной `s` в зависимости от номера дня недели будет присвоено его название. В листинге 2.13 приведен требуемый сценарий.

Листинг 2.13. Определение по номеру дня его названия

```
<HTML>
  <HEAD>
    <TITLE>Определение по номеру дня его названия</TITLE>
    <script language="JavaScript">
    <!-- //
      function  numday (obj)
       { var m = Number(obj.num1.value);
         d= document
         var s
         switch (m)
         { case 1: s="понедельник"; break;
           case 2: s="вторник"; break;
           case 3: s="среда"; break;
           case 4: s="четверг"; break;
           case 5: s="пятница"; break;
           case 6: s="суббота"; break;
           case 7: s="воскресенье"; break;
           default: s="ошибка в номере дня"
         }
        obj.res.value=s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Определение названия дня по его номеру</P>
    <FORM name="form1">
      Введите номер дня: <input type="text" size=7 name="num1"><hr>>
      Название дня:
      <input type="button" value=Определить onClick=" numday(form1)">
      <input type="text"  size=20 name="res"><hr>>
      <input type="reset">
    </FORM>
  </BODY>
</HTML>
```

Номер квартала

Необходимо написать программу, которая по номеру месяца определяет, в какой квартал он попадает.

Напомним, что в первый квартал попадают январь, февраль, март, во второй — апрель, май, июнь и т. д. Обратите внимание на формулу, которая обеспечивает вычисление номера квартала:

```
m=(n-1-(n-1)%3)/3+1
```

HTML-код приведен в листинге 2.14.

Листинг 2.14. Определение по номеру месяца номера квартала

```
<HTML>
  <HEAD>
    <TITLE>Определение по номеру месяца номера квартала</TITLE>
    <script language="JavaScript">
    <!-- //
      function  numkv (obj)
       { var n = Number(obj.num1.value);
         d = document
         var m=(n-1-(n-1)%3)/3+1;
         obj.res.value=m
         switch (m)
          { case 1: d.write("первый квартал"); break;
            case 2: d.write("второй квартал"); break;
            case 3: d.write("третий квартал"); break;
            case 4: d.write("четвертый квартал"); break;
            default: d.write("ошибка")
          }
       }
      function  test ()
       { for (var n=1; n<= 12; ++n)
           document.write (n, "  ",(n-1-(n-1)%3)/3+1,"<br>" )
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление номера квартала по номеру месяца</P>
    <FORM name="form1">
```

```
      Введите номер месяца: <input type="text" size=7 name="num1"><hr>>
      Месяц относится к кварталу номер
      <input type="button" value=Определить onClick="numkv (form1)">
      <input type="text"  size=7 name="res"><hr>>
      <input type="reset">
    </FORM>
  </BODY>
</HTML>
```

Определение номера дня по его названию

Напишем программу, которая по названию дня определяет его номер.

Обратите внимание, что при описании синтаксиса переключателя в качестве метки варианта `L` выступает литерал. В языке JavaScript строка является литералом. В других языках строка, как правило, относится к сложным структурам данных. В переключателе можно использовать в качестве метки варианта строковое значение, что облегчает решение задачи. Приведем только сценарий для данной задачи (листинг 2.15).

Листинг 2.15. Определение номера по его названию

```
<HTML>
  <HEAD>
    <TITLE>Определение по названию номера дня</TITLE>
    <script language="JavaScript">
    <!-- //
      function daynum (obj)
       { var m = obj.num1.value;
         var s
         switch (m)
          { case 'понедельник': s=1; break;
            case 'вторник': s=2; break;
            case 'среда':   s=3; break;
            case 'четверг': s=4; break;
            case 'пятница': s=5; break;
            case 'суббота': s=6; break;
            case 'воскресенье': s=0; break;
            default: s=' ошибка в названии дня'
          }
         obj.res.value=s
       }
```

```
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Определение  по  названию дня его номера</P>
    <FORM name="form1">
      Введите день: <input type="text" size=20 name="num1"><hr>>
      Номер дня:
      <input type="button" value=Определить onClick="daynum (form1)">
      <input type="text"  size=25 name="res"><hr>>
      <input type="reset" value="Обновить">
    </FORM>
  </BODY>
</HTML>
```

Траектория движения точки

Точка движется вдоль ломаной в указанном направлении. Требуется написать сценарий, который определяет координаты точки в конце шага.

Вид документа при выполнении теста приведен на рис. 2.13.

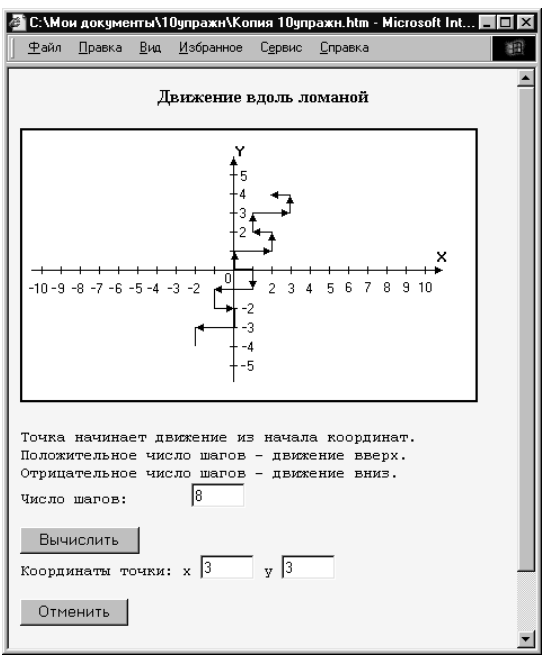


Рис. 2.13. Пример движения точки вдоль ломаной

При решении задачи удобно провести прямые, соединяющие точки ломаной. Так как точка движется только по целочисленным координатам, то в зависимости от того, какому участку принадлежит точка, можно определить ее координату в конце шага. В функции используется переключатель для определения соотношений между координатами *x* и *y* в конце шага.

HTML-код приведен в листинге 2.16.

Листинг 2.16. Движение точки вдоль ломаной на один шаг

```
<HTML>
  <TITLE>Движение вдоль ломаной</TITLE>
  <HEAD>
    <script>
    <!--//
      function movepoint(obj)
       { var a=Number(obj.num.value)
         var b=a%5
         var c=a-b
         var d=c/5
         var e=d
         var f=2*d
         if (a>0)
          { switch (b)
           { case 1: f+=f; break;
             case 2: f+=f; e+=1; break;
             case 3: f+=1; e+=2; break;
             case 4: f+=2; e+=2; break;
           }
          }
         else
          { switch (b)
           { case -1: e+=1; break;
             case -2: f-=1; e+=1; break;
             case -3: f-=1; break;
             case -4: f-=1; e-=1; break;
           }
          }
         obj.x.value=e
```

```
          obj.y.value=f
        }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=F8F8FF>
    <H4 align=center>Движение вдоль ломаной</H4>
    <FORM name="form1">
      <IMG src="Graf_2.jpg" name="c1" border=2>
      <PRE>
Точка начинает движение из начала координат.
Положительное число шагов — движение вверх.
Отрицательное число шагов — движение вниз.
Число шагов:       <input name="num" type="text" size=5><br>
<input type="button" value="Вычислить" name="b1" On-
Click="movepoint(form1)">
Координаты точки: x <input name="x" type="text" size=5> y <input name="y"
type="text"size=5>
<br>
<input type="reset" value="Отменить">
      </PRE>
    </FORM>
  </BODY>
</HTML>
```

Перестановка изображений

В документе расположены три изображения. Два текстовых поля дают возможность пользователю ввести номера изображений, которые требуется поменять местами. Напишите сценарий, осуществляющий такой обмен.

Считаем, что изображения пронумерованы слева направо, начиная с нуля. Если значения номеров рисунков введены неправильно, то требуется выдать сообщение об ошибке. Документ после обмена первого и второго изображений приведен на рис. 2.14.

В функции `ch3pict()` исследуются введенные пользователем значения. Если значения менее 1 или более 3, то выдается сообщение о недопустимых исходных данных. Далее исследуются варианты обмена в следующх парах: 1 и 2, 1 и 3, 2 и 3, 2 и 1, 3 и 1. Все эти случаи рассмотрены и реализованы с помощью условных операторов. Приведем HTML-код документа в листинге 2.17.

Рис. 2.14. Перестановка изображений

Листинг 2.17. Обмен трех изображений

```
<HTML>
  <HEAD>
    <TITLE>Обмен трех изображений</TITLE>
    <script language="JavaScript">
    <!-- //
      function ch3pict ()
       { var d=document
         var a= d.form1.num1.value
         var b= d.form1.num2.value
         var l
         if (a<1||a>3||b<1||b>3)
           alert ("Неверно заданы параметры")
         else
         { if (a==1)
           { l=d.pm1.src
             if (b==2)
             { d.pm1.src=d.pm2.src;  d.pm2.src=l }
             else
               if (b==3)
                 { d.pm1.src=d.pm3.src; d.pm3.src=l }
           }
           else
             if (a==2)
```

```
              { l=d.pm2.src
                if (b==1)
                 { d.pm2.src=d.pm1.src; d.pm1.src=l }
                else
                  if (b==3)
                   { d.pm2.src=d.pm3.src;  d.pm3.src=l }
              }
            else
              if (a==3)
                { l=d.pm3.src
                  if (b==1)
                   { d.pm3.src=d.pm1.src; d.pm1.src=l }
                   else
                    if (b==2)
                     { d.pm3.src=d.pm2.src;  d.pm2.src=l }
                }
          }
        }
    //-->
    </script>
  </HEAD>
  <BODY  bgcolor=F8F8FF>
    <CENTER>
    <H4>Обмен трех изображений</H4>
    <IMG src="m1.gif"  name=pm1 width=100>
    <IMG src="m2.gif"  name=pm2 width=100>
    <IMG src="m3.gif"  name=pm3 width=100>
    <FORM name="form1">
      <input  type="text" name="num1" size=3> и
      <input  type="text" name="num2" size=3>
      <input type="button" value="Обменять" onClick="ch3pict()">
    </FORM>
    </CENTER>
  </BODY>
</HTML>
```

Функцию `ch3pict()` можно упростить, если объединить варианты "1 и 2" и "2 и 1"; "1 и 3" и "3 и 1", "2 и 3" и "3 и 2". Для этого надо найти максимальное и минимальное из введенных пользователем значений. Если эти значе-

ния не совпадают, то исследовать надо следующие варианты: "1 и 2", "1 и 3" и "2 и 3". Приведем пример функции `ch3pict()`.

```
function ch3pict()
  { var d=document
    var a= d.form1.num1.value
    var b= d.form1.num2.value
    var a1=Math.min(a,b)
    var b1=Math.max(a,b)
    var l
    if (a1<1||b1>3)
       alert ("Неверно заданы параметры")
    else
     {if (a1 != b1)
        {if (a1==1)
           {l=d.pm1.src
              if (b1==2)
                {d.pm1.src=d.pm2.src;  d.pm2.src=l}
              else
                if (b1==3)
                  {d.pm1.src=d.pm3.src; d.pm3.src=l}
           }
         else
           if (a1==2)
             {l=d.pm2.src; d.pm2.src=d.pm3.src; d.pm3.src=l}
         }
     }
  }
```

В третьем варианте описания функции `ch3pict()` используется переключатель. Он позволяет текст функции сделать более наглядным.

```
function ch3pict()
  { var d=document
    var a= d.form1.num1.value
    var b= d.form1.num2.value
    var a1=Math.min(a,b)
    var b1=Math.max(a,b)
    var l
    if (a1<1||b1>3)
       alert ("Неверно заданы параметры")
    else
```

```
    {if (a1 != b1)
       switch (a1)
        {case 1: l=d.pm1.src;
                 switch (b1)
                  {case 2: d.pm1.src=d.pm2.src;  d.pm2.src=l; break;
                   case 3:  d.pm1.src=d.pm3.src; d.pm3.src=l;  break
                   }; break;
         case 2: l=d.pm2.src;
                 d.pm2.src=d.pm3.src; d.pm3.src=l; break
        }
    }
}
```

Принятие решения о принадлежности точки некоторой оболасти

Напишем сценарий, в результате работы которого определяется, принадлежит ли точка с координатами (x, y) заштрихованной области. Пользователь вводит значения, определяющие область, и координаты точки на плоскости, затем нажимает на кнопку **Вычислить**. Результат работы сценария отображается в текстовом поле (рис. 2.15).

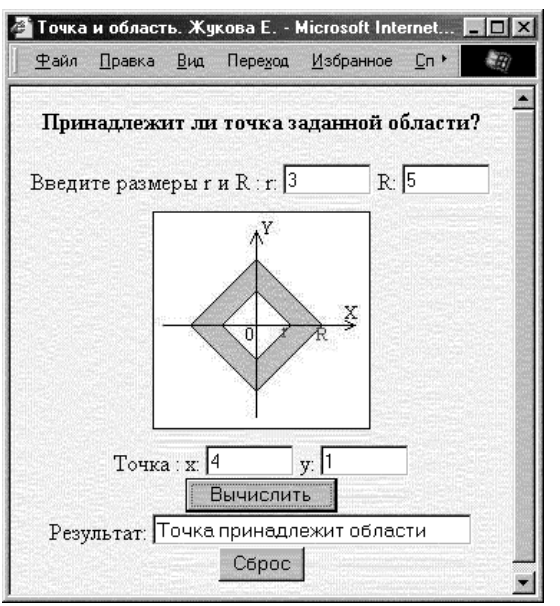


Рис. 2.15. Точка и область

Точка с координатами (x, y) лежит внутри ромба, четыре вершины которого задаются координатами (r, 0), (0, $-r$), ($-r$, 0), (0, r), в случае, если выполняется следующее неравенство $|x| + |y| \le r$. При решении задачи нас интересует область, которая расположена внутри ромба с координатами R и вне ромба с координатами r. Для определения абсолютного значения некоторой величины следует обратиться к методу `abs` стандартного объекта `Math`. Функция `pointcare(obj)` выполняет требуемую проверку.

Полностью HTML-код документа приведен в листинге 2.18.

Листинг 2.18. Точка и область

```
<HTML>
  <HEAD>
    <TITLE>Точка и область. Жукова Е.</TITLE>
    <script>
    <!-- //
      function pointcare(obj)
       { var x = Number(obj.num1a.value)
         var y = Number(obj.num2a.value)
         var r = Number(obj.num3a.value)
         var R = Number(obj.num4a.value)
         var e=Math.abs(x)+Math.abs(y)
         if ((e>=r)&&(e<=R))
          {obj.res.value = "Точка принадлежит области"}
         else
          {obj.res.value = "Точка не принадлежит области"}}
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=F8F8FF>
    <CENTER>
    <H4>Принадлежит ли точка заданной области?</H4>
    <FORM name = "form1">
      Введите размеры r и R:
      r: <input type = "text" name = "num3a" size = 7>
      R: <input type = "text" name = "num4a" size = 7>
      <br><IMG src = "1.jpg" border=1 hspace=10 vspace=10><br>
      Точка:
      x: <input type = "text" name = "num1a" size = 7>
```

```
      y: <input type = "text" name = "num2a" size = 7><br>
      <input type = "button" value = "Вычислить"
             onclick =   "pointcare(form1)"><br>
      Результат:
      <input type = "text" name = "res" size = 30><br>
      <input type = reset><br>
    </FORM>
    </CENTER>
  </BODY>
</HTML>
```

Упражнения

1. Вводится последовательность из пяти чисел. Напишите сценарий, в котором определяется число максимальных элементов.
2. Напишите программу, которая определяет, можно ли построить треугольник с заданными длинами сторон.
3. Точка на плоскости задается своими координатами. Определите, какой из четвертей прямоугольной системы координат принадлежит заданная точка.
4. Участникам тестирования было предложено шесть задач. За решение каждой из задач ставились баллы: 0, 1/3, 2/3 или 1. Всех участников, проходивших тестирование, распределили по четырем категориям в зависимости от результатов. В первую категорию включили участников, все решения которых оценены максимальным баллом 1. Во вторую категорию вошли участники, все задачи которых оценивались не ниже, чем 2/3, но обязательно была хоть одна задача, решение которой оценено на 1. В третью категорию попали участники, у которых все задачи были оценены на 2/3 балла. Остальных участников отнесли к четвертой категории.
   - Создайте анкету конкретного участника тестирования. В анкете должна быть указана фамилия, номер школы, оценки за решения задач. При обработке анкеты для участника требуется определить сумму набранных баллов и категорию, в которую зачислен участник.
   - Подготовьте четыре рисунка, на каждом из которых указан номер категории. Напишите сценарий обработки анкеты, при выполнении которого в документе появляется рисунок с номером категории тестируемого.
5. По результатам сдачи экзаменов в сессию решено назначить стипендию по следующим правилам. Учащимся, сдавшим все шесть экзаменов на оценку 5, назначается стипендия в размере 200 у. е. Тем учащимся, кото-

рые получили оценку не ниже 4 и три из экзаменов сданы на 5, назначается стипендия в 100 у. е. Студентам, все экзамены которых сданы на оценку 4, назначается стипендия в размере 50 у. е. Все остальные студенты стипендию не получают.

- Создайте анкету учащегося, в которой указана фамилия, номер группы и результаты оценок за шесть экзаменов, сданных во время сессии. При обработке анкеты требуется определить средний балл за экзамен и размер назначенной стипендии.
- Подготовьте четыре рисунка, на каждом из которых указан размер стипендии. Напишите сценарий обработки анкеты, при выполнении которого в документе появляется рисунок с размером назначенной стипендии.

6. Напишите программу, которая по заданному значению x вычисляет и выводит значение y. График зависимостей значений y от значений x приведен на рис. 2.16.

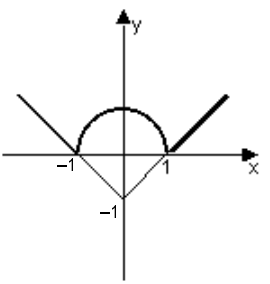


Рис. 2.16. График функции

7. Участникам Олимпиады было предложено шесть задач. За решение каждой из задач ставились баллы: 0, 10, 20 или 30. Первое место присуждалось участникам, все задачи которых были оценены максимальным числом баллов. Второе место заняли участники, решения всех задач которых были оценены не ниже, чем на 20 баллов, причем, по крайней мере, одна задача имела высший балл. Третье место заняли участники, решения всех задач которых оценивались 20 баллами. Призовые места остальным участникам не присуждались.

- Создайте анкету участника Олимпиады. При обработке анкеты требуется определить сумму набранных за решения задач баллов и место, на которое участник Олимпиады претендует.
- Подготовьте три рисунка, на каждом из которых указано место (1, 2 или 3). Напишите сценарий обработки анкеты, при выполнении которого в документе появляется рисунок с номером присужденного места.

8. Точка движется по ломаной, указанной на рис. 2.17, передвигаясь за один шаг по целым значениям. Напишите программу, которая определяет координаты точки в конце шага, если известны координаты точки в начале шага. Значения координат — целые.

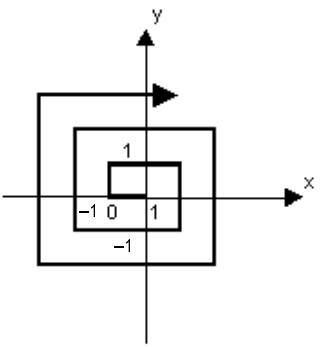


Рис. 2.17. Изображение ломаной

9. Участники конкурса представляют по семь работ. Каждая из работ оценивается 3, 5, 7 или 10 баллами. За участие в конкурсе претендентам назначалась премия в зависимости от результатов оценки работ. Участникам, все работы которых получили высшую оценку, назначается премия в размере 500 у. е. Участникам, все работы которых получили балл не менее 7, но хотя бы одна работа была оценена высшим баллом, назначается премия в размере 300 у. е. Участникам, все работы которых были оценены на 7, назначается премия в размере 200 у. е. Остальным участникам конкурса была назначена премия в 50 у. е.
   - Создайте анкету участника конкурса, в которой указывается фамилия участника и оценки за представленные работы. При обработке анкеты требуется определить сумму баллов за работы и размер назначенной премии.
   - Подготовьте четыре рисунка, на каждом из которых приведен размер назначенной премии (500, 300, 200 или 50). Напишите сценарий обработки анкеты, при выполнении которого в документе появляется рисунок с размером назначенной премии.
10. Заданы три целых значения. Напишите сценарий, при выполнении которого определяется и выводится в документ информация о том, можно ли построить треугольник с длинами сторон, равными заданным значениям. Если треугольник построить можно, то требуется выяснить его вид: прямоугольный, остроугольный, тупоугольный. При определении вида треугольника в документе должно появиться соответствующее изображение.

11. Напишите программу, которая определяет, находится ли точка с заданными координатами внутри заштрихованной области. Область представлена на рис. 2.18.

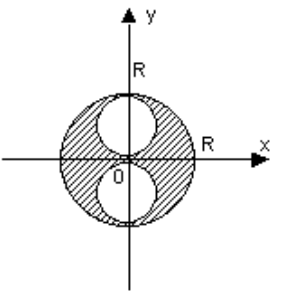


Рис. 2.18. Точка и заштрихованная область

12. Три точки на плоскости задаются координатами своих вершин. Напишите сценарий, при работе которого определяется, можно ли построить треугольник с вершинами в заданных точках и выводится соответствующая информация. Если треугольник построить можно, то требуется выяснить его вид: равносторонний, равнобедренный, разносторонний. При определении вида треугольника в документе должно появиться изображение треугольника соответствующего вида.
13. Два треугольника на плоскости задаются координатами своих вершин. Напишите сценарий, в результате работы которого определяется, являются ли треугольники подобными. В каждом из вариантов решения должно быть выведено соответствующее графическое изображение.
14. Отрезок на плоскости задается координатами своих концов. Напишите сценарий, который для точки на плоскости, заданной собственными координатами, определяет:
    - лежит ли заданная точка на отрезке;
    - точка не лежит на отрезке, но находится на той же прямой, что и отрезок;
    - точка не лежит на отрезке и не принадлежит прямой, содержащей отрезок.

    Для каждого из вариантов ответа требуется вывести изображение, соответствующее ситуации.
15. Два круга на плоскости задаются координатами центра и радиусом. Напишите сценарий, в результате работы которого определяется положение кругов относительно друг друга. Варианты могут быть следующие:
    - один круг лежит внутри другого;
    - круги не имеют общих точек;

- круги пересекаются;
- круги имеют точку касания.

В каждом из вариантов ответа должно быть выведено соответствующее графическое изображение.

16. Круг на плоскости задается координатами центра и радиусом, квадрат — координатой левой верхней вершины и длиной стороны. Напишите сценарий, определяющий взаимное положение круга и квадрата. Возможны следующие варианты:
    - круг и квадрат не имеют общих точек;
    - круг лежит внутри квадрата;
    - квадрат лежит внутри круга;
    - круг и квадрат пересекаются.

    В каждом из вариантов ответа должно быть выведено соответствующее графическое изображение.
17. Треугольник на плоскости задается координатами своих вершин. Напишите сценарий, в результате работы которого выясняется, лежит ли заданная точка на плоскости внутри треугольника, вне его или на одной из сторон. В каждом из вариантов ответа должно быть выведено соответствующее графическое изображение.
18. Точка может двигаться вдоль ломаной, указанной на рис. 2.19, смещаясь за один шаг на одну клетку. Напишите программу, которая определяет координаты точки в конце шага, если известны координаты в начале шага.
19. Напишите сценарий, определяющий координаты точки, которая движется вдоль ломаной через заданное число шагов. Вид документа, содержащего рисунок ломаной и поля для ввода числа шагов и вывода результата, приведен на рис. 2.19.
20. Создайте документ, оглавление которого представляется вертикальным графическим меню. Над оглавлением помещается графическое изображение, которое изменяется при выборе соответствующего пункта меню, например, на месте изображения появляется краткий комментарий к выбранному пункту.
21. Напишите сценарий, в результате работы которого определяется, принадлежит ли точка заштрихованной области. Созданный документ должен иметь вид как на рис. 2.20.
22. Напишите программу, которая по номеру дня недели определяет, является он рабочим или выходным.
23. Напишите программу, которая по номеру учебной пары определяет время начала и конца занятий.
24. Напишите программу, которая в зависимости от номера месяца в году определяет, к какому времени года относится месяц.

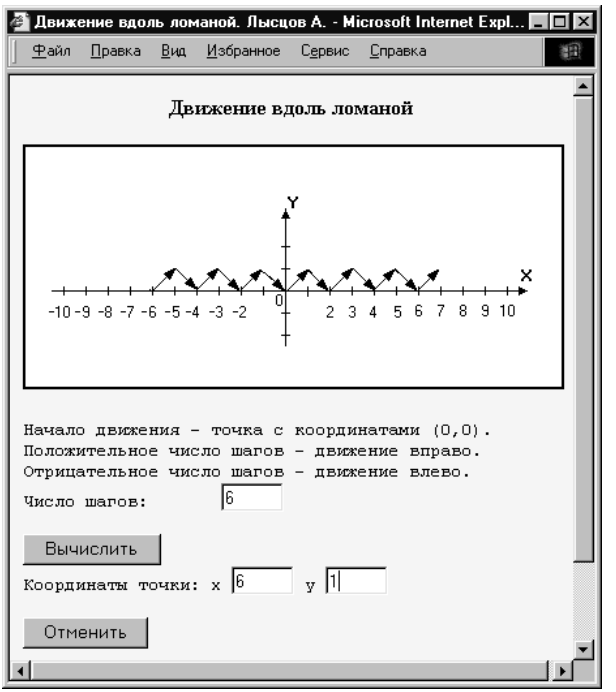


Рис. 2.19. Движение точки вдоль ломаной

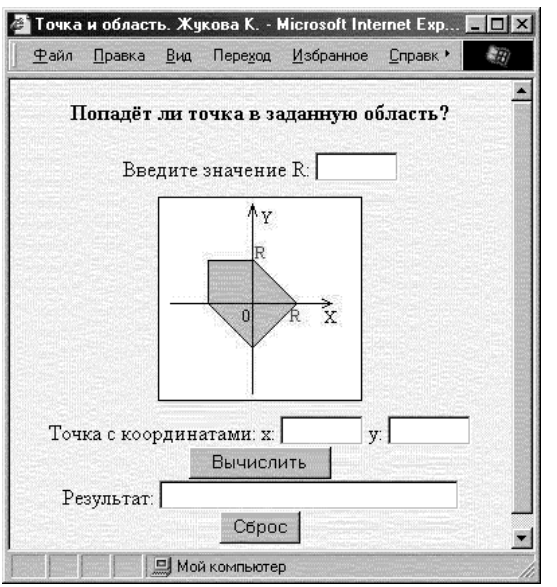


Рис. 2.20. Точка и заданная область

Глава 3

Объекты клиента

Общие сведения

Во время интерпретации HTML-документа браузером создаются объекты JavaScript. При создании сценариев объекты языка JavaScript используются в качестве основных инструментальных средств. Многие свойства объектов определяются значениями параметров тегов языка HTML. Структура документа отражается в иерархической структуре объектов, соответствующих HTML-тегам. Родителем всех объектов является объект `windows`, расположенный на самом верхнем уровне иерархии, он представляет окно браузера и создается при запуске браузера. Для того чтобы открыть новое окно в сценарии JavaScript и отобразить в нем новый документ, применяется метод `open`, для закрытия окна можно воспользоваться методом `close`. Метод `alert()` объекта `windows` отображает диалоговое окно с текстом, переданным методу в качестве параметра. Данный метод используется в случаях проверки правильности вводимых данных с помощью формы. Свойства объекта `windows` относятся ко всему окну, в котором отображается документ.

Подчиненными объектами (или объектами нижнего уровня) являются объекты `document`, `history`, `location`, `frame`. Свойства объекта `history` представляют адреса ранее загружаемых HTML-страниц. Свойства объекта `location` связаны с URL-адресом отображаемого документа, объекта `frame` — со специальным способом представления данных.

Свойства объекта `document` определяются содержимым самого документа: шрифт, цвет фона, формы, изображения и т. д. Объект `document` в зависимости от своего содержимого может иметь объекты, являющиеся для него подчиненными или дочерними. В частности подчиненными для объекта `document` являются объекты `form`, `image`, `link`, `area` и др. Иерархическая структура объектов представлена на рис. 3.1.

Для каждой страницы создается один объект `document`, некоторые его свойства соответствуют параметрам тега `<BODY>`: `bgColor`, `fgColor`, `linkColor`, `alinkColor`, `vlinkColor`. Методы `write` и `writeln` записывают в документ текст, задаваемый параметром.

Если документ содержит изображения, то доступ к объекту, определяющему изображение, можно получить с помощью переменной, указанной в параметре `name` тега `<IMG>`, как мы и поступали ранее. Объект `image` имеет свой-

ство images, которое содержит ссылки на все изображения, расположенные в документе. Ссылки перенумерованы, начиная с нуля. Доступ к первому изображению можно получить с помощью составной конструкции document.images[0], ко второму — document.images[1]. Если на странице пять изображений, то доступ к последнему изображению можно получить, воспользовавшись ссылкой document.images[4].

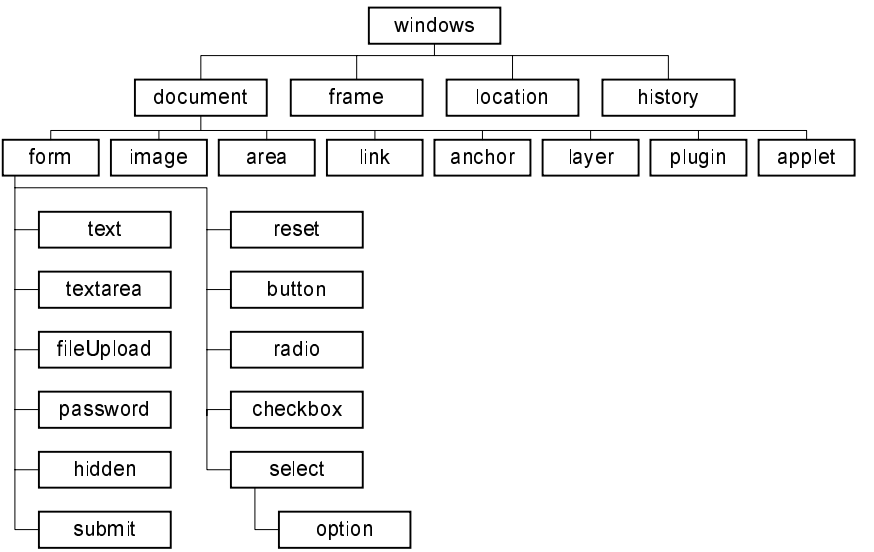


Рис. 3.1. Иерархическая структура объектов

Если на странице расположена форма, то все ее элементы являются подчиненными объектами этой формы. Тегу <FORM> соответствует объект form, являющийся подчиненным по отношению к объекту document. Доступ к форме можно осуществить с помощью значения, заданного в параметре name тега <FORM>. Объект form имеет свойство forms, в котором содержатся ссылки на все формы документа. Ссылки перенумерованы с нуля. Доступ к первой форме документа можно получить так: document.forms[0], ко второй — document.forms[1] и т. д. Вместо индекса в свойстве-массиве forms можно указывать строку, значение которой — имя переменной для формы. Все элементы формы порождают соответствующие объекты, подчиненные объекту родительской формы.

Рассмотрим примеры, в которых используются различные свойства объектов.

Напомним, что для встраивания изображений в HTML-документ служит тег <IMG>, имеющий обязательный параметр src, определяющий URL-адрес файла с изображением. Можно задавать размеры выводимого изображения. Значение параметра width определяет ширину изображения, значение пара-

метра `height` — высоту изображения. Значения параметров ширины и высоты могут не совпадать с истинными размерами изображений, тогда при загрузке изображения автоматически выполняется перемасштабирование.

Изображение можно поместить в рамку. Для этого используется параметр `border`. Значением параметра должно быть число, определяющее толщину рамки в пикселах. По умолчанию рамка вокруг изображения отсутствует, если только изображение не является ссылкой.

Параметр `alt` определяет альтернативный текст. При наведении курсора мыши на изображение появляется комментарий.

Изменение параметров изображения

Необходимо написать сценарий, который для изображения в документе позволяет менять значения параметров ширины и высоты, создавать рамку вокруг изображения и задавать альтернативный текст.

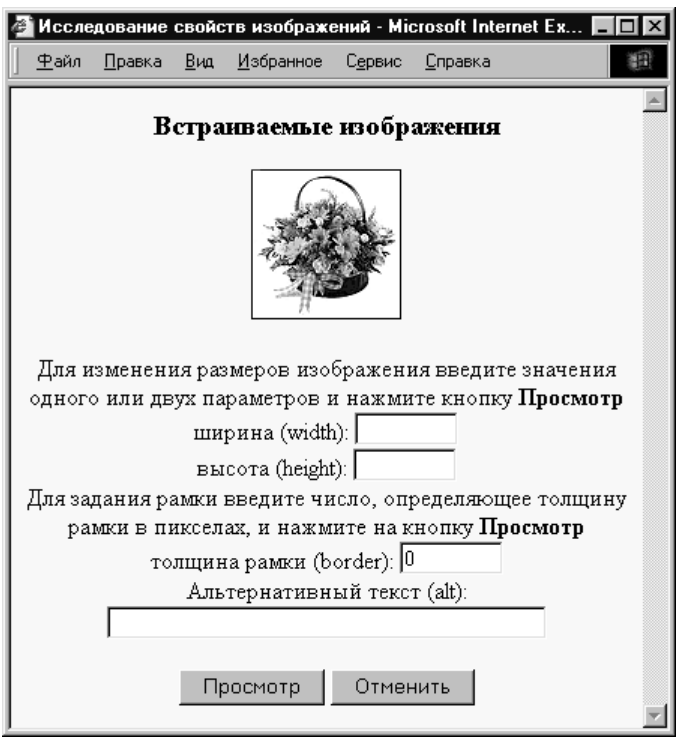


Рис. 3.2. Изменение параметров изображения с помощью сценария

На рис. 3.2 представлены изображение и поля ввода, позволяющие задать значения, чтобы изменить некоторые параметры изображения. Следующие свойства объекта `image`: `border`, `width`, `height`, `alt` соответствуют атрибутам тега `<IMG>`: `border`, `width`, `height`, `alt`. В сценарии следует предусмотреть ситуацию, когда значение какого-либо из параметров не задано. Доступ к объекту `image` осуществляется с помощью значения, указанного в параметре `name` тега `<IMG>`. В рассмотренных ранее примерах изменялись такие свойства изображения, как `width`, чтобы создавать эффект приближения или удаления рисунка, и свойство `src` при организации смены изображений в документе. На рис. 3.3 приведен вид документа после задания параметров графического объекта и выполнения сценария.

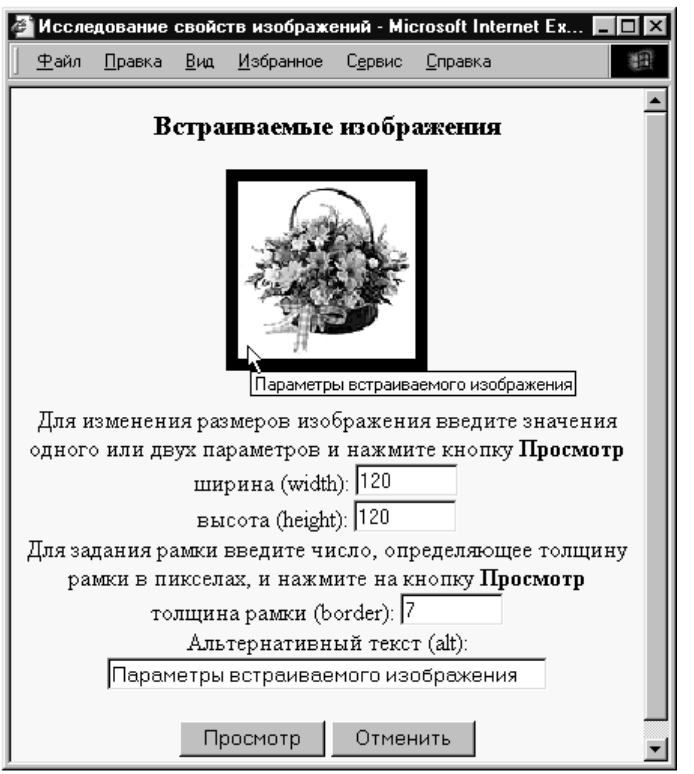


Рис. 3.3. Результат выполнения сценария

Приведенный пример иллюстрирует некоторые параметры тега `<IMG>`. Можно изменять параметры и сразу видеть, как в зависимости от их значений меняется изображение. Сначала в тегах HTML-документа зададим значение параметра `name`, затем в сценарии получим доступ к различным элементам, используя значение параметра.

HTML-код представлен в листинге 3.1, а.

Листинг 3.1, а. Изменение размеров изображения и толщины рамки вокруг него

```
<HTML>
  <HEAD>
    <TITLE> Исследование свойств изображений </TITLE>
    <script>
    <!--
      function chpict(obj)
       { var w=obj.wd.value
         var h= obj.hg.value
         if (w !=0) document.mypict.width=w
         if (h !=0) document.mypict.height=h
         document.mypict.border=obj.br.value
         document.mypict.alt=obj.al.value
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <CENTER>
    <H3>Встраиваемые изображения</H3>
    <IMG src=p1.gif name=mypict>
    <FORM name="form1">
      Для изменения размеров изображения введите значения одного
      или двух параметров и нажмите кнопку <B>Просмотр</B><br>
      ширина (width): <input type="text" name="wd" size=8><br>
      высота (height): <input type="text" name="hg" size=8><br>
      Для задания рамки введите число, определяющее толщину рамки
      в пикселах, и нажмите на кнопку <B>Просмотр</B><br>
      толщина рамки (border): <input type="text" name="br"
                                     size=8 value=0><br>
      Альтернативный текст (alt): <input type="text"
                                          name="al" size=40><P>
      <input type="button" value= "Просмотр"  onclick="chpict(form1)">
      <input type="reset" value="Отменить">
    </FORM>
```

```
    </CENTER>
  </BODY>
</HTML>
```

В описанном документе имеется лишь одно изображение. Доступ к нему можно получить, воспользовавшись свойством `images` объекта `document`, следующим образом: `document.images[0]`. Доступ же к различным свойствам изображения можно получить с помощью свойств самого изображения, например, так:

```
document.images[0].width
document.images[0].height
document.images[0].border
document.images[0].alt
```

Диалог с пользователем обеспечивается с помощью формы. В последнем примере используется одна форма. Доступ к форме можно получить так: `document.forms[0]`. Форма содержит шесть элементов, четыре текстовых поля и две кнопки. Свойство `elements` формы хранит информацию обо всех ее элементах в том порядке, в каком они встречаются в HTML-документе. Получить доступ к объектам формы можно, воспользовавшись свойствами:

```
document.forms[0].elements[0]
document.forms[0].elements[1]
document.forms[0].elements[2]
document.forms[0].elements[3]
document.forms[0].elements[4]
document.forms[0].elements[5]
```

В следующем варианте решения задачи (листинг 3.1, б) доступ ко всем значениям осуществляется с помощью свойств `images` и `forms` объекта `document`.

Листинг 3.1, б. Исследование свойств изображений

```
<HTML>
  <HEAD>
    <TITLE> Исследование свойств изображений </TITLE>
    <script>
    <!--
      function chpict1()
       { var w=document.forms[0].elements[0].value
         var h=document.forms[0].elements[1].value
         if (w !=0)
```

```
          document.images[0].width=w
        if (h !=0)
          document.images[0].height=h
        document.images[0].border= document.forms[0].elements[2].value
        document.images[0].alt= document.forms[0].elements[3].value
      }
  //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <CENTER>
  <H3>Встраиваемые изображения</H3>
  <IMG src=p1.gif>
  <FORM>
    Для изменения размеров изображения введите значения
    одного или двух параметров и нажмите на кнопку <B>Просмотр</B><br>
    ширина (width): <input type="text"  size=8><br>
    высота (height): <input type="text"  size=8><br>
    Для задания рамки введите число, определяющее толщину рамки
    в пикселах, и нажмите на кнопку <B>Просмотр</B><br>
    толщина рамки (border): <input type="text" size=8 value=0><br>
    Альтернативный текст (alt): <input type="text" size=40><P>
    <input type="button" value= "Просмотр" onclick="chpict1()">
    <input type="reset"  value="Отменить">
  </FORM>
  </CENTER>
</BODY>
</HTML>
```

Обратите внимание, что в этом варианте не заданы параметры `name` в HTML-тегах. В них нет необходимости, т. к. доступ к необходимым значениям осуществлялся на основе объектной модели JavaScript.

Перестановка изображений

Напишем сценарий, который реализует обмен рисунков в документе. Пусть в документе расположено четыре изображения, пронумерованных от 1 до 4 (рис. 3.4). В текстовых полях указываются номера рисунков, которые необходимо поменять местами. Требуется, чтобы после нажатия кнопки **Поменять местами** изображения переместились на нужные места.

Рис. 3.4. Обмен рисунков в документе

Сначала проверим, правильно ли заданы номера изображений, если это не так, то выдадим сообщение. Переменная `z` служит для запоминания адреса первого графического изображения. Доступ к изображению с номером `r1` производится с помощью конструкции `document.images[r1-1]`. Для того чтобы на место изображения с номером `r1` поместить изображение с номером `r2`, требуется выполнить оператор присваивания:

```
document.images[r1-1].src=document.images[r2-1].src
```

И, наконец, на место изображения с номером `r2` помещается изображение, которое ранее было на месте с номером `r1`, и адрес которого запомнили в переменной `z`:

```
document.images[r2-1].src=z
```

Приведем полностью документ со сценарием (листинг 3.2).

Листинг 3.2. Перестановка изображений с помощью сценария

```
<HTML>
  <HEAD>
    <TITLE>Перестановка изображений</TITLE>
    <script>
      function chan(obj)
       { var r1=Number(obj.a1.value)
         var r2=Number(obj.a2.value)
```

```
        if ((r1<1)||(r1>4)||(r2<1)||(r2>4))
          alert ("Неверно заданы номера рисунков!")
        else
         { var z=document.images[r1-1].src
           document.images[r1-1].src=document.images[r2-1].src;
           document.images[r2-1].src=z
         }
       }
    </script>
  </HEAD>
  <BODY>
    <CENTER>
    <H4>Галерея рисунков</H4><br>
    <IMG src="p1.gif" width="90"  name="pic1">
    <IMG src="p2.gif" width="90"  name="pic2">
    <IMG src="p3.gif" width="90"  name="pic3">
    <IMG src="p4.gif" width="90"  name="pic4"><br><br>
    <FORM name=form1>
      Рисунки с номерами<br>
      <input type="text" name="a1" size=1>  и
      <input type="text" name="a2" size=1><P>
      <input type="button" value="Поменять местами"
             onClick="chan(form1)">
    </FORM>
    </CENTER>
  </BODY>
</HTML>
```

Простое вертикальное меню

Напишем сценарий, реализующий вертикальное графическое меню. При наведении курсора мыши на пункт меню меняется цветовая палитра, соответствующая выделенному пункту меню (рис. 3.5).

Такая задача уже решалась при обсуждении событий и реакции на них. Рассмотрим другие способы решения этой задачи.

Как и ранее, каждому пункту меню соответствует два изображения: первое изображение, когда пункт меню не выбран, второе — при выбранном пункте меню, цветовая палитра рисунка изменена. Графические изображения, соответствующие ситуации, когда пункты меню не выбраны, хранятся в

файлах с именами pch1.gif, pch2.gif, pch3.gif, pch4.gif, pch5.gif. Соответствующие им графические изображения с измененной палитрой хранятся в файлах с именами wpch1.gif, wpch2.gif, wpch3.gif, wpch4.gif, wpch5.

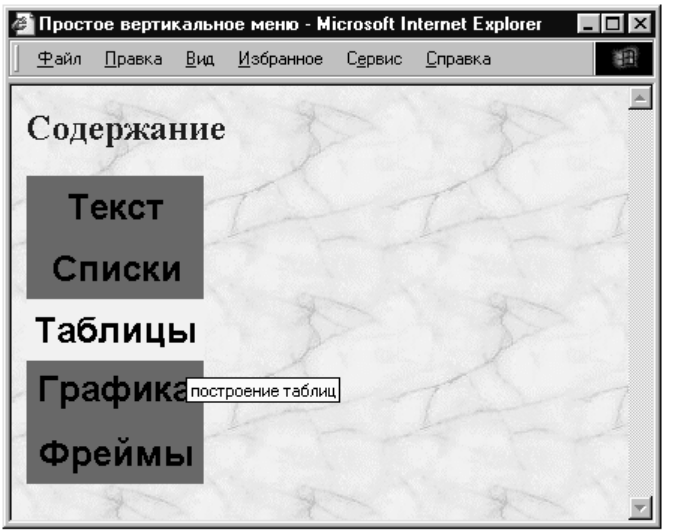


Рис. 3.5. Графическое вертикальное меню

Функция `img` имеет два параметра. Первый параметр задает выбор пункта меню, второй параметр — `n` — определяет номер пункта меню. От этого параметра зависит, какое изображение в документе требуется изменить (`document.images[n-1].src`), и файл с каким именем использовать (`"wpch"+n+".gif"` или `pch"+n+".gif`). Имя файла формируется динамически и представляет собой конкатенацию строк, одна из составляющих которой — значение второго параметра. Если имена файлов не подчинены общему правилу, то в функции потребуется дополнительный анализ, какой файл подгрузить. Это сделать нетрудно, зная место в документе, из которого произошел вызов функции. Документ со сценарием, реализующий вертикальное графическое меню, представлен в листинге 3.3.

Листинг 3.3. Простое вертикальное меню

```
<HTML>
  <HEAD>
    <TITLE>Простое вертикальное меню</TITLE>
    <script language="JavaScript">
      function img(n, action)
       {if (action)
```

```
          {document.images[n-1].src="wpch"+n+".gif"}
        else
          {document.images[n-1].src="pch"+n+".gif"}
       }
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <H2><FONT color="#0000FF">Содержание</FONT></H2>
    <A href="tch1.htm" onmouseover="img(1, 1)"  onmouseout="img(1, 0)">
      <IMG src="pch1.gif" alt="форматирование текста" border="0"
           width="103" height="35"></A><br>
    <A href="tch2.htm" onmouseover="img(2, 1)"  onmouseout="img(2, 0)">
      <IMG src="pch2.gif" alt="создание списков" border="0"
           width="103" height="35"></A><br>
    <A href="tch3.htm" onmouseover="img(3, 1)"  onmouseout="img(3, 0)">
      <IMG src="pch3.gif" alt="построение таблиц" border="0"
           width="103" height="35"></A><br>
    <A href="tch4.htm" onmouseover="img(4, 1)" onmouseout="img(4, 0)">
      <IMG src="pch4.gif" alt="использование графики" border="0"
           width="103" height="35"></A><br>
    <A href="tch5.htm" onmouseover="img(5, 1)" onmouseout="img(5, 0)">
      <IMG src="pch5.gif" alt="создание фреймовой структуры"
           border="0" width="103" height="35"></A> <br>
  </BODY>
</HTML>
```

Простое горизонтальное меню

Напишем сценарий, реализующий горизонтальное графическое меню. При наведении курсора мыши на пункт меню изменяется световая гамма так, как изображено на рис. 3.6.

При реализации горизонтального графического меню функция `img` выполняет те же действия, что и в предыдущем примере. Графические изображения, соответствующие пунктам меню, помещены в таблицу, которая состоит из одной строки и пяти ячеек. В каждую из ячеек помещено изображение для соответствующего пункта. Значения параметров `cellpadding` и `cellspacing` тега `<TABLE>`, определяющих расстояние между ячейками таблицы и расстояние от содержимого ячейки и границы, равны нулю для того, чтобы создавалось впечатление единой графической панели. Сценарий, реализующий графическое меню, приведен в листинге 3.4.

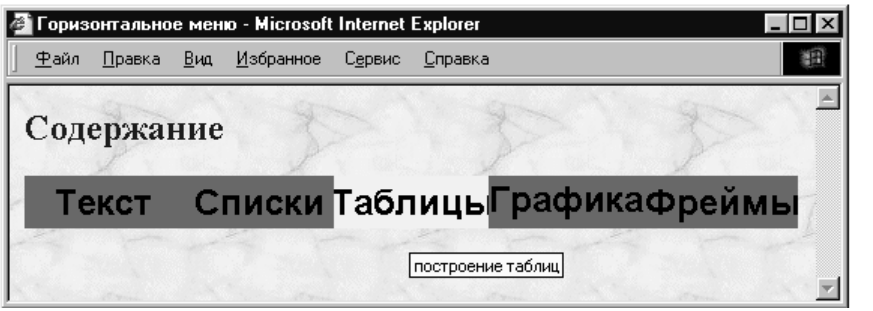


Рис. 3.6. Графическое горизонтальное меню

Листинг 3.4. Простое горизонтальное меню

```
<HTML>
  <HEAD>
    <TITLE>Горизонтальное меню</TITLE>
    <script language="JavaScript">
      function img(n, action)
       {if (action)
         {document.images[n-1].src="wpch"+n+".gif"}
        else
         {document.images[n-1].src="pch"+n+".gif"}
       }
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <H2><FONT color="#0000FF">Содержание</FONT></H2>
    <TABLE border=0 cellpadding=0 cellspacing=0>
      <TR>
        <TD>
          <A href="tch1.htm" onmouseover="img(1, 1)"
                             onmouseout="img(1, 0)">
            <IMG src="pch1.gif" alt="форматирование текста"
                 border="0" width="103" height="35"></A></TD>
        <TD>
          <A href="tch2.htm" onmouseover="img(2, 1)"
                             onmouseout="img(2, 0)">
            <IMG src="pch2.gif" alt="создание списков" border="0"
                 width="103" height="35"></A></TD>
```

```
        <TD>
          <A href="tch3.htm" onmouseover="img(3, 1)"
                             onmouseout="img(3, 0)">
            <IMG src="pch3.gif" alt="построение таблиц"
                 border="0" width="103" height="35"></A></TD>
        <TD>
          <A href="tch4.htm" onmouseover="img(4, 1)"
                             onmouseout="img(4, 0)">
           <IMG src="pch4.gif" alt="использование графики"
                border="0" width="103" height="35"></A></TD>
        <TD>
          <A href="tch5.htm" onmouseover="img(5, 1)"
                             onmouseout="img(5, 0)">
          <IMG src="pch5.gif" alt="создание фреймовой структуры"
               border="0" width="103" height="35"></A></TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```

Анкета "Нагрузка преподавателя"

Необходимо написать сценарий обработки анкеты преподавателя. В анкете заданы поля, в которые требуется ввести количество часов, отведенных на чтение лекций и проведение практических занятий, и число студентов. Если по предмету читаются лекции, то дополнительно планируется нагрузка: 10% времени от лекционных часов отводится на консультации, и для приема экзамена планируется по 30 минут на человека. Если по предмету проводятся практические занятия, то предусмотрена контрольная работа из расчета 15 минут на человека, зачет из расчета 20 минут на человека. Форма анкеты представлена на рис. 3.7.

Аналогичные задачи неоднократно решались, написать сценарий предлагается в качестве упражнения. Предположим, что нагрузку преподавателя требуется представить в виде диаграммы. В анкете преподавателя заполняются первые три поля, а затем после нажатия кнопки **Вычислить** в документе должна появиться диаграмма, построенная по значениям тех полей, которые вычислялись в анкете, а именно: контрольные работы, зачеты, консультация, экзамены. Диаграмма представляется с помощью изображений. Сначала в зависимости от исходных данных считаются значения требуемых величин, затем вычисляется величина каждого из столбцов, и нужные изображения помещаются в документ для формирования столбцов диаграммы. Столбцы диаграммы располагаются в ячейках таблицы, для каждого столбца выделена своя ячейка.

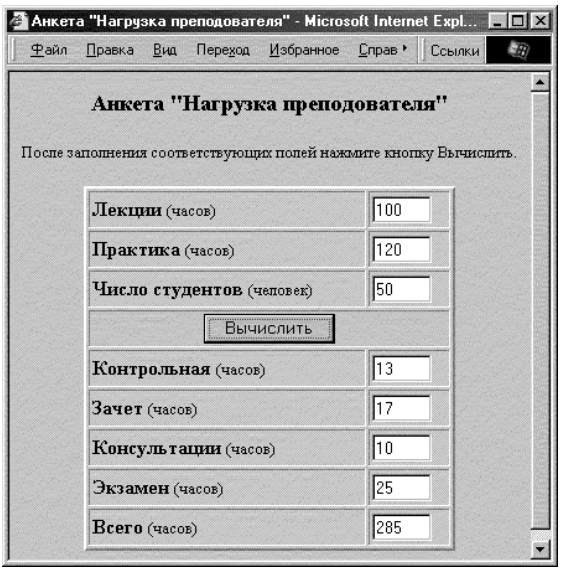


Рис. 3.7. Пример заполнения анкеты

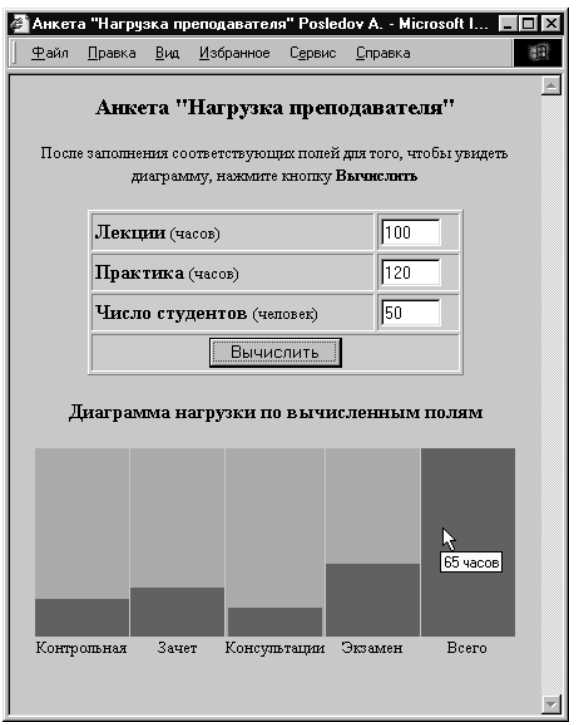


Рис. 3.8. Нагрузка с диаграммой

Вторая строка таблицы содержит подписи к столбцам диаграммы. Если подвести курсор мыши к столбцу, то выводится числовое значение, соответствующее столбцу диаграммы. Документ с введенными полями и построенной диаграммой изображен на рис. 3.8.

Приведем для этого случая HTML-код со сценарием (листинг 3.5).

Листинг 3.5. Нагрузка преподавателя с диаграммой

```
<HTML>
  <HEAD>
    <TITLE>Анкета "Нагрузка преподавателя" Posledov A.</TITLE>
    <script language=JavaScript>
    <!--
      function graph(obj)
       { var k1 = 0;
         var l = obj.lec.value
         var p = obj.pract.value
         var s = obj.stud.value
         var tableHeight = 150;
         var knTime=0;
         var eTime=0;
         var krTime=0;
         var zTime=0;
         var d= document
         if (s == "") alert("Введите количество студентов?")
         if ((l != "")&&(l != "0"))
          { knTime=Math.round( l*10/100); eTime=Math.round(s*30/60) }
         if ((p != "")&&(l != "0"))
          { krTime=Math.round(s*15/60);   zTime=Math.round(s*20/60) }
         sTime = knTime+eTime+krTime+zTime
         k1 = tableHeight/sTime;
         d.images[0].height=Math.round(krTime*k1);
         d.images[0].alt=krTime+" часов";
         d.images[1].height=Math.round(zTime*k1);
         d.images[1].alt=zTime+" часов";
         d.images[2].height=Math.round(knTime*k1);
         d.images[2].alt=knTime+" часов";
         d.images[3].height=Math.round(eTime*k1);
         d.images[3].alt=eTime+" часов";
```

```
        d.images[4].height=Math.round(sTime*k1);
        d.images[4].alt=sTime+" часов";
       }
      function CheckValHours(hours)
       { if (hours < 0)
        alert("Недопустимо значение количества часов меньшее нуля!");
       }
      function CheckValNum(num)
       { if (num <= 0)
        alert("Недопустимо значение меньшее либо равное нулю!");
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=#crcccc>
    <CENTER>
    <H3>Анкета &quot;Нагрузка преподавателя&quot;</h3>
    <SMALL>После заполнения соответствующих полей для того,
         чтобы увидеть диаграмму, нажмите кнопку <B>Вычислить</B></SMALL>
    <FORM name=data>
      <TABLE bgcolor=#cccccc width=300 cellPadding=2
            cellSpacing=2 border=1>
        <TR><TD align=left><b>Лекции</b> <small>(часов)</small></TD>
            <TD align=left><input type=text name=lec  size=5 value=""
                        onChange="CheckValHours(this.value)"></TD>
        </TR>
        <TR><TD align=left><b>Практика</b> <small>(часов)</small></TD>
            <TD align=left><input type=text name=pract size=5  value=""
                onChange="CheckValHours(this.value)"></TD>
        </TR>
        <TR><TD align=left><b>Число студентов</b>
                        <small>(человек)</small></TD>
            <TD align=left><input type=text name=stud size=5 value=""
                onChange="CheckValNum(this.value)"></TD>
        </TR>
        <TR align=center><TD colspan=2  bgcolor=#cccccc>
           <input type=button value="Вычислить"
                  onClick="graph(data)"></TD></TR>
      </TABLE>
```

```
    </FORM>
    <H4>Диаграмма нагрузки по вычисленным полям<h4>
    <TABLE width=212 height=190 cellPadding=0 cellSpacing=1 border=0>
      <TR valign=bottom width=100 align=center>
        <TD bgcolor=#aaaaaa height=150>
          <IMG src=1x1.gif width=75 height=0 border=0></TD>
        <TD bgcolor=#aaaaaa height=150>
          <IMG src=1x1.gif width=75 height=0 border=0></TD>
        <TD bgcolor=#aaaaaa height=150>
          <IMG src=1x1.gif width=75 height=0 border=0></TD>
        <TD bgcolor=#aaaaaa height=150>
          <IMG src=1x1.gif width=75 height=0 border=0></TD>
        <TD bgcolor=#aaaaaa height=150>
          <IMG src=1x1.gif width=75 height=0 border=0></TD>
      </TR>
      <TR valign=top align=center width=100>
        <TD height=40><small>Контрольная</small></TD>
        <TD height=40><small>Зачет</small></TD>
        <TD height=40><small>Консультации</small></TD>
        <TD height=40><small>Экзамен</small></TD>
        <TD height=40><small>Всего</small></TD>
      </TR>
    </TABLE>
    </CENTER>
  </BODY>
</HTML>
```

Диаграмма в анкете преподавателя

Необходимо написать сценарий обработки анкеты преподавателя. В анкете заданы поля и требуется вычислить те значения, которые определены в предыдущем примере. Кроме того, для дополнительной нагрузки необходимо построить диаграмму, которая изображена на рис. 3.9. При наведении курсора мыши на соответствующее поле выводится вычисленное значение.

Поля, соответствующие разным категориям нагрузки, имеют разный цвет. В этом случае в зависимости от полученных значений вычисляется ширина графического изображения для каждого из видов нагрузки. Считается, что ширина всей таблицы соответствует суммарной дополнительной нагрузке. Самостоятельно напишите сценарий и создайте документ.

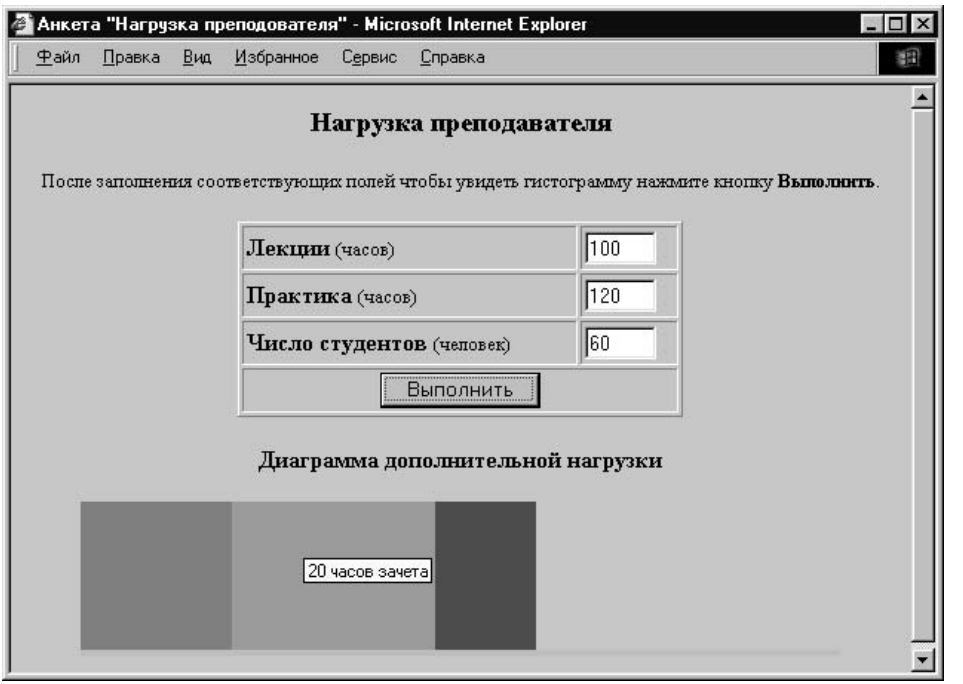


Рис. 3.9. Диаграмма дополнительной нагрузки

Изменение таблицы при разных значениях ее параметров

Напишем сценарий, который позволяет продемонстрировать, как будет меняться таблица и ее ячейки при изменении значений параметров: `border`, `cellspacing`, `cellpadding`.

Напомним, что параметр `border` управляет видом рамки вокруг каждой ячейки, отображает линию сетки таблицы и вокруг всей таблицы. По умолчанию рамки не рисуются. Численное значение параметра определяет толщину рамок в пикселах, рисуемую вокруг всей таблицы. На толщину рамок вокруг каждой ячейки это значение не влияет. Параметр `cellspacing` используется в виде `cellspacing=n`, где `n` — численное значение параметра, определяющее расстояние между рамками двух смежных ячеек, как по горизонтали, так и по вертикали. Значение задается в пикселах и по умолчанию принимается равным двум. При значении `cellspacing=0` рамки смежных ячеек сольются, создадут впечатление единой сетки таблицы. Параметр `cellpadding` применяется в виде `cellpadding=n`, где `n` — численное значение параметра в пикселах, которое может быть опущено. Величина `n` указывает на размер отступа между рамкой ячейки и данными внутри ячейки. По

умолчанию значение равно 1. Все три параметра действуют независимо друг от друга; если какой-либо параметр опущен, то его значение берется по умолчанию.

Пусть таблица состоит из четырех ячеек, в каждой из которых хранится изображение, как на рис. 3.10.

Рис. 3.10. Изображения и таблицы

Для того чтобы воздействие параметров было наглядным, специально выбран разный фон для документа, таблицы и ячеек таблицы. Можно вводить значения для указанных параметров, изображение таблицы в документе будет изменено в соответствии с параметрами. В сценарии требуется изменить значения свойств таблицы.

Используем еще один способ доступа к элементам форм. При задании тегов можно применять параметр `id`. В теге `<TABLE>` задан параметр `id="itab"`. Модификация размера рамки таблицы произойдет при изменении свойства `document.all("itab").border`. Изменение других свойств таблицы про-

изойдет лишь в случае, когда в соответствующих полях расположены числовые данные. Сценарий приведен в листинге 3.6.

Листинг 3.6. Изменение толщины рамки и полей внутри ячейки таблицы

```
<HTML>
  <HEAD>
    <TITLE>Таблицы, толщина рамки, ширина отступов</TITLE>
    <script>
    <!--
      function rset(obj)
       { document.all("itab").border=1
         document.all("itab").cellSpacing=15
         document.all("itab").cellPadding=10
         obj.bor.value=1
         obj.cells.value=15
         obj.cellp.value=10
       }
      function set(obj)
       { document.all("itab").border=Number(obj.bor.value)
         if (obj.cells.value != "")
           document.all("itab").cellSpacing=Number(obj.cells.value)
         if (obj.cellp.value != "")
           document.all("itab").cellPadding=Number(obj.cellp.value)
       }
    //-->
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <H4 align=center>Таблицы, границы, ширина отступов</H4>
    <TABLE id="itab" border=1 cellspacing=15 cellpadding=10
           bgcolor=silver background="fon5.gif" align=center>
      <TR>
        <TD bgcolor=silver><A href="ruins.gif">
          <IMG src="ruins.gif" border="0" width="120"></A></TD>
        <TD bgcolor=silver><A href="sphinx.gif">
          <IMG src="sphinx.gif" border="0" width="120"></A></TD></TR>
      <TR>
        <TD bgcolor=silver><A href="b04.jpg">
          <IMG src="b04.jpg" border="0" width="120"></A></TD>
```

```
        <TD bgcolor=silver><A href="b07.jpg">
          <IMG src="b07.jpg" border="0" width="120"></A></TD></TR>
    </TABLE>
    Выберите значение параметров, которые требуется изменить,
    и нажмите кнопку <B>Выполнить</B>
    <FORM name="form1">
      <PRE>
BORDER       <input type="text" size=8 name=bor value=1> Толщина рамки
CELLSPACING  <input type="text" size=8 name=cells value=15> Ширина проме-
жутка между ячейками
CELLPADDING  <input type="text" size=8 name=cellp value=10> Поля внутри
ячейки
      </PRE>
      <input type="button" value= "Выполнить"  onclick="set(form1)">
      <input type="reset"  value= "Обновить"   onclick="rset(form1)">
    </FORM>
  </BODY>
</HTML>
```

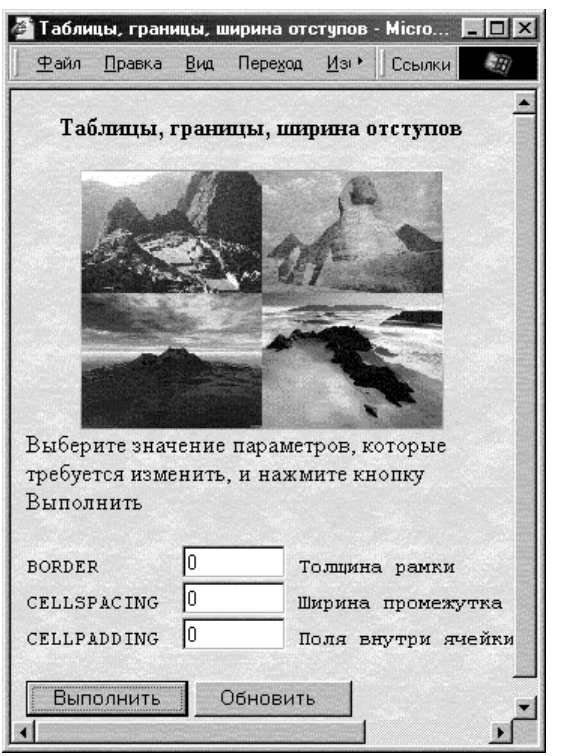


Рис. 3.11. Пример работы сценария создания компактной таблицы

Считается, что смежные ячейки таблицы имеют общую границу, однако, в HTML-таблицах по умолчанию между ячейками оставляется место. Для того чтобы казалось, что смежные ячейки имеют общую границу, надо установить параметр `cellspacing` равным нулю.

Наиболее компактная таблица получится в случае, когда значения рассмотренных трех параметров равны 0. Тогда ячейки окажутся расположенными вплотную друг к другу. Если все ячейки таблицы содержат рисунки одинакового размера и эти рисунки требуется расположить вплотную друг к другу, то следует установить значения параметров равными нулю. Таблица примет вид как на рис. 3.11, если значения всех трех параметров задать равными нулю.

Упражнения

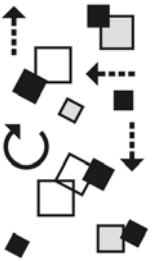
1. В анкете для каждого из пяти сотрудников приводятся данные: фамилия, зарплата, количество детей. Требуется написать сценарий для определения дохода в семье на человека. Кроме того, необходимо определить количество сотрудников, имеющих минимальный доход на человека. Постройте диаграмму, отражающую доход в семье на человека.
2. В анкете для каждого из шести сотрудников приводятся данные: фамилия и год приема на работу. Требуется написать сценарий вычисления стажа работы и определения максимального числа сотрудников с одинаковым стажем. Постройте диаграмму, отражающую стаж работы сотрудников.
3. В анкете для каждого из шести сотрудников приводится информация: фамилия и зарплата. Решено каждому из сотрудников назначить премию по принципу: если его зарплата меньше, чем средняя, то размер премии составляет 50% от зарплаты, в остальных случаях — 30% от зарплаты. Требуется написать сценарий определения суммы, выдаваемой сотруднику на руки (зарплата плюс премия). Кроме того, необходимо определить количество сотрудников, которые получили на руки максимальную сумму.
4. Приводятся данные о закупках пяти наименований товаров: цена за единицу и количество приобретенных экземпляров. Напишите сценарий, определяющий сумму, затраченную на приобретенные товары. Определите, имеются ли товары, на которые потрачена одинаковая сумма, и сколько их. Постройте диаграмму, отражающую затраченные суммы на приобретение разных товаров.
5. В анкете для каждого из десяти студентов приводится информация: фамилия и две оценки за контрольную работу. Студенты разделяются на несколько категорий. Категория "отличники" состоит из студентов, у которых обе контрольные написаны на оценку 5, к категории "хорошо успевающие" относятся студенты, у которых оценка за каждую контрольную — 4 или 5, но студент не отличник. Категорию "успевающие" состав-

ляют студенты, у которых хотя бы одна контрольная написана на 3, наконец "неуспевающие" — те студенты, которые имеют 2 хотя бы за одну контрольную. Требуется написать сценарий определения числа студентов в каждой категории. Постройте диаграмму распределения студентов по категориям.

6. В анкете заполняется информация о шести студентах: фамилия и четыре оценки за сессию. Напишите сценарий определения категории студента и количество человек в каждой категории. Категории определяются следующим образом: сдавшие все экзамены на 5 относятся к категории "отличники", имеющие хотя бы одну 2 — к категории "неуспевающие", все остальные студенты относятся к категории "успевающие".

7. По результатам тестирования заполняется анкета: фамилия и результат выполнения каждого теста (плюс, если успешно, и минус, если тест не выполнен). Если выполнены все тесты, то оценка за работу 5, если выполнены четыре теста, то оценка 4, если выполнены три теста, то оценка 3, в остальных случаях оценка 2. Напишите сценарий вычисления оценки и формирования сводки. В сводке содержатся сведения о количестве студентов, получивших за работу 5, 4, 3, 2.

Глава 4



Переключатели

Общие сведения

Данные удобно представлять с помощью элемента управления "переключатель" (или "радиокнопка") в том случае, когда из нескольких вариантов может быть выбран лишь один. Например, для указания возраста можно задать интервалы:

- "моложе 20";
- "от 21 до 30";
- "от 31 до 40";
- "старше 41".

Пользователь может выбрать тот интервал, который определяет его возраст. Оценки, полученные на экзамене, также удобно представить переключателем, т. к. каждый студент способен получить лишь одну из оценок (2, 3, 4 или 5). Для представления информации о полученном зачете достаточно двух радиокнопок. С помощью переключателя можно представлять информацию о семейном положении, образовании и многое другое.

Предположим, что некоторая фирма принимает заказы на изготовление витражей. Заказчик должен выбрать форму витража, количество, указать размеры, материал и др. Требуется определить стоимость заказа. Мы рассмотрим лишь фрагмент, в котором выбирается форма витража. Этот выбор осуществляется с помощью переключателя (или радиокнопки). Элемент "переключатель" отображается в виде круглой кнопки и существует только в составе группы подобных элементов. Может быть осуществлен выбор лишь одного элемента группы. Все элементы группы должны иметь одинаковое значение параметра `name`. Обязательный параметр `value` должен иметь уникальное значение для каждого элемента группы.

Пусть для выбора фигуры задана следующая форма:

```
<FORM name="form1">
  Введите значение
  <input type="text" name="data" size=10><hr>
  Укажите форму:<br>
  <input type="radio" name="fv" value=1>квадрат<br>
```

```
  <input type="radio" name="fv" value=2>круг<br>
  <input type="radio" name="fv" value=3>треугольник<hr>
  <input type="reset" value="Отменить"><hr>
  Площадь:  <input type="text" name="res" size=10>
</FORM>
```

В этой форме шесть элементов. Первый элемент служит для ввода строки текста. Следующие три элемента образуют группу и являются переключателями. Пятый элемент создает кнопку сброса, нажатие которой отменяет все сделанные изменения. Шестой элемент является элементом для ввода строки.

Так как объект `forms` имеет свойство-массив `elements`, в котором содержатся ссылки на элементы формы в порядке их перечисления в теге `<FORM>`, то получить доступ к первому элементу формы можно либо с помощью значения параметра `name` этого элемента (`document.form1.data`), либо используя объектную модель JavaScript (`document.forms[0].elements[0]`). Второй элемент рассматриваемой формы можно получить, если воспользоваться конструкцией `document.forms[0].elements[1]`. Это элемент-переключатель, определенный в составе группы элементов. В рассматриваемом примере группа элементов состоит из трех переключателей. В одну группу входят элементы с одинаковым значением параметра `name`. Доступ к следующим элементам группы может быть осуществлен так: `document.forms[0].elements[2]`, `document.forms[0].elements[3]`. Обязательный параметр `value` должен иметь уникальное значение для каждого элемента группы. Пользователь может выбрать только один вариант.

Вычисление площади фигуры

Напишем сценарий, в котором в зависимости от длины стороны или радиуса и формы выбранного витража вычисляется его площадь. Для простоты будем считать, что витраж может иметь либо форму квадрата (задается его сторона), либо форму круга (задается радиус), либо форму равностороннего треугольника (задается его сторона).

Площадь рассматриваемых фигур считается по формуле $k \times a^2$, где k — коэффициент, зависящий от формы выбранной фигуры; a — задаваемое пользователем значение. Вычисления будут проще, если коэффициент k указать в качестве значения параметра `value` соответствующего переключателя. Щелчок на элементе "переключатель" соответствует событию `Click`, обработка которого заключается в вызове функции `test`. Функция имеет единственный параметр, значение параметра — `value` переключателя, которое служит для вычисления площади фигуры.

HTML-код приведен в листинге 4.1.

Листинг 4.1. Вычисление площади выбранной с помощью переключателя фигуры

```
<HTML>
  <HEAD>
    <TITLE>Данные из формы типа "переключатель". Событие Click</TITLE>
    <script language="JavaScript">
    <!-- //
      function  test (k)
       { var a= form1.data.value
         if (a !="" )
           form1.res.value= k*Math.pow(a,2)
         else alert ("Введите значение")
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      Введите значение
      <input type="text" name="data" size=10>
      <hr>
      Укажите форму <br>
      <input type="radio" name="fv" value=1
             onClick="test(form1.elements[1].value)">квадрат<br>
      <input type="radio" name="fv" value=3.14
             onClick="test(form1.elements[2].value)">круг<br>
      <input type="radio" name="fv" value=0.42
             onClick="test(form1.elements[3].value)">треугольник<hr>
      <input type="reset" value="Отменить"><hr>
      Площадь:  <input type="text" name="res" size=10>
    </FORM>
  </BODY>
</HTML>
```

Перепишем функцию `test`, осуществляющую доступ к элементам формы через свойство-массив `elements`.

```
function  test (k)
 { var a= document.forms[0].elements[0].value
   if (a !="" )
```

```
    document.forms[0].elements[5].value= k*Math.pow(a,2)
  else alert ("Введите значение")
}
```

В следующем примере проверяется правильность обработки данных элемента управления "переключатель". Объект `radio` имеет свойства `name`, `type`, `value`. Событие `Focus` возникает, когда элемент формы получает фокус. При получении фокуса переключателем в поле для многострочного текста отображаются значения свойства для выбранного переключателя.

Свойства переключателя

Необходимо написать сценарий, в котором для каждого переключателя выводятся значения соответствующих ему свойств, как на рис. 4.1.

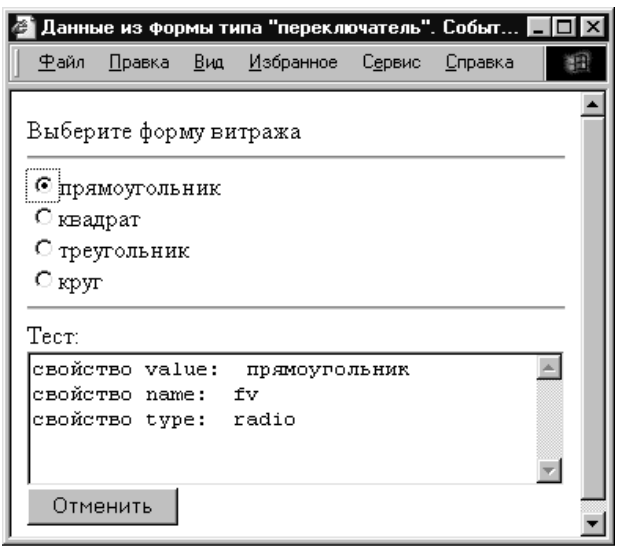


Рис. 4.1. Свойства переключателя

Группа элементов состоит из четырех переключателей, пятый элемент формы — поле ввода многострочного текста. При выполнении сценария формируется строка `s`, содержащая требуемые значения. Фактическим параметром функции обработки события `test` является элемент формы — переключатель, а не значение параметра `value` выбранного переключателя, как в предыдущем примере. Далее в сценарии для получения свойств переключателя используются конструкции `obj.value`, `obj.name`, `obj.type`. В этом примере в качестве значения параметра `value` в переключателе применяется строковый литерал.

HTML-код приведен в листинге 4.2.

Листинг 4.2. Свойства выбранного элемента-переключателя

```
<HTML>
  <HEAD>
    <TITLE>Данные из формы типа "переключатель". Событие Focus</TITLE>
    <script language="JavaScript">
    <!-- //
      function  test (obj)
       { var s="свойство value:  " + obj.value + "\n\r" +
               "свойство name:  " + obj.name + "\n\r" +
               "свойство type:  " + obj.type
         form1.elements[4].value=s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      Выберите форму витража<hr>
      <input type="radio" name="fv" value="прямоугольник"
             onFocus=test(form1.elements[0])>прямоугольник<br>
      <input type="radio" name="fv" value="квадрат"
             onFocus=test(form1.elements[1])>квадрат<br>
      <input type="radio" name="fv" value="треугольник"
             onFocus=test(form1.elements[2])>треугольник<br>
      <input type="radio" name="fv" value="круг"
             onFocus=test(form1.elements[3])>круг<hr>
      Тест: <br>
      <textarea name="res" cols=55 rows=5></textarea><br>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Объект "переключатель" содержит свойство `form`, значение которого соответствует форме, где расположен переключатель. Это обеспечивает доступ к объекту-родителю, а, следовательно, и к его свойствам.

Свойства формы

Напишем сценарий, в котором для каждого переключателя выводятся значения соответствующего ему свойства `value`, а также свойства `name` и `length` той формы, на которой расположен переключатель.

Если, как в предыдущем примере, в функцию `test` будем передавать в качестве параметра выбранный элемент. Тогда в сценарии, выполняя конструкцию `obj.form`, мы получим доступ к форме, на которой расположен элемент. Чтобы определить имя формы и количество элементов на ней, следует обратиться к свойствам `obj.form.name` и `obj.form.length`. Поле ввода многострочного текста в форме является пятым элементом управления, для изменения значения `value` которого в сценарии можно воспользоваться оператором присваивания

```
obj.form.elements[4].value=s
```

Таким образом, от переключателя перешли к форме, а в форме выбрали нужный элемент, используя свойство-массив `elements`. Событие `Blur` наступает, когда элемент формы теряет фокус. В этом случае переключателем в поле ввода многострочного текста помещаются свойства формы, на которой расположен переключатель. HTML-код со сценарием приведен в листинге 4.3.

Листинг 4.3. Свойства формы, в которой расположен переключатель

```
<HTML>
  <HEAD>
    <TITLE>Данные из формы типа "переключатель". Событие Blur</TITLE>
    <script language="JavaScript">
    <!-- //
      function  test (obj)
       { var s="свойство value переключателя: " + obj.value + "\n\r" +
               "свойство name формы: "+obj.form.name + "\n\r" +
               "Число элементов формы: " + obj.form.length
         obj.form.elements[4].value=s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      Выберите форму витража<hr>
      <input type="radio" name="fv" value="прямоугольник"
             onBlur="test(form1.elements[0])">прямоугольник<br>
```

```
      <input type="radio" name="fv" value="квадрат"
             onBlur="test(form1.elements[1])">квадрат<br>
      <input type="radio" name="fv" value="треугольник"
             onBlur="test(form1.elements[2])">треугольник<br>
      <input type="radio" name="fv" value="круг"
             onBlur="test(form1.elements[3])">круг<hr>
      Тест: <br>
      <textarea name="res"  cols=45 rows=5></textarea><br>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

На рис. 4.2 приведен вид документа, когда третий переключатель формы потерял фокус.

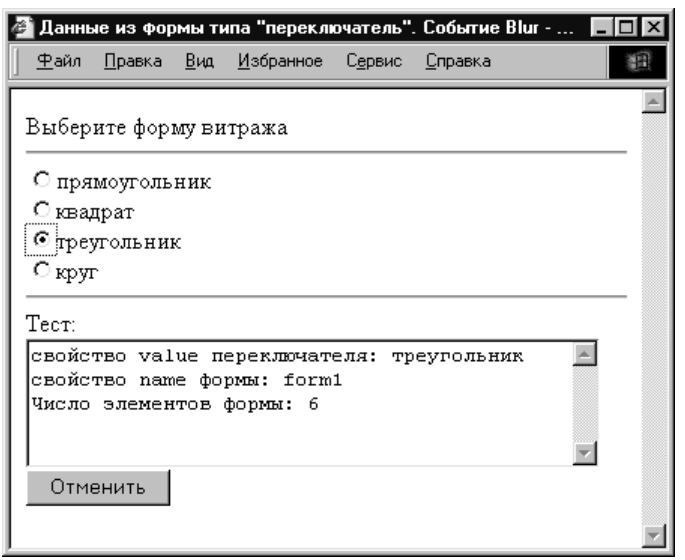


Рис. 4.2. Свойства формы с переключателем

Определение выделенного элемента

Необходимо написать сценарий, в котором определяется выбранный переключатель при щелчке мышью по кнопке в форме.

Свойство `checked` переключателя возвращает логическое значение `true`, если тот установлен, и `false` — в противном случае. Выяснить, установлен ли

переключатель, являющийся первым элементом формы с именем `form1`, можно по значению следующего выражения:

```
(document.form1.elements[0]).checked
```

Приведем программу, которая демонстрирует работу с переключателем (листинг 4.4).

Листинг 4.4. Определение выделенного элемента в переключателе

```
<HTML>
  <HEAD>
    <TITLE>Данные из формы типа "переключатель". Выделенный элемент</TITLE>
    <script language="JavaScript">
    <!-- //
      function vid()
       { var d=document
         var k=0;
         if ((d.form1.elements[0]).checked)
           k=(d.form1.elements[0]).value
         else
           if ((d.form1.elements[1]).checked)
             k=(d.form1.elements[1]).value
           else
             if ((d.form1.elements[2]).checked)
               k=(d.form1.elements[2]).value
             else
               if ((d.form1.elements[3]).checked)
                 k=(d.form1.elements[3]).value
         d.form1.elements[6].value=k
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      Выберите форму витража<hr>
      <input type="radio" name="fv" value=1>прямоугольник<br>
      <input type="radio" name="fv" value=2>квадрат<br>
      <input type="radio" name="fv" value=3>треугольник<br>
```

```
      <input type="radio" name="fv" value=4>круг<hr>
      <input type="button" value="Выполнить" onClick="vid()">
      <input type="reset" value="Отменить"><hr>
      Тест: <input type="text" name="res">
    </FORM>
  </BODY>
</HTML>
```

В следующем примере продемонстрируем использование параметра `id` для анализа данных, задаваемых с помощью переключателя.

Уникальные имена

Напишем сценарий, в котором определяется выбранный переключатель при щелчке мышью по определяемой в форме кнопке. Доступ к элементу требуется осуществить, используя уникальные имена (листинг 4.5).

Листинг 4.5. Уникальные имена

```
<HTML>
  <HEAD>
    <TITLE>Данные, представленные переключателем. Идентификаторы</TITLE>
    <script language="JavaScript">
    <!-- //
      function vid()
       { var d=document
         var k=0;
         if (d.all("i1").checked) k=1
         else
           if (d.all("i2").checked) k=2
           else
             if (d.all("i3").checked) k=3
             else
               if (d.all("i4").checked) k=4
         d.all("resch").value=k
       }
    //-->
    </script>
  </HEAD>
```

```
  <BODY>
    <FORM name="form1">
      Выберите форму витража<hr>
      <input type="radio" name="fw" value=1 id="i1">прямоугольник<br>
      <input type="radio" name="fw" value=2 id="i2">квадрат<br>
      <input type="radio" name="fw" value=3 id="i3">треугольник<br>
      <input type="radio" name="fw" value=4 id="i4">круг<br>
      <hr>
      <input type="button" value="Выполнить" onClick="vid()">
      <input type="reset" value="Отменить"><hr>
      <input type="text" name="res" id="resch">
    </FORM>
  </BODY>
</HTML>
```

Выбор параметров обтекания изображения текстом

Напишем сценарий, который предоставляет возможность пользователю задавать значения параметров, определяющих, к какому полю окна (левому или правому) прижимается изображение, и, соответственно, с какой стороны текст его обтекает. Кроме того, требуется предусмотреть возможность задания величины отступов по вертикали и горизонтали, отделяющих текст от изображения.

Если значение параметра `align` равно `Left`, то изображение прижимается к левому краю окна просмотра браузера, а текст или другие элементы документа "обтекают" изображение с правой стороны. Текст, размещаемый рядом с изображением, может занимать несколько строк. В примере на рис. 4.3 значение параметра `align` задано равным `Right`, поэтому изображение прижато к правому полю окна браузера, а текст обтекает изображение слева. Текст занимает несколько строк. Для того чтобы изображение и текст не "сливались", можно задать параметры, значения которых определяют величину отступа текста от изображения по горизонтали и вертикали. В приведенном примере значения этих величин равны 15 и 10 соответственно. По умолчанию значение параметра `align` равно `Left`, а значение отступов — 0. При нажатии на кнопку **Обновить** для изображения и текста будут установлены значения параметров, принимаемых по умолчанию.

Пример HTML-кода, который содержит сценарий, обеспечивающий выполнение действий, задаваемых пользователем, приведен в листинге 4.6.

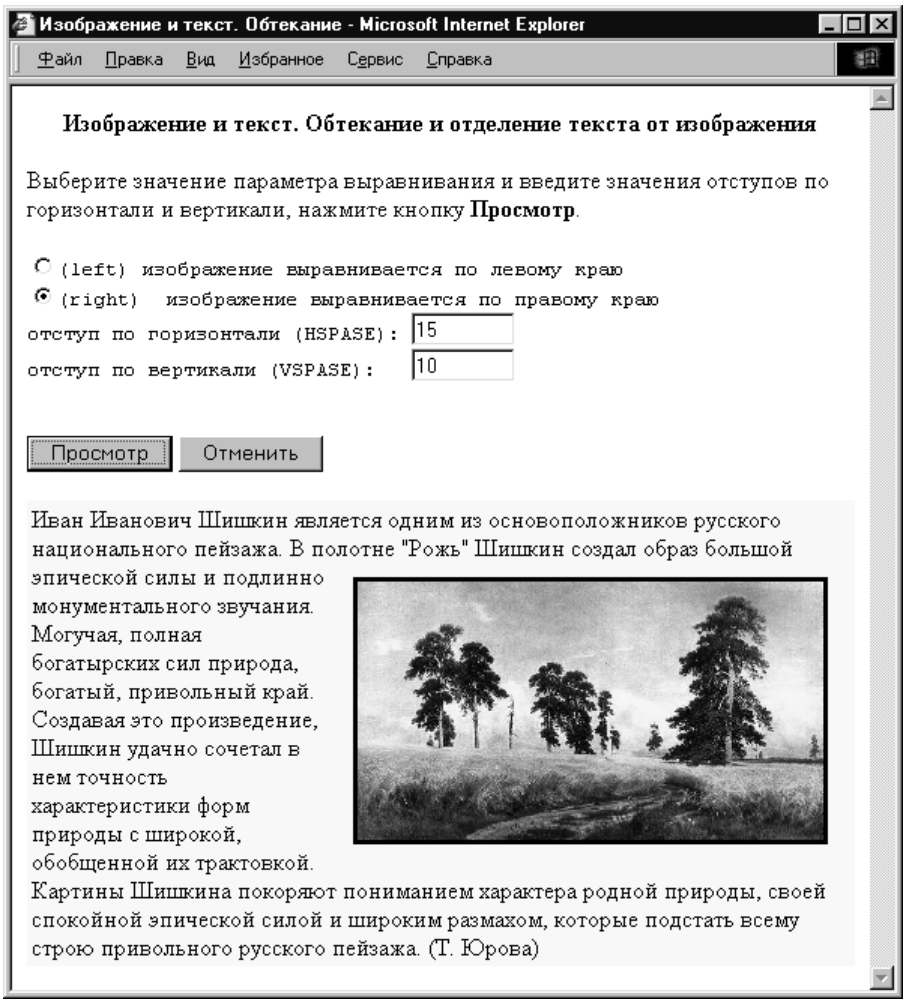


Рис. 4.3. Обтекание текстом изображения

Листинг 4.6. Обтекание текстом изображения

```
<HTML>
  <HEAD>
    <TITLE>Изображение и текст. Обтекание</TITLE>
    <script>
    <!--
      function chpict(obj)
       { var h=obj.hsp.value
```

```
        var v=obj.vsp.value
        document.mypict.hspace=h
        document.mypict.vspace=v
        if ((obj.elements[0]).checked)
          document.mypict.align="Left"
        else
          document.mypict.align="Right"
       }
      function rset(obj)
       { document.mypict.align="Left"
        document.mypict.hspace=0
        document.mypict.vspace=0
        obj.hsp.value=0
        obj.vsp.value=0
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <CENTER>
      <H4>Изображение и текст.
          Обтекание и отделение текста от изображения</H4>
    </CENTER>
    <FORM name="form1">
      Выберите значение параметра выравнивания
      и введите значения отступов по горизонтали и вертикали,
      нажмите кнопку <B>Просмотр</B>.<br>
      <PRE>
<input type="radio" name="alg" checked value=ld>(left) изображение
выравнивается по левому краю
<input type="radio" name="alg" value=rd>(right)  изображение
выравнивается по правому краю
отступ по горизонтали (HSPASE): <input type="text" name="hsp" size=8
value=0>
отступ по вертикали (VSPASE):   <input type="text" name="vsp" size=8
value=0>
      </PRE>
      <input type="button" value= "Просмотр"  onclick="chpict(form1)">
      <input type="reset" value="Отменить"    onclick="rset(form1)">
    </FORM>
```

```
    <TABLE bgcolor="F8F8FF">
      <TR><TD>Иван Иванович Шишкин является одним из основоположников
            русского национального пейзажа.
          <IMG src=p1.jpg name=mypict align=left border=3 width=310>
            В полотне "Рожь" Шишкин создал образ большой эпической силы
            и подлинно монументального звучания. Могучая, полная
            богатырских сил природа, богатый, привольный край.
            Создавая это произведение, Шишкин удачно сочетал в нем
            точность характеристики форм природы с широкой, обобщенной
            их трактовкой. Картины Шишкина покоряют пониманием
            характера родной природы, своей спокойной эпической силой
            и широким размахом, которые подстать всему строю
            привольного русского пейзажа. (Т. Юрова)
      </TD></TR>
    </TABLE>
  </BODY>
</HTML>
```

Если изображение рассматривается как элемент строки, то значения параметров выравнивания задают расположение изображения относительно строки текста. Верхняя граница изображения может быть выровнена либо по самому высокому текстовому элементу текущей строки, либо по самому высокому элементу в строке (например, другому изображению). Базовой считается нижняя часть линии текста, которая проводится без учета нижней части некоторых символов. Середину изображения можно выровнять либо по базовой линии, либо по середине текущей строки. Нижнюю часть изображения можно выровнять по базовой линии, либо по нижней границе текущей строки.

Размещение изображения относительно строки

Напишем сценарий, позволяющий по значению параметра выравнивания изображения `align` определить те действия, которые будут выполняться при заданном значении.

Параметр `align` в рассматриваемом случае может принимать одно из шести значений, которые задаются переключателем. На рис. 4.4 приведен документ в случае, когда выбрано значение параметра `align` равное `TEXTTOP`.

При задании пользователем значения параметра выравнивания выбранный элемент получает фокус, и как реакция на это событие, выполняется функция `set`. В функции `set` с параметром `obj` анализируется значение `obj.value`, и переменной `s` присваивается соответствующее значение.

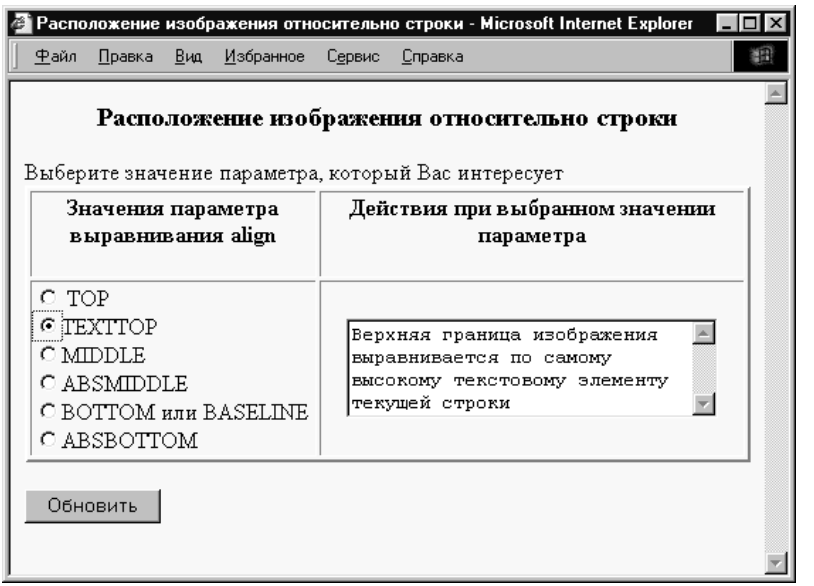


Рис. 4.4. Параметры горизонтального выравнивания

HTML-код, содержащий сценарий, обеспечивающий описание действий в зависимости от значения параметра, приведен в листинге 4.7.

Листинг 4.7. Расположение изображения относительно строки

```
<HTML>
  <HEAD>
    <TITLE>Расположение изображения относительно строки</TITLE>
    <script>
    <!--
      var s1="Верхняя граница изображения выравнивается " +
             "по самому высокому элементу текущей строки"
      var s2="Верхняя граница изображения выравнивается " +
             "по самому высокому текстовому элементу текущей строки"
      var s3="Выравнивание середины изображения "+
             "по базовой линии текущей строки"
      var s4="Выравнивание середины изображения " +
             "по середине текущей строки"
      var s5="Выравнивание нижней границы изображения по базовой линии "+
             "текущей строки"
      var s6="Выравнивание нижней границы изображения " +
             "по нижней границе текущей строки"
```

```
    var s=""
    function set(obj)
     { switch (Number(obj.value))
      { case 0: s=s1; break;
        case 1: s=s2; break;
        case 2: s=s3; break;
        case 3: s=s4; break;
        case 4: s=s5; break;
        case 5: s=s6; break;
      }
      obj.form.elements[6].value=s
     }
  //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <H3 align=center>Расположение изображения относительно строки</H3>
  Выберите значение параметра, который Вас интересует
  <TABLE border=2>
    <TR><TD align=center><H4>Значения параметра выравнивания align</H4>
        <TD align=center><H4>Действия при выбранном значении
                            параметра</H4></TR>
    <TR><TD>
      <FORM name="form1">
        <input type="radio" name="ln" value=0 checked
               onFocus="set(form1.elements[0])">TOP<br>
        <input type="radio" name="ln" value=1
               onFocus="set(form1.elements[1])">TEXTTOP<br>
        <input type="radio" name="ln" value=2
               onFocus="set(form1.elements[2])">MIDDLE<br>
        <input type="radio" name="ln" value=3
               onFocus="set(form1.elements[3])">ABSMIDDLE<br>
        <input type="radio" name="ln" value=4
               onFocus="set(form1.elements[4])">BOTTOM или BASELINE<br>
        <input type="radio" name="ln" value=5
               onFocus="set(form1.elements[5])">ABSBOTTOM<br>
      <FORM name="form1">
    </TD>
    <TD align=center> <textarea name="res" cols=30 rows=4>
      Верхняя граница изображения выравнивается по самому высокому
      элементу текущей строки </textarea><br></TD></TR>
```

```
    </TABLE>
    <P>
    <input type="reset" value="Обновить">
  </BODY>
</HTML>
```

В следующем примере предлагается написать сценарий, который демонстрирует, как будет изменяться документ при выборе различных параметров выравнивания изображения.

Изображение как часть строки

Напишем сценарий, который позволяет продемонстрировать, как изменяется текущая строка в зависимости от значения параметра выравнивания изображения, рассматриваемого как элемент строки.

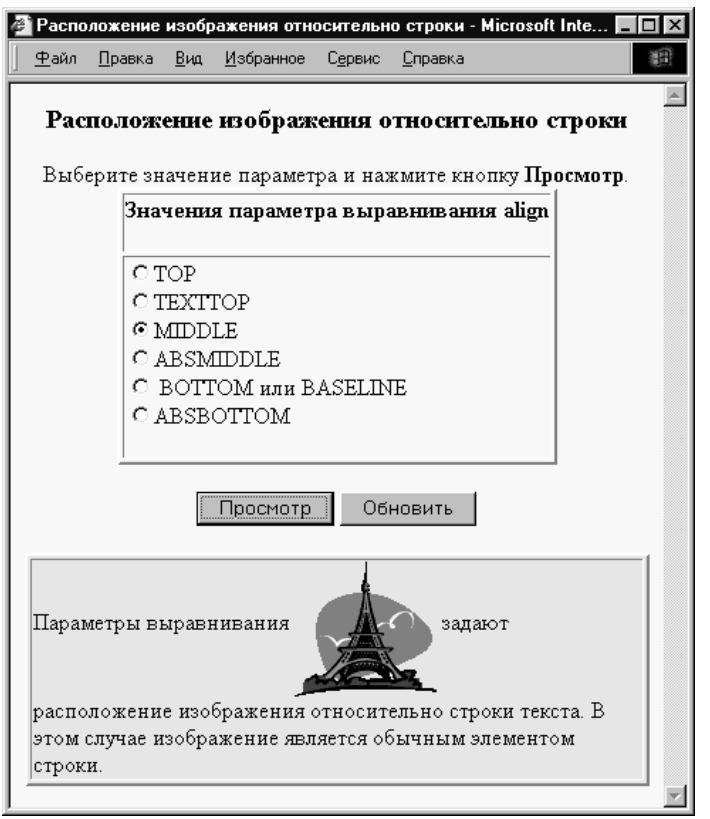


Рис. 4.5. Расположение изображения относительно строки

Документ состоит из двух частей. В верхней части перечислены возможные значения параметра выравнивания изображения, рассматриваемого как элемент строки. В нижней части документа располагаются текст и изображение, к которому применяются выбранные действия при нажатии кнопки **Просмотр**. Рис. 4.5 соответствует ситуации, при которой происходит выравнивание середины изображения по базовой линии текущей строки.

Документ, содержащий HTML-код для решения задачи, представлен в листинге 4.8.

Листинг 4.8. Изображение как часть строки. Параметры выравнивания

```
<HTML>
  <HEAD>
    <TITLE>Расположение изображения относительно строки</TITLE>
    <script>
    <!--
      function set(obj)
       { if(obj.elements[0].checked) document.mypict.align="TOP"
         else
           if(obj.elements[1].checked) document.mypict.align="TEXTTOP"
           else
             if(obj.elements[2].checked) document.mypict.align="MIDDLE"
             else
              if(obj.elements[3].checked)
                document.mypict.align="ABSMIDDLE"
              else
                if(obj.elements[4].checked)
                  document.mypict.align="BOTTOM"
                else
                  if(obj.elements[5].checked)
                    document.mypict.align= "ABSBOTTOM"
      }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <CENTER>
    <H3 align=center>Расположение изображения относительно строки</H3>
```

```
    Выберите значение параметра и нажмите кнопку <B>Просмотр</B>.
    <TABLE border=2>
      <TR>
        <TD><H4>Значения параметра выравнивания align</H4>
      </TR>
      <TR><TD>
        <FORM name="form1">
          <input type="radio" name="im" value="top" checked>TOP<br>
          <input type="radio" name="im" value="texttop">TEXTTOP<br>
          <input type="radio" name="im" value="middle">MIDDLE<br>
          <input type="radio" name="im" value="absmiddle">ABSMIDDLE<br>
          <input type="radio" name="im" value="botton">
            BOTTOM или BASELINE <br>
          <input type="radio" name="im" value="absbotton">ABSBOTTOM<br>
        </FORM>
      </TD></TR>
    </TABLE><br>
    <input type="button" value= "Просмотр"  onclick="set(form1)">
    <input type="reset" value="Обновить"><P>
    </CENTER>
    <TABLE border=2 bgcolor="#FFDCDC">
      <TR><TD>Параметры выравнивания <IMG src="p511.jpg" name="mypict"
                                        align=TOP>
              задают расположение изображения относительно строки текста.
              В этом случае изображение является обычным элементом
              строки.
      </TD></TR></TABLE>
  </BODY>
</HTML>
```

Во многих случаях данные удобно представить в виде таблицы. Поскольку таблица содержит ячейки, то для каждой из ячеек можно определить, как разместить в ней данные. Горизонтальное выравнивание позволяет содержимое ячейки расположить по центру, левому или правому краю. Кроме того, текст или графику внутри ячейки можно разместить, определив параметры вертикального выравнивания.

Расположение текста и изображения в ячейке таблицы

Напишем сценарий, который позволяет продемонстрировать, как изменяется содержимое ячейки таблицы в зависимости от значений параметров горизонтального и вертикального выравнивания.

В документе слева находится список параметров, которые пользователь выбирает для горизонтального и вертикального выравнивания. В правой части в ячейке таблицы хранится изображение, положение которого будет меняться в зависимости от выбранных переключателей после нажатия кнопки **Просмотр** (рис. 4.6).

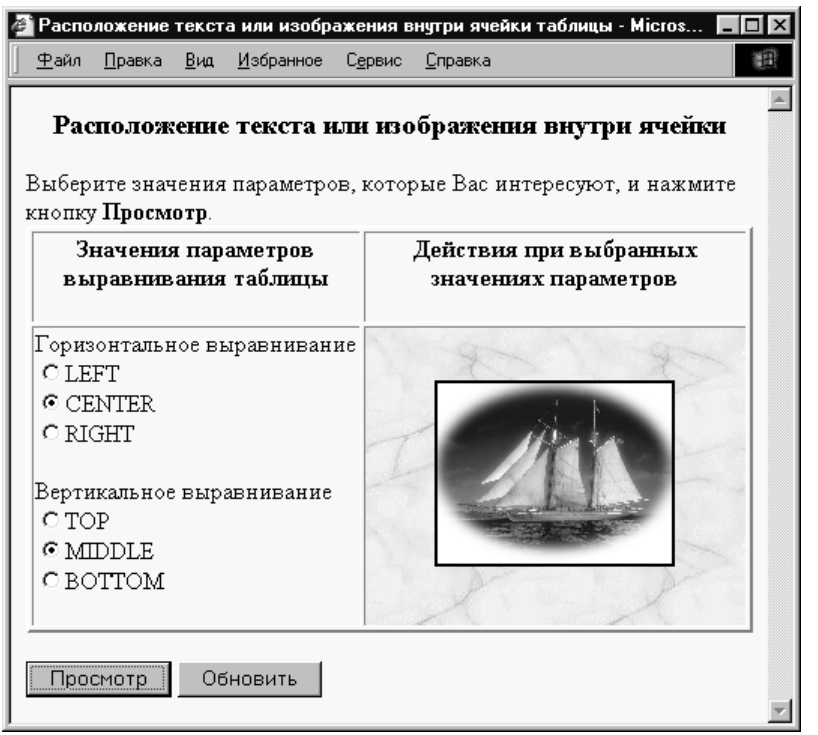


Рис. 4.6. Расположение изображения внутри ячейки

После нажатия кнопки **Просмотр** выполняется функция, анализирующая, какие параметры выбраны, и присваивает определенные значения атрибутам выравнивания содержимого ячейки. В правой части документа изображение занимает определенное параметрами выравнивания место в ячейке таблицы. Доступ к ячейке таблицы осуществляется по значению параметра `id`. При

нажатии кнопки **Обновить** изображение требуется вернуть на место. По умолчанию горизонтальное выравнивание предполагает расположение содержимого по левому краю, вертикальное выравнивание — по верхней границе. HTML-код, содержащий сценарии, представлен в листинге 4.9.

Листинг 4.9. Расположение текста или изображения внутри ячейки таблицы

```
<HTML>
  <HEAD>
    <TITLE>Расположение текста или изображения
           внутри ячейки таблицы</TITLE>
    <script>
    <!--
      function set(obj)
       { var d=document
         if(obj.elements[0].checked) {d.all("itab").align="LEFT"}
         else
           if(obj.elements[1].checked) {d.all("itab").align="MIDDLE"}
           else
             if(obj.elements[2].checked) {d.all("itab").align="RIGHT"}
         if(obj.elements[3].checked) {d.all("itab").vAlign="TOP"}
         else
           if(obj.elements[4].checked) {d.all("itab").vAlign="MIDDLE"}
           else
             if(obj.elements[5].checked) {d.all("itab").vAlign="BOTTOM"}
       }
      function rset(obj)
       { var d=document
         d.all("itab").align="LEFT"
         d.all("itab").vAlign="TOP"
       }
    //-->
    </script>
  <BODY bgcolor="F8F8FF">
    <H3 align=center>Расположение текста или  изображения
                    внутри ячейки</H3>
    Выберите значения параметров, которые Вас интересуют,
    и нажмите кнопку <B>Просмотр</B>.
    <TABLE border=2>
```

```
      <TR>
        <TD ><H4 align=center>Значения параметров выравнивания
                              таблицы</H4>
        <TD><H4 align=center>Действия при выбранных значениях
                             параметров</H4>
      </TR>
      <TR>
        <TD>
          <FORM name="form1">
            Горизонтальное выравнивание<br>
            <input type="radio" name="al" value=0 checked>LEFT<br>
            <input type="radio" name="al" value=1>CENTER<br>
            <input type="radio" name="al" value=2>RIGHT<P>
            Вертикальное выравнивание<br>
            <input type="radio" name="vl" value=3 checked>TOP<br>
            <input type="radio" name="vl" value=4>MIDDLE<br>
            <input type="radio" name="vl" value=5>BOTTOM<br>
          </FORM>
        </TD>
        <TD id="itab" bgcolor=blue background="firering.jpg" height=160
            valign=TOP align=LEFT>
          <IMG src="6rkranim.gif" border=2 height=120>
        </TD>
      </TR>
    </TABLE><P>
    <input type="button" value= "Просмотр"  onclick="set(form1)">
    <input type="reset" value="Обновить" onclick="rset()">
  </BODY>
</HTML>
```

Таблицы в HTML используются не только для представления табличных данных, но и как средство размещения на странице различных текстовых и графических объектов. Можно задать цвет фона таблицы или фоновое изображение, отличное от цвета фона или фонового изображения всего документа. Рамку с тенью можно указать, варьируя толщину рамки таблицы. Интересные эффекты получаются, если в таблицу поместить изображение, а затем подбирать для нее цвет фона. При выборе фона изображение каждый раз будет в разном обрамлении.

Фоновое изображение таблицы

Необходимо написать сценарий, который позволяет выбрать фоновое изображение для таблицы, содержащей рисунок. Требуется предусмотреть возможность изменения толщины рамки таблицы для того, чтобы создать эффект выпуклости изображения.

Варианты для выбора фонового изображения расположим в документе слева, а справа поместим таблицу, для которой определяется фон, и которая содержит изображение (рис. 4.7).

Рис. 4.7. Выбор фона таблицы и толщины рамки

Атрибут `CELLSPACING` назначает ширину промежутка между ячейками. В рассматриваемом примере чем больше значение этого параметра, тем шире рамка вокруг изображения. Задание фонового изображения для таблицы определяет цвет рамки вокруг изображения. Имена графических файлов, используемых в качестве фонового изображения: fon1.gif, fon2.gif, ..., fon8.gif. Все графические файлы содержатся в текущей папке.

HTML-код приведен в листинге 4.10.

Листинг 4.10. Выбор фонового изображения для таблицы и задание толщины рамки

```
<HTML>
  <HEAD>
```

```
  <TITLE>Таблицы, графика, фон</TITLE>
  <script>
  <!--
    function gr(m)
     { document.all("tab").background="fon"+m+".gif" }
    function set(obj)
     { if (obj.bor.value != "")
         document.all("tab").border=obj.bor.value}
    function rset()
     { document.all("tab").background="fon4.gif"
       document.all("tab").border=15
       form1.bor.value=15
     }
  //-->
  </script>
</HEAD>
<BODY background="fon1.gif" bgcolor=green>
  <H3 align=center>Таблицы, графика, фон</H3>
  <TABLE border=1 align=center>
    <TR><TD>
      <FORM name="form1">
        Толщина рамки <input type="text" size=8 name=bor
                      value=15 onChange="set(form1)"><br>
        Выберите фон для таблицы<br>
        <TABLE border=2 align=center>
          <TR>
            <TD align=center><IMG src="fon1.gif"
                                  width=20 height=50></TD>
            <TD align=center><IMG src="fon2.gif"
                                  width=20 height=50> </TD>
            <TD align=center><IMG src="fon3.gif"
                                  width=20 height=50></TD>
            <TD align=center><IMG src="fon4.gif"
                                  width=20 height=50></TD>
            <TD align=center><IMG src="fon5.gif"
                                  width=20 height=50></TD>
            <TD align=center><IMG src="fon6.gif"
                                  width=20 height=50></TD>
```

```
            <TD align=center><IMG src="fon7.gif"
                                 width=20 height=50></TD>
            <TD align=center><IMG src="fon8.gif"
                                 width=20 height=50></TD>
          </TR>
          <TR>
            <TD align=center><input type="radio" name="op" value=1
                                 onfocus="gr(1)"></TD>
            <TD align=center><input type="radio" name="op" value=2
                                 onfocus="gr(2)"></TD>
            <TD align=center><input type="radio" name="op" value=3
                                 onfocus="gr(3)"></TD>
            <TD align=center><input type="radio" name="op" value=4
                                 onfocus="gr(4)"></TD>
            <TD align=center><input type="radio" name="op" value=5
                                 onfocus="gr(5)"></TD>
            <TD align=center><input type="radio" name="op" value=6
                                 onfocus="gr(6)"></TD>
            <TD align=center><input type="radio" name="op" value=7
                                 onfocus="gr(7)"></TD>
            <TD align=center><input type="radio" name="op" value=8
                                 onfocus="gr(8)"></TD>
          </TR>
        </TABLE>
      </FORM><br>
      <CENTER>
      <input type="reset" value="Обновить  " onClick="rset()">
    </TD>
    <TD align=center >
      <TABLE id="tab" border=15 cellspacing=30 cellpadding=10
             height=160 bgcolor=green align=center>
        <TR><TD bgcolor=silver><IMG src="bfly.gif" alt="Поздравление"
                                 align=center></TD></TR>
      </TABLE>
    </TD></TR>
  </TABLE>
 </BODY>
</HTML>
```

Фоновое изображение документа, таблицы и ячейки таблицы

Напишем сценарий, который позволяет задать фоновое изображение для документа, таблицы и ячеек таблицы. Сначала пользователь с помощью кнопки-переключателя выбирает часть документа, чье фоновое изображение он хочет изменить, а затем указывает фон.

Как и в предыдущем случае, фоновое изображение задается с помощью кнопок-переключателей. В верхней части документа располагается таблица, состоящая из строки из трех ячеек. В каждую из ячеек помещено одно и то же изображение, размеры которого различны. Далее располагается таблица, в которой задаются параметры с помощью переключателей. Если выбрана ячейка таблицы, то фоновое изображение назначается всем ячейкам. На рис. 4.8 заданы фоновые изображения для всех элементов документа.

Рис. 4.8. Фон для документа, таблицы и ячейки

Доступ к документу, таблице и ячейкам таблицы осуществляется с помощью параметра `id`. При попадании фокуса на переключатель функции обработки события передается параметр, который используется при формировании имени графического файла, соответствующего выбранной кнопке. Если имена графических файлов не удовлетворяют описанному условию, то сценарий несколько усложнится. Задавать фоновое изображение отдельным частям документа можно в произвольном порядке. На рис. 4.8 приведена таблица, толщина рамки которой равна нулю. Приведем HTML-код документа, изображенного на рисунке (листинг 4.11).

Листинг 4.11. Фоновое изображение документа, таблицы, ячейки

```
<HTML>
  <HEAD>
    <TITLE>Фоновое изображение документа, таблицы, ячейки</TITLE>
    <script>
    <!--
      function  gr(i)
       { var d=document
         if (d.form1.bor.value !="")
           { d.all("itab").border=Number(d.form1.bor.value) }
         if (d.form1.elements[1].checked)
          { d.all("doc").background="fon" +i+".gif" }
         else
           if (d.form1.elements[2].checked)
            { d.all("itab").background="fon" +i+".gif" }
           else
             if (d.form1.elements[3].checked)
              { d.all("itab1").background="fon" +i+".gif"
                d.all("itab2").background="fon" +i+".gif"
                d.all("itab3").background="fon" +i+".gif"
              }
       }
    //-->
    </script>
  </HEAD>
  <BODY id="doc">
    <H4 align=center>Фоновое изображение документа, таблицы, ячейки</H4>
    <TABLE id="itab" border=1 cellspacing=20
           cellpadding =10 align=center>
```

```
    <TR><TD id="itab1" bgcolor=silver><IMG src="p1.gif" width=30></TD>
        <TD id="itab2" bgcolor=silver><IMG src="p1.gif" width=50></TD>
        <TD id="itab3" bgcolor=silver><IMG src="p1.gif" width=100></TD>
    </TR>
  </TABLE>
  <FORM name="form1"><CENTER>
    Введите значение толщины рамки (BORDER)
    <input type="text" size=8 name=bor
onChange='document.all("itab").border=1*(document.form1.bor.value)'><br>
    Для изменения фонового изображения выберите, в какой части
    документа хотите его изменить, выберите фон<br>
    <TABLE border=2 align=center>
      <TR><TD colspan=8>
        <input type="radio" name= "fp" value="doc">документ
        <input type="radio" name= "fp" value="tab">таблица
        <input type="radio" name= "fp" value="ct">ячейка
      </TD></TR>
      <TR>
        <TD align=center><IMG src="fon1.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon2.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon3.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon4.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon5.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon6.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon7.gif" width=20 height=50></TD>
        <TD align=center><IMG src="fon8.gif" width=20 height=50></TD>
      </TR>
      <TR><TD align=center>
        <input type="radio" name="op" value=1 onfocus="gr(1)"></TD>
      <TD align=center>
         <input type="radio" name="op" value=2 onfocus="gr(2)"></TD>
      <TD align=center>
         <input type="radio" name="op" value=3 onfocus="gr(3)"></TD>
      <TD align=center>
          <input type="radio" name="op" value=4 onfocus="gr(4)"></TD>
      <TD align=center>
          <input type="radio" name="op" value=5 onfocus="gr(5)"></TD>
      <TD align=center>
          <input type="radio" name="op" value=6 onfocus="gr(6)"></TD>
      <TD align=center>
```

```
            <input type="radio" name="op" value=7 onfocus="gr(7)"></TD>
        <TD align=center>
            <input type="radio" name="op" value=8 onfocus="gr(8)"></TD>
        </TR>
      </TABLE><br>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите сценарий, который позволяет продемонстрировать изменения размеров и положения на странице горизонтальной линии так, как показано на рис. 4.9.

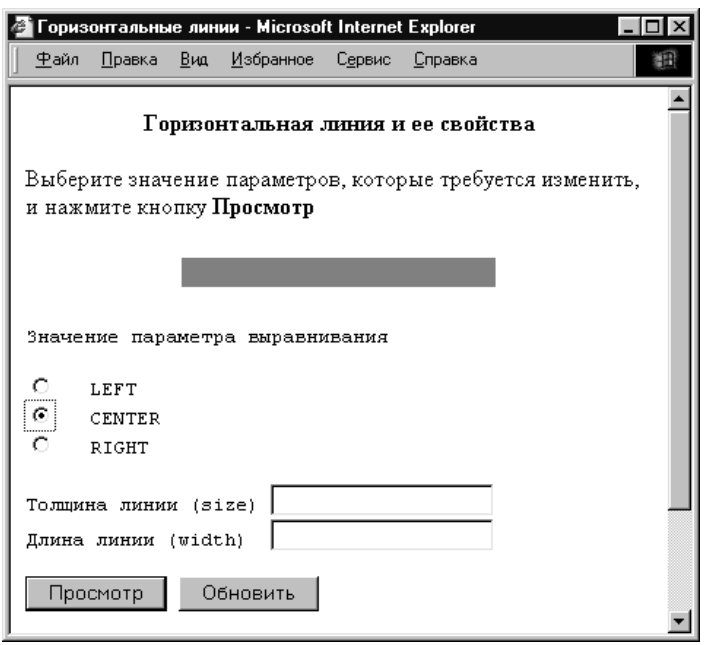


Рис. 4.9. Исследование свойств горизонтальной линии

2. Разработайте анкету, определяющую характер человека. Пользователю предлагается ответить "да" или "нет" на следующие вопросы.
   - Считаете ли Вы, что у многих ваших знакомых хороший характер?

- Раздражают ли Вас мелкие повседневные обязанности?
- Верите ли Вы, что ваши друзья преданы Вам?
- Неприятно ли Вам, когда незнакомый человек делает Вам замечание?
- Способны ли Вы ударить собаку или кошку?
- Часто ли Вы принимаете лекарства?
- Часто ли Вы меняете магазин, в который ходите за продуктами?
- Продолжаете ли Вы отстаивать свою точку зрения, поняв, что ошиблись?
- Тяготят ли Вас общественные обязанности?
- Способны ли Вы ждать более 5 минут, не проявляя беспокойства?
- Часто ли Вам приходят в голову мысли о Вашей невезучести?
- Сохранилась ли у Вас фигура по сравнению с прошлым?
- Можете ли Вы с улыбкой воспринимать подтрунивание друзей?
- Нравится ли Вам семейная жизнь?
- Злопамятны ли Вы?
- Находите ли Вы, что стоит погода, типичная для данного времени года?
- Случается ли Вам с утра быть в плохом настроении?
- Раздражает ли Вас современная живопись?
- Надоедает ли Вам присутствие чужих детей в доме более одного часа?

Поставьте плюс, если вы ответили "да" в вопросах с номерами 3, 9, 10, 13, 14, 19 и "нет" в вопросах с номерами 1, 2, 4, 5, 7, 8, 11, 12, 14, 16, 17, 18. Посчитайте сумму набранных очков. Если она оказалась более 15, то у вас покладистый характер; если сумма в интервале от 8 до 15, то вы не лишены недостатков, но с вами можно ладить; если сумма менее 8 очков, то вашим друзьям можно посочувствовать.

Напишите сценарий обработки теста. Ответ на каждый из вопросов представьте с помощью переключателя. Для каждого из трех возможных результатов предусмотрите вывод соответствующего результату изображения.

3. Разработайте анкету, в которой для пяти преподавателей оценивается качество преподавания по критериям, приведенным в таблице на рис. 4.10. Каждый из критериев оценивается по пятибалльной системе. Требуется написать сценарий, определяющий для каждого преподавателя набранную сумму баллов. Полученные результаты оформить в виде диаграммы.

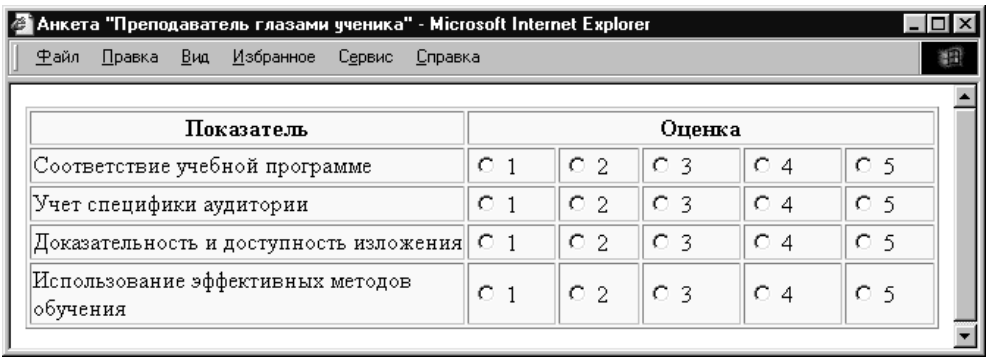


Рис. 4.10. Показатели качества преподавания

4. Разработайте анкету для анализа ответов на вопросы по теме "Графический редактор: назначение и применение". Для заданных в анкете вопросов пользователь должен выбрать один из правильных ответов. Ответы должны быть представлены с помощью переключателей. Напишите сценарий обработки результатов опроса.

   Выберите один правильный ответ из данных.

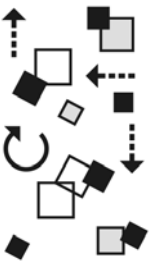
   - Графика с представлением изображения в виде совокупности отрезков прямых называется…
     - ◊ Фрактальной
     - ◊ Векторной
     - ◊ Прямолинейной
     - ◊ Растровой
   - Разрешающая способность экрана в текстовом режиме определяется количеством…
     - ◊ Байтов на символ
     - ◊ Символов в строке экрана
     - ◊ Пикселов по горизонтали и вертикали
     - ◊ Строк и столбцов на экране
     - ◊ Строк на экране
   - При работе дисплея в текстовом режиме одну позицию экрана занимает…
     - ◊ Один пиксел
     - ◊ Один символ

- ◊ Одно слово
- ◊ Часть символа
- ◊ Восемь пикселов

- Информация о графическом изображении в видеопамяти формируется...
  - ◊ Графическим процессором
  - ◊ Центральным процессором
  - ◊ Графическим адаптером
  - ◊ Дисплейным процессором
  - ◊ Дисплеем
- Разрешающая способность экрана в графическом режиме определяется количеством...
  - ◊ Строк на экране и количеством символов в строке
  - ◊ Пикселов по вертикали
  - ◊ Объемом видеопамяти на пиксел
  - ◊ Пикселов на единицу длины
- Пиктограмма — это...
  - ◊ Точка на графическом экране
  - ◊ Рисунок на рабочем поле графического редактора
  - ◊ Файл, в котором сохранено графическое изображение
  - ◊ Текстовая информация в рабочем поле графического редактора
  - ◊ Условный рисунок, элемент меню графического редактора
- Цвет точки на экране цветного монитора формируется из сигналов основных цветов:
  - ◊ Красного, синего, зеленого
  - ◊ Белого и черного
  - ◊ Красного, синего, зеленого, коричневого
  - ◊ Белого, черного и яркости
- Разрешающая способность — это...
  - ◊ Число точек экрана
  - ◊ Количество пикселов на один квадратный дюйм
  - ◊ Число точек на единицу длины
  - ◊ Ступенчатый характер изображения
  - ◊ Ничто из перечисленного выше не верно

- Минимальным объектом, используемым в векторном графическом редакторе, является…
  - ◊ Точка экрана (пиксел)
  - ◊ Объект (прямоугольник, круг и т. д.)
  - ◊ Палитра цветов
  - ◊ Знакоместо (символ)
- Минимальным объектом, используемым в растровом графическом редакторе, является…
  - ◊ Точка экрана (пиксел)
  - ◊ Объект (прямоугольник, круг и т. д.)
  - ◊ Палитра цветов
  - ◊ Знакоместо (символ)

Глава 5



Флажки

Элемент управления "флажок" используется в случае, когда из предложенных вариантов можно выбрать как один, так и несколько. Каждый вариант выбора задается флажком, который можно либо установить, либо сбросить. Флажок определяется в теге `<input>` значением `checkbox` параметра `type`. Обязательным параметром является параметр `value`, значение которого будет передано на обработку в случае выбора нажатием кнопки.

Выбор характеристик издания

Предположим, читателю предлагается заполнить анкету, в которой требуется указать название любимого издания и выбрать из предложенного списка характеристики, которые присущи рассматриваемому изданию.

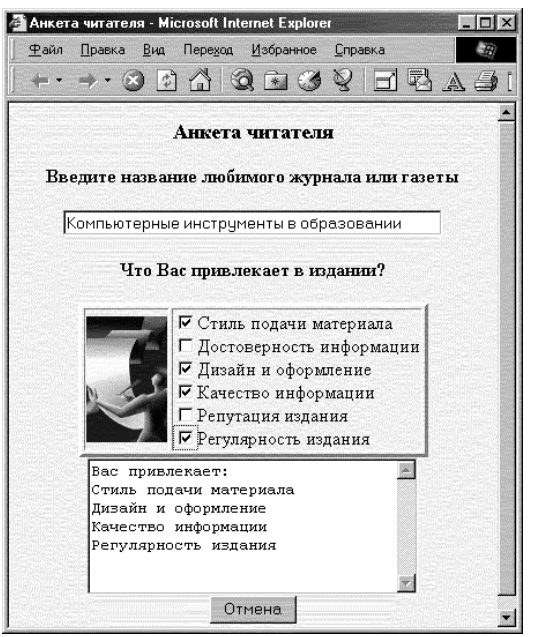


Рис. 5.1. Пример обработки анкеты читателя

Для задания характеристик издания можно воспользоваться флажком. Пользователь устанавливает флажки для тех свойств, которыми, по его мнению, обладает издание. Обработка анкеты будет состоять в том, что выбранные свойства будут отражены в поле ввода многострочного текста (рис. 5.1).

При щелчке мышью по флажку возникает событие `Click`, обработка которого состоит в вызове функции `set` с одним параметром, принимающим значение параметра `value` флажка. Для формирования строки результата служит глобальная переменная `s`; к имеющемуся значению добавляется значение параметра функции и помещается в текстовое поле. Если нажать на кнопку **Отмена**, то очистятся все поля формы. Однако следует позаботиться о том, чтобы значение переменной `s` изменилось на начальное. Значение параметра реакции на событие `Click` при щелчке по кнопке **Отмена** задается оператором присваивания, обеспечивающим начальные условия.

HTML-код представлен в листинге 5.1, а.

Листинг 5.1, а. Анкета читателя

```
<HTML>
  <HEAD>
    <TITLE>Анкета читателя</TITLE>
    <script>
    <!--
      var s="Вас привлекает: \r\n"
      function set(vch)
       { s=s+vch + "\r\n";  document.form1.area.value=s }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <CENTER>
    <H3 align="center">Анкета читателя</H3>
    <FORM name="form0">
      <H4>Введите название любимого журнала или газеты</H4>
      <input type="text" name="n1" size=45><br>
    </FORM>
    <FORM name="form1">
      <H4>Что Вас привлекает в издании?</H4>
      <TABLE border=3 align=center>
        <TR>
          <TD><img src="purpsm.gif" align="center"></TD>
```

```
          <TD><input type="checkbox" name="m1"
                     value="Стиль подачи материала"
                     onClick="set(form1.elements[0].value)">
                     Стиль подачи материала<br>
              <input type="checkbox" name="m2"
                     value="Достоверность информации"
                     onClick="set(form1.elements[1].value)">
                     Достоверность информации<br>
              <input type="checkbox" name="m3"
                     value="Дизайн и оформление"
                     onClick="set(form1.elements[2].value)">
                     Дизайн и оформление<br>
              <input type="checkbox" name="m4"
                     value="Качество информации"
                     onClick="set(form1.elements[3].value)">
                     Качество информации<br>
             <input type="checkbox" name="m5"
                    value="Репутация издания"
                    onClick="set(form1.elements[4].value)">
                    Репутация издания<br>
             <input type="checkbox" name="m6"
                    value="Регулярность издания"
                    onClick="set(form1.elements[5].value)">
                    Регулярность издания<br>
      </TD></TR></TABLE>
      <textarea name="area" cols=35 rows=7></textarea><br>
      <input type="reset" value="Отмена"
             onclick= "s='Вас привлекает: \r\n'">
    </FORM>
  </BODY>
</HTML>
```

Если флажок получает фокус, то происходит событие `Focus`, в качестве значения параметра обработки события, как и в предыдущем случае, может быть вызов функции `set`:

```
<input type="checkbox" name="m1" value="Стиль подачи материала"
       onfocus="set(form1.elements[0].value)">Стиль подачи материала<br>
```

И, наконец, потеря объектом фокуса вызовет событие `Blur`, обработка которого может быть произведена аналогичным способом:

```
<input type="checkbox" name="m1" value="Стиль подачи материала"
       onblur="set(form1.elements[0].value)">Стиль подачи материала<br>
```

Объект `checkbox` обладает свойствами `value`, `name`, `type`, которые соответствуют параметрам тега, описывающего флажок. Функция `set` получает в качестве параметра объект `checkbox` и формирует для выбранного флажка значения соответствующих ему свойств так, как показано на рис. 5.2.

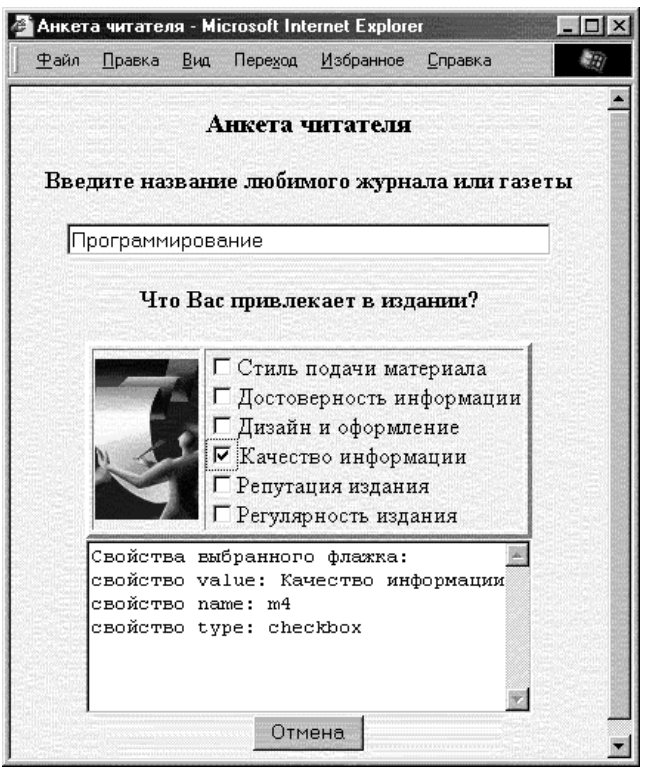


Рис. 5.2. Исследование свойств флажка

В представленном HTML-коде (листинг 5.1, б) обратите внимание на фактический параметр вызова функции `set` при обработке события `click`.

Листинг 5.1, б. Анкета читателя. Свойства флажка

```
<HTML>
  <HEAD>
```

```
  <TITLE>Анкета читателя </TITLE>
  <script>
  <!--
    var s="Свойства выбранного флажка: \r\n"
    function set(objch)
     {s+="свойство value: "+objch.value +"\n"+
          "свойство name: "+objch.name +"\n"+
          "свойство type: "+objch.type +"\n"
      document.form1.area.value=s}
  //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <CENTER>
  <H3 align="center">Анкета читателя</H3>
  <FORM name="form0">
    <H4>Введите название любимого журнала или газеты</H4>
    <input type="text" name="n1" size=45><br>
  </FORM>
  <FORM name="form1">
    <H4>Что Вас привлекает в издании?</H4>
    <TABLE border=3 align=center>
      <TR>
        <TD><img src="purpsm.gif" align="center"></TD>
        <TD><input type="checkbox" name="m1"
                   value="Стиль подачи материала"
                   onClick="set(form1.elements[0])">
                   Стиль подачи материала<br>
            <input type="checkbox" name="m2"
                   value="Достоверность информации"
                   onClick="set(form1.elements[1])">
                   Достоверность информации<br>
            <input type="checkbox" name="m3"
                   value="Дизайн и оформление"
                   onClick="set(form1.elements[2])">
                   Дизайн и оформление<br>
            <input type="checkbox" name="m4"
                   value="Качество информации"
                   onClick="set(form1.elements[3])">
```

```
                Качество информации<br>
            <input type="checkbox" name="m5"
                   value="Репутация издания"
                   onClick="set(form1.elements[4])">
                   Репутация издания<br>
            <input type="checkbox" name="m6"
                   value="Регулярность издания"
                   onClick="set(form1.elements[5])">
                   Регулярность издания<br>
      </TD></TR></TABLE>
      <textarea name="area" cols=35 rows=7></textarea><br>
      <input type="reset" value="Отмена"
             onclick="s='Свойства выбранного флажка: \r\n'">
    </FORM>
  </BODY>
</HTML>
```

В качестве параметра в функцию `set` можно передать номер флажка, а уже в самой функции определить свойства выбранного элемента. В этом случае функция `set` может быть описана следующим образом:

```
function set(i)
{ var d=document.forms[1].elements[i-1]
  s+="свойство value: "+d.value +"\n"+
      "свойство name: "+d.name +"\n"+
      "свойство type: "+d.type +"\n"
  document.forms[1].elements[6].value=s}
```

Обратите внимание, что в документе определены две формы. Одна содержит текстовое поле ввода названия журнала, а во второй форме расположены флажки, текстовое поле и кнопка сброса.

Разделы молодежного издания

Пусть читателю предлагается заполнить анкету, указав разделы, которые должны присутствовать в молодежном издании.

Названия разделов представлены в виде флажков. Пользователь устанавливает те флажки, разделы которых требуется включать в издание. Обработка анкет, как и ранее, состоит в том, что выбранные названия разделов помещаются в поле, расположенное в анкете справа, как изображено на рис. 5.3.

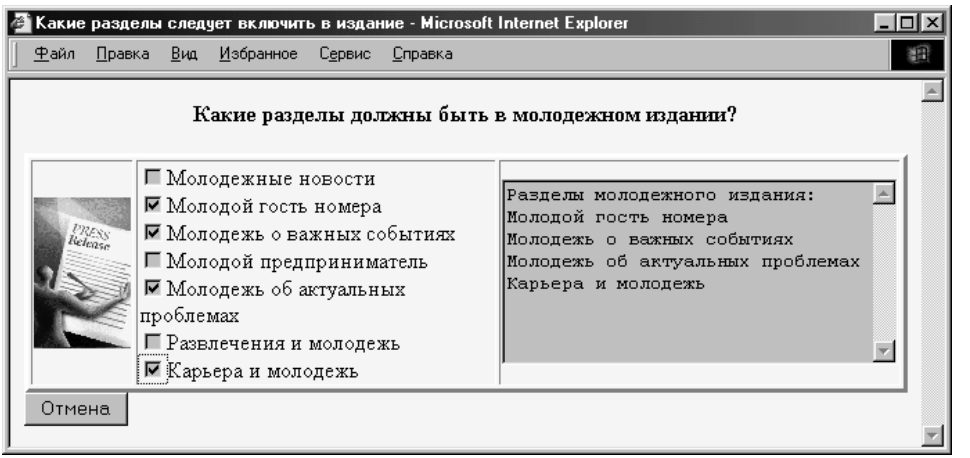


Рис. 5.3. Разделы молодежного издания

Объект `checkbox` кроме свойств `value`, `name` и `type`, которые соответствуют значениям параметра тега, описывающего флажок, обладает свойством `form`. Указанное свойство позволяет получить доступ к форме, на которой расположен флажок. Используя свойства формы, можно осуществить доступ к любому элементу формы. Доступ к восьмому элементу формы получается из функции `set`: `objch.form.elements[7].value`. От объекта `checkbox` с помощью свойства `form` перешли к форме, на которой расположен флажок, форма имеет свойство `elements`, позволяющее получить доступ к любому элементу формы, а далее выбирается свойство элемента, которое нас интересует. Приведем HTML-код (листинг 5.2), соответствующий документу на рис. 5.3.

Листинг 5.2. Разделы молодежного издания

```
<HTML>
  <HEAD>
    <TITLE>Какие разделы следует включить в издание</TITLE>
    <script>
    <!--
      var s="Разделы молодежного издания: \r\n"
      function set(objch)
       { s=s+objch.value +"\r\n";  objch.form.elements[7].value=s }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
```

```
    <H4 align=center>Какие разделы должны быть в молодежном издании?</H4>
    <TABLE border=3>
      <TR><TD><img src="smpress.gif"></TD>
      <TD><FOMRM name="form1">
            <input type="checkbox" name="m1" value="Молодежные новости"
                   onfocus="set(form1.m1)">Молодежные новости<br>
            <input type="checkbox" name="m2" value="Молодой гость номера"
                   onfocus="set(form1.m2)" >Молодой гость номера<br>
            <input type="checkbox" name="m3"
                   value="Молодежь о важных событиях"
                   onfocus="set(form1.m3)">Молодежь о важных событиях<br>
            <input type="checkbox" name="m4"
                   value="Молодой предприниматель"
                   onfocus="set(form1.m4)">Молодой предприниматель<br>
            <input type="checkbox" name="m5"
                   value="Молодежь об актуальных проблемах"
                   onfocus="set(form1.m5)">
                   Молодежь об актуальных проблемах<br>
            <input type="checkbox" name="m6"
                   value="Развлечения и молодежь"
                   onfocus="set(form1.m6)">Развлечения и молодежь<br>
            <input type="checkbox" name="m7" value="Карьера и молодежь"
                   onfocus="set(form1.m7)">Карьера и молодежь<br>
      </TD>
      <TD><textarea name="area" cols=33 rows=8></textarea></TD>
      </TR>
    </TABLE>
    <input type="reset" value="Отмена"
           onclick='s="Разделы молодежного издания: \r\n"'>
    </FORM>
  </BODY>
</HTML>
```

В рассмотренных примерах значения параметра `name` флажков были различны, поскольку каждый флажок существовал независимо от других. Флажки можно объединить в группу. Для этого следует всем флажкам присвоить одно и то же значение параметра `name`.

Использование флажков в анкете переводчика

В анкете требуется указать те языки, которыми владеет переводчик. Предположим, что за знание каждого языка назначается определенная сумма. Размер вознаграждения определяется после заполнения анкеты в зависимости от тех языков, которыми пользователь владеет. По результатам заполненной переводчиком анкеты напишите сценарий определения размера вознаграждения.

Для задания сведений о том, владеет ли пользователь определенным языком, удобно применять флажок. При щелчке мышью по кнопке **Вознаграждение** выполняется функция `grant()`. Требуется проанализировать состояние флажков. Свойство `checked` возвращает логическое значение, представляющее текущее значение отдельного флажка (`true` или `false`). Воспользуемся тем, что каждый объект `form` имеет свойство-массив `elements`, получим доступ к каждому флажку формы. Состояние первого флажка можно определить с помощью следующей конструкции:

```
(document.form1.elements[0]).checked
```

второго —

```
(document.form1.elements[1]).checked
```

и т. д. В переменной `k` накапливается сумма. Шаг увеличения этой переменной задается в качестве значения параметра `value`. После анализа всех флажков полученная сумма выводится в документ.

HTML-код представлен в листинге 5.3.

Листинг 5.3. Данные, представленные флажком. Анкета переводчика

```
<HTML>
  <HEAD>
    <TITLE>Данные, представленные флажком. Анкета переводчика</TITLE>
    <script language="JavaScript">
    <!-- //
      function grant()
       { var d= document
         var k=0;
         if ((d.form1.elements[0]).checked)
           k=k+Number(d.form1.elements[0].value)
         if ((d.form1.elements[1]).checked)
           k=k+Number(d.form1.elements[1].value)
         if ((d.form1.elements[2]).checked)
           k=k+Number(d.form1.elements[2].value)
```

```
        if ((d.form1.elements[3]).checked)
          k=k+Number(d.form1.elements[3].value)
        if ((d.form1.elements[4]).checked)
          k=k+Number(d.form1.elements[4].value)
        if ((d.form1.elements[5]).checked)
          k=k+Number(d.form1.elements[5].value)
        document.write("Вам полагается вознаграждение ",k, " у.е.")
      }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H3>Анкета для переводчиков</H3>
    Укажите те языки, которыми Вы владеете в совершенстве: <br>
    <FORM name="form1">
      <input type="checkbox" name="lan" value=100>русский<BR>
      <input type="checkbox" name="lan" value=200 >английский<BR>
      <input type="checkbox" name="lan" value=300>французский <BR>
      <input type="checkbox" name="lan" value=400>немецкий<BR>
      <input type="checkbox" name="lan" value=500>китайский<BR>
      <input type="checkbox" name="lan" value=600>японский<BR><HR>
      <input type="button" value=Вознаграждениие onClick="grant()"> <HR>
      <input type="reset"  value="Отменить">
    </FORM><hr>
  </BODY>
</HTML>
```

Параметр `id` применим в следующем примере при задании элементов флажка.

Использование параметра *id*

Определим размер вознаграждения по результатам заполнения анкеты. Обратите внимание на описание элементов форм и на анализ состояния флажков в функции `grant` (листинг 5.4).

Листинг 5.4. Переводчики. Использование параметра `id`

```
<HTML>
  <HEAD>
    <TITLE>Данные из формы типа checkbox. Переводчики</TITLE>
```

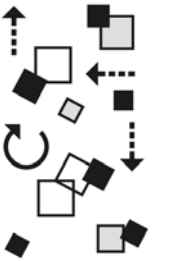
```
    <script language="JavaScript">
    <!-- //
      function  grant()
       { var d= document
         var k=0;
         if (d.all("1").checked) k+=Number(d.all("1").value)
         if (d.all("2").checked) k+=Number(d.all("2").value)
         if (d.all("3").checked) k+=Number(d.all("3").value)
         if (d.all("4").checked) k+=Number(d.all("4").value)
         if (d.all("5").checked) k+=Number(d.all("5").value)
         if (d.all("6").checked) k+=Number(d.all("6").value)
         form1.res.value="Вам полагается вознаграждение "+k+ " у.е."
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H3>Анкета для переводчиков </H3>
    Укажите те языки, которыми Вы владеете в совершенстве: <br>
    <FORM name="form1">
     <input type="checkbox" name="lan" value=100  id="1">русский<br>
     <input type="checkbox" name="lan" value=200  id="2">английский<br>
     <input type="checkbox" name="lan" value=300  id="3">французский<br>
     <input type="checkbox" name="lan" value=400  id="4">немецкий<br>
     <input type="checkbox" name="lan" value=500  id="5">китайский<br>
     <input type="checkbox" name="lan" value=600  id="6">японский<hr>
     <input type="button" value=Вознаграждениие onClick="grant()">
     <input type="text"  size=45 name="res"><HR>
     <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Упражнение

Напишите сценарий обработки анкеты слушателя курсов. Пользователь может выбрать курс, его продолжительность, язык, на котором он готов работать с преподавателем, и форму отчетности. В зависимости от этих параметров определяется стоимость отдельного курса и стоимость всего обучения. Анкета приведена на рис. 5.4.

Рис. 5.4. Анкета слушателя курсов

Глава 6



Списки

Если элементов много, то представление их с помощью флажков или переключателей увеличивает размер формы. В этом случае варианты выбора могут быть представлены в окне браузера более компактно с помощью тега `<select>`. Напомним, что тег имеет несколько параметров. Параметр `name` является обязательным. Для того чтобы установить число одновременно видимых элементов, следует задать параметр `size=n`. Когда `n` равно 1, то отображается ниспадающее меню или список выбора; при `n>1` выводится список с `n` одновременно видимыми значениями. Если параметр `size` не задан, то по умолчанию принимается значение равное 1. Указание параметра `multiple` означает, что из меню или списка можно выбрать несколько элементов. Элементы меню задаются внутри тега `<select>` с помощью тега `<option>`. Общий вид тега таков:

`<option selected value=`*строка*`>`

Параметр `selected` означает, что данный элемент списка считается выбранным по умолчанию. Параметр `value` содержит значение, которое передается, если данный элемент выбран из списка или меню.

Использование списка в задаче оформления заказа на витражи

Вернемся к задаче оформления заказа на витражи. В предыдущей версии форма витража задавалась с помощью переключателей. Теперь воспользуемся тегом `<select>`.

```
<BODY>
  <FORM name="form1">
    Выберите форму витража<br>
    <select name="list" size=1>
      <option value=0>прямоугольник
      <option value=1>квадрат
      <option value=2>треугольник
      <option value=3>круг
```

```
    </select>
    <input type="button" value="Проверить" onClick="testsel()">
    <input type="reset" value="Отменить">
  </FORM>
</BODY>
```

Выбрать форму витража можно с помощью ниспадающего меню. В приведенном выше коде форма содержит три элемента управления. Первый определяется тегом `<select>`, следующие два элемента представляют собой кнопки **Проверить** и **Отменить**.

Для того чтобы осуществить доступ к первому элементу формы (списку), следует воспользоваться одной из следующих конструкций

```
document.form1.list
document.form1.elements[0]
document.forms ["form1"].elements[0]
document.forms[0].elements[0]
```

Список обладает свойствами `length` и `name`. В рассматриваемом примере значение `document.form1.list.length` равно четырем, а значение `document.form1.list.name` равно `list`.

Свойства `selected` и `value` связаны с текущим выбором элемента из списка. Свойство `selected` возвращает значение `true`, если элемент выбран, и `false` — в противном случае. В рассматриваемом варианте кода если выбран первый элемент (прямоугольник), то значение `((document.forms["form1"].elements[0])[0].selected` станет равным `true`. Для анализа следующих трех элементов списка следует определить значения:

```
document.forms["form1"].elements[0])[1].selected
document.forms["form1"].elements[0])[2].selected
document.forms["form1"].elements[0])[3].selected
```

Свойство `value` возвращает значение параметра `value`, определенного в теге `<option>`. Значение `((d.forms["form1"].elements[0])[0]).value` равно нулю. Свойство `text` возвращает текст, записанный после тега `<option>` в объекте `select`. Так значением `((d.forms["form1"].elements[0])[0]).text` является строковый литерал `"прямоугольник"`.

Функция `testsel()` играет роль тестируемой программы и проверяет, правильно ли выбираются параметры. Полностью документ со сценарием будет выглядеть так, как представлено в листинге 6.1.

Листинг 6.1. Данные, представленные с помощью тега `select`

```
<HTML>
  <HEAD>
```

```
    <TITLE>Данные, представленные с помощью тега select</TITLE>
    <script ="JavaScript">
    <!-- //
      function testsel()
       { var d=document
         var r="Выбранная форма: "
         if (((d.forms["form1"].elements[0])[0]).selected)
           r=r+((d.forms["form1"].elements[0])[0]).text+"  "
         else
           if (((d.forms["form1"].elements[0])[1]).selected)
             r=r+((d.forms["form1"].elements[0])[1]).text+"  "
           else
             if (((d.forms["form1"].elements[0])[2]).selected)
               r=r+((d.forms["form1"].elements[0])[2]).text+"  "
             else
               if (((d.forms["form1"].elements[0])[3]).selected)
                 r=r+((d.forms["form1"].elements[0])[3]).text+"  "
         d.write(r)
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      Выберите форму витража<br>
      <select name="forma" size=1>
        <option value=0>прямоугольник
        <option value=1>квадрат
        <option value=2>треугольник
        <option value=3>круг
      </select>
      <input type="button" value="Проверить" onClick="testsel()">
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Использование списка в анекте переводчика

Напишем сценарий обработки анкеты переводчика. Сведения о тех языках, которыми тот владеет, требуется задать с помощью списка.

Так как переводчик может владеть несколькими языками, то в теге `<select>` необходимо указать параметр `multiple`, означающий, что из списка могут быть выбраны несколько элементов. Определим значение параметра `size` равным четырем. Тогда будет отображен список прокрутки с четырьмя одновременно видимыми значениями и тремя выбранными, как на рис. 6.1.

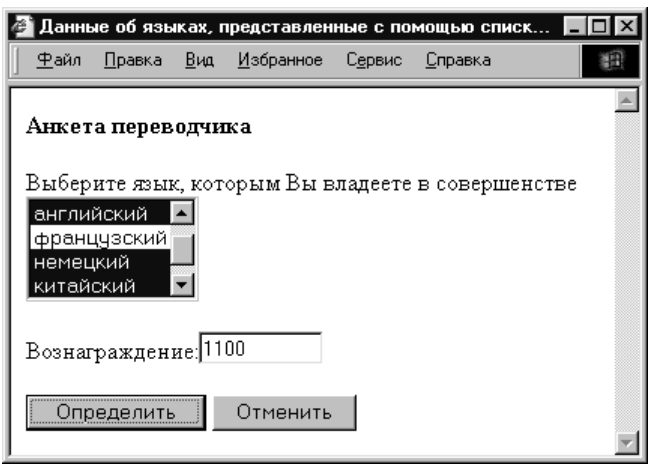


Рис. 6.1. Анкета переводчика

Приведем описание тела HTML-кода документа (листинг 6.2, а).

Листинг 6.2, а. Анкета переводчика. Представление списком

```
<BODY>
  <FORM name="form1">
    <H4>Анкета переводчика</H4>
    Выберите язык, которым Вы владеете в совершенстве<br>
    <select name="forma" size=4 multiple>
      <option  value="русский">русский
      <option  value="английский">английский
      <option  value="французский">французский
```

```
      <option  value="немецкий">немецкий
      <option  value="китайский">китайский
      <option  value="японский">японский
    </select><P>
    Вознаграждение:<input type="text" name="res" size=10><P>
    <input type="button" value="Определить"
           onClick="form1.res.value=testsel()">
    <input type="reset" value="Отменить">
  </FORM>
</BODY>
```

В форме содержатся несколько элементов. Нас будет интересовать первый элемент, а в первом элементе те значения, которые выбраны. Как и в предыдущем примере, определить, какие элементы выбраны, можно, если проанализировать следующие значения:

```
((document.forms["form1"].elements[0])[0]).selected
((document.forms["form1"].elements[0])[1]).selected
((document.forms["form1"].elements[0])[2]).selected
((document.forms["form1"].elements[0])[3]).selected
((document.forms["form1"].elements[0])[4]).selected
((document.forms["form1"].elements[0])[5]).selected
```

За знание каждого языка назначается определенная сумма. После анализа всех выбранных значений определяется вознаграждение. Для определения суммы вознаграждения следует щелкнуть по кнопке **Определить**. Реакция на это событие — запись в поле формы вычисленного значения (листинг 6.2, б).

Листинг 6.2, б. Данные об языках, представленные с помощью списка

```
<HTML>
  <HEAD>
    <TITLE>Данные об языках, представленные с помощью списка</TITLE>
    <script language="JavaScript">
    <!-- //
      function testsel()
       { var d=document
         var s=0
         if (((d.forms["form1"].elements[0])[0]).selected) s+=100
         if (((d.forms["form1"].elements[0])[1]).selected) s+=200
```

```
        if (((d.forms["form1"].elements[0])[2]).selected) s+=300
        if (((d.forms["form1"].elements[0])[3]).selected) s+=400
        if (((d.forms["form1"].elements[0])[4]).selected) s+=500
        if (((d.forms["form1"].elements[0])[5]).selected) s+=600
        return s
      }
  //-->
  </script>
</HEAD>
<BODY>
  <FORM name="form1">
  <H4>Анкета переводчика</H4>
    Выберите язык, которым Вы владеете в совершенстве<br>
    <select name="forma" size=4 MULTIPLE>
      <option value="русский">русский
      <option value="английский">английский
      <option value="французский">французский
      <option value="немецкий">немецкий
      <option value="китайский">китайский
      <option value="японский">японский
    </select><P>
    Вознаграждение: <input type="text" name="res" size=10><P>
    <input type="button" value="Определить"
           onClick="form1.res.value=testsel()">
    <input type="reset" value="Отменить">
  </FORM>
</BODY>
</HTML>
```

Обработка анкеты переводчика

Напишем сценарий обработки анкеты переводчика. Сведения о тех языках, которыми владеет переводчик, требуется задать с помощью списка. Выбранные языки следует помещать в поле ввода многострочного текста, как на рис. 6.2.

Напомним, что событие `Change` происходит в тот момент, когда значение элемента формы `text`, `select` или `textarea` изменилось, и элемент потерял фокус. Будем обрабатывать анкету переводчика следующим образом. Пара-

метр обработки события поместим в тег <select>. Как только выбран конкретный язык, т. е. произошло событие Change, выполняется функция gr:

```
function gr(obj,m)
 { var r=100*(Number(((obj.elements[0])[m]).value)+1)
   s+=((obj.elements[0])[m]).text+"\r\n"
   obj.restext.value=s
   sum+=r
   obj.res.value=r
 }
```

Рис. 6.2. Выбор с помощью списка

Первый параметр — имя формы, второй — значение параметра value выбранного элемента. Второй оператор обеспечивает формирование строки всех выбранных пользователем языков. Третий оператор помещает вычисленное значение в текстовое поле. В результате выполнения четвертого оператора присваивания в переменной sum формируется значение, которое затем при нажатии кнопки **Сумма** будет выведено в текстовое поле. Последний оператор помещает значение для выбранного языка в соответствующее поле формы. Полностью документ со сценарием и формами может быть описан так, ка указано в листинге 6.3.

Листинг 6.3. Реакция на событие `Change` в теге `<select>`

```
<HTML>
  <HEAD>
    <TITLE>Реакция на событие Change в теге select</TITLE>
    <script language="JavaScript">
    <!-- //
      var s=""
      var sum=0
      function gr(obj,m)
       { var r=100*(Number(((obj.elements[0])[m]).value)+1)
         s+=((obj.elements[0])[m]).text+"\r\n"
         obj.restext.value=s
         sum+=r
         obj.res.value=r
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      <H3>Анкета переводчика</H3>
      <TABLE border=3  bgcolor=silver>
        <TR><TH>Выбранный язык</TH><TH>Результат</TH></TR>
        <TR>
          <TD valign=top>
            <select name="data" size=6
                    onChange="gr(form1,form1.data.value)">
              <option value=0>русский
              <option value=1>английский
              <option value=2>французский
              <option value=3>немецкий
              <option value=4>китайский
              <option value=5>японский
             </select><P>
             <input type="text" name="res" size=15>
          </TD>
```

```
          <TD><TEXTAREA name="restext" cols=15 rows=6>
              </TEXTAREA><P>
              <input type="button" value=Сумма
                     onClick="form1.resgr.value=sum">
              <input type="text" name="resgr" size=10>
      </TD></TR></TABLE><p>
      <input type="reset" value="Отменить" onClick="sum=0; s=''">
    </FORM>
  </BODY>
</HTML>
```

Рассмотрим случай, когда значение параметра обработки события — функция с одним параметром, который является именем тега `<select>`.

```
<select name="data" size=6 onChange="gr(form1.data)">
  <option value=0>русский
  <option value=1>английский
  <option value=2>французский
  <option value=3>немецкий
  <option value=4>китайский
  <option value=5>японский
</select>
```

Для того чтобы получить доступ к значению в сценарии, выполняется конструкция `obj.value`, а свойство списка `form` позволяет получить доступ к объектам формы. Форма имеет несколько элементов, к которым доступ получен различными способами в следующем сценарии:

```
function gr(obj)
 { var r=(Number(((obj.form.elements[0])[obj.value]).value)+1)*100
  s+=((obj.form.elements[0])[obj.value]).text+"\r\n"
  obj.form.restext.value=s
  sum+=r
  obj.form.res.value=r
 }
```

Анкета читателя

Пусть при заполнении анкеты читатель выбирает в порядке важности для него некоторые заданные в анкете характеристики книг. В результате выбора должен быть сформирован список так, как показано на рис. 6.3.

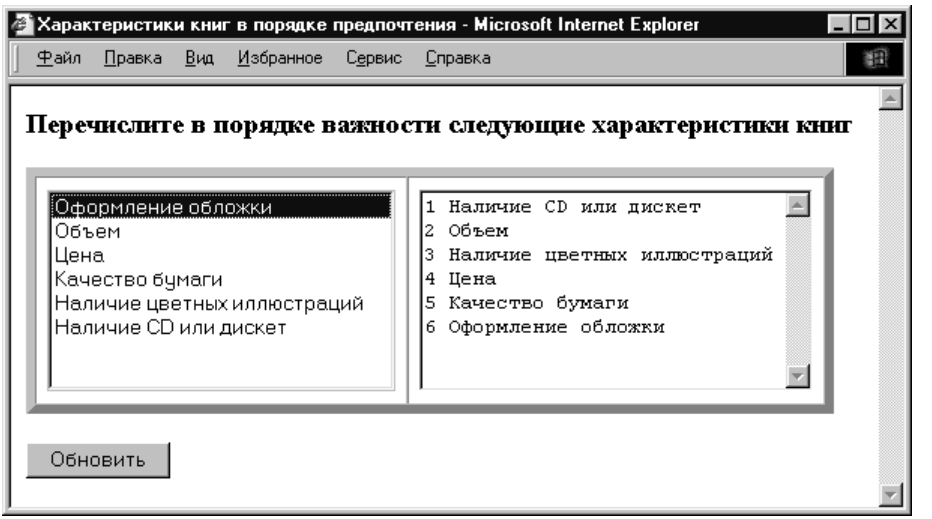


Рис. 6.3. Обработка анкеты читателя

При выборе очередной характеристики в теге `<select>` возникает событие `Change`, обработка которого состоит в помещении выбранной характеристики в качестве элемента списка, формируемого в многострочном текстовом поле. При обновлении полей формы глобальным переменным, используемым в сценарии, присваиваются начальные значения. HTML-код представлен в листинге 6.4.

Листинг 6.4. Характеристики книг в порядке предпочтения

```
<HTML>
  <HEAD>
    <TITLE>Характеристики книг в порядке предпочтения</TITLE>
    <script>
      var k=1
      var s=""
      function mov(n)
       { s=s+k+' '+form1.data[n].text+"\r\n"
         k=k+1
         form1.res.value=s
       }
      function ref()
       { k=1;    s="" }
    </script>
```

```
  </HEAD>
  <BODY>
    <H3>Перечислите в порядке важности следующие
        характеристики книг</H3>
    <FORM name="form1">
      <TABLE border="6" cellpadding="7" cellspacing="0" width="100">
        <TR>
          <TD>
            <select name="data" size="8"
                    onChange="mov(form1.data.value)">
              <option value=0>Оформление обложки
              <option value=1>Объем
              <option value=2>Цена
              <option value=3>Качество бумаги
              <option value=4>Наличие цветных иллюстраций
              <option value=5>Наличие CD или дискет
            </select>
          </TD>
          <TD valign="bottom" width="95%">
            <textarea name="res" rows="8" cols="30"></textarea>
          </TD>
        </TR>
      </TABLE><p>
      <input type="reset" value="Обновить" onClick=ref()>
    </FORM>
  </BODY>
</HTML>
```

Обработка анкеты читателя

Необходимо написать сценарий обработки анкеты читателя. Выбранные характеристики должны формироваться в виде списка.

После выбора очередной характеристики книги следует щелкнуть по кнопке **Добавить**, и характеристика с номером предпочтения появится в правом столбце. В результате выполнения функции `test` проверяется, какой элемент выбран, и указанная характеристика помещается в конец формируемого списка. Количество элементов в формируемом списке должно быть не более, чем в исходном. Если делается попытка поместить большее число элементов, то выводится сообщение об ошибке (листинг 6.5).

Листинг 6.5. Данные, представленные списком и помещаемые в список

```
<HTML>
  <HEAD>
    <TITLE>Данные, представленные списком</TITLE>
    <script language="JavaScript">
    <!-- //
      var n=0
      function  test(obj)
       { if (n<=(obj.elements[2]).length-1)
         { if (((obj.elements[0])[0]).selected)
           {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[0]).text+" "; n= n+1}
           if (((obj.elements[0])[1]).selected)
             {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[1]).text+" "; n= n+1}
           if (((obj.elements[0])[2]).selected)
              {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[2]).text+" "; n= n+1}
           if (((obj.elements[0])[3]).selected)
              {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[3]).text+" "; n= n+1}
           if (((obj.elements[0])[4]).selected)
              {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[4]).text+" "; n= n+1}
           if (((obj.elements[0])[5]).selected)
              {((obj.elements[2])[n]).text=n+1+
                 " "+((obj.elements[0])[5]).text+" "; n= n+1}
         }
         else alert ("Ваш выбор содержит ошибку")
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <FORM name="form1">
      <H3>Анкета читателя</H3>
      <P>Перечислите в порядке важности следующие
         шесть характеристик книг</P>
```

```
      <select name="forma" size=7 multiple>
        <option value="обложка">Оформление обложки
        <option value="объем">Объем книги
        <option value="цена">Цена книги
        <option value="качество бумаги">Качество бумаги
        <option value="иллюстраций">Наличие цветных иллюстраций
        <option value="диски">Наличие CD или дискет
      </select>
      <input type="button" value=Добавить onClick="test(form1)">
      <select name="formres" size=7 multiple>
        <option value="n1">
        <option value="n2">
        <option value="n3">
        <option value="n4">
        <option value="n5">
        <option value="n6">
      </select>
    </FORM>
  </BODY>
</HTML>
```

Недостаток сценария заключается в том, что в результирующем списке могут оказаться одинаковые элементы.

Выбор изображения из списка

Напишем сценарий, во время работы которого при выборе из заданного списка названия рисунка в документе появляется соответствующее изображение (рис. 6.4).

Названия рисунков задаются с помощью тега `<select>`. При выборе названия возникает событие `Change`, обработка которого состоит в том, что в документе появляется соответствующий выбранному названию рисунок. Связь между названием рисунка и изображением устанавливается параметром `value`. При выполнении оператора `switch` в функции, обрабатывающей событие, анализируется выбранное значение и загружается соответствующий рисунок. Для того чтобы отобразить на странице другой рисунок, следует изменить параметр `src` объекта `image`. Документ, реализующий сценарий, представлен в листинге 6.6.

Рис. 6.4. Выбор изображения из списка

Листинг 6.6. Выбор изображения из списка

```
<HTML>
  <HEAD>
    <TITLE>Выбор изображения из списка</TITLE>
    <script>
    <!--
      function ref(obj)
       { obj.pict.value='0'
         ch_pict(0)
         obj.res.value=''
         obj.mypict.src="picture0.gif"
       }
      function ch_pict(num)
       { var d=document
         switch (num)
         { case "0": d.mypict.src="picture0.gif"; break;
           case "1": d.mypict.src="picture1.gif"; break;
           case "2": d.mypict.src="picture2.gif"; break;
           case "3": d.mypict.src="picture3.gif"; break;
           case "4": d.mypict.src="picture4.gif"; break;
           case "5": d.mypict.src="picture5.gif"; break;
```

```
          case "6": d.mypict.src="picture6.gif"; break;
        }
        form1.res.value=num*100
      }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#eaf5ef">
    <H3>Выбор рисунка</H3>
    <FORM name="form1">
      <TABLE border=1>
        <TR>
          <TD>
            <select name="pict" size="7"
                    onChange="ch_pict(form1.pict.value)">
              <option value="0">Нет
              <option value="1">Дом
              <option value="2">Книги
              <option value="3">Письмо
              <option value="4">Краски
              <option value="5">Вопрос
              <option value="6">Дерево
            </select>
          </TD>
          <TD><IMG src="picture0.gif" name="mypict" width=100></TD></TR>
      </TABLE>
      <br>
      Стоимость: <input type="text" name="res" size=12><P>
      <input type="button" value="Обновить" onClick="ref(form1)">
   </FORM>
  </BODY>
</HTML>
```

В приведенном примере считается, что файлы с изображением хранятся в той же папке, что и создаваемый HTML-документ. Если это не так, то в правой части операторов присваивания оператора `switch` следует указать полное имя файла с изображением. Тот факт, что выбранное название рисунка и имя графического файла с изображением связаны значением пара-

метра value тега <option>, позволяет сократить текст программы и описать функцию ch_pict следующим образом:

```
function ch_pict(num)
 { var d=document
   d.mypict.src="picture"+num+".gif"
   form1.res.value=num*100
 }
```

Имя файла изображения формируется в виде строки; значение num определяет по названию имя файла с изображением.

В документе лишь одно изображение, поэтому для доступа к объекту image на странице можно воспользоваться конструкцией document.images[0]. Функция ch_pict в этом случае может быть описана следующим образом.

```
function ch_pict(num)
 { var d=document
   d.images[0].src="picture"+num+".gif"
   form1.res.value=num*100
 }
```

Изменение свойств горизонтальной линии

Напишем сценарий, при выполнении которого горизонтальная линия представлена в документе с заданными параметрами. Предусмотрите возможность изменения цвета линии. После задания параметров выравнивания, длины, толщины и цвета линии документ примет вид, как на рис. 6.5.

Определим в виде списка название цветов для горизонтальной линии, которые может задать пользователь. При выборе цвета реакцией на событие Change будет вызов функции gr(form1,form1.col.value) с двумя параметрами. Первый параметр необходим, чтобы получить доступ к элементам документа, а второй указывает значение выбранного элемента в списке цветов. По последнему параметру определяется название цвета. Функцию gr можно упростить, если в качестве значения параметра value тега <option> задать название цвета.

Доступ к свойствам, определяющим горизонтальную линию, осуществляется с помощью параметра id.

HTML-код, содержащий требуемый сценарий, представлен в листинге 6.7.

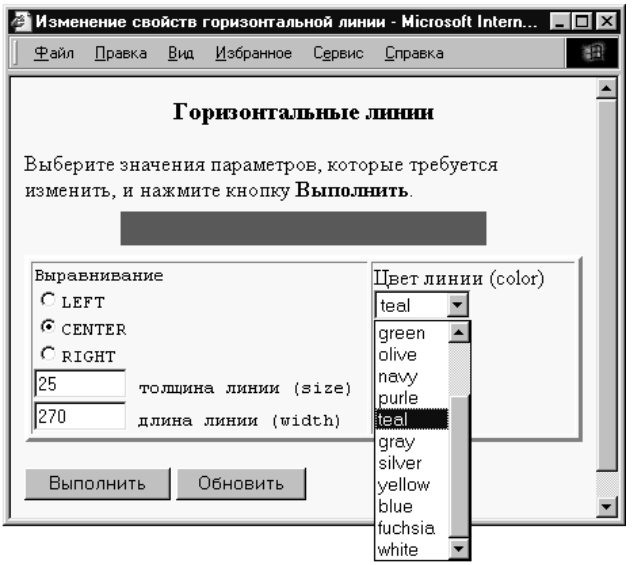


Рис. 6.5. Изменение свойств горизонтальгой линии

Листинг 6.7. Изменение свойств горизонтальной линии

```
<HTML>
  <HEAD>
    <TITLE>Изменение свойств горизонтальной линии</TITLE>
    <script>
    <!--
      function  gr(obj,m)
       { document.all("l").color=((obj.elements[5])[m]).text }
      function set(obj)
       { var s=""
         if(obj.elements[0].checked) s=obj.elements[0].value
         else
           if(obj.elements[1].checked) s=obj.elements[1].value
         else
           if(obj.elements[2].checked) s=obj.elements[2].value
             document.all("l").align=s
         if (obj.wid.value != "")  document.all("l").width=obj.wid.value
         if (obj.siz.value != "")  document.all("l").size=obj.siz.value
       }
      function rset(obj)
```

```
       { document.all("l").width=200
         document.all("l").size=20
         document.all("l").color="red"
         document.all("l").align="left"
         obj.wid.value=200
         obj.siz.value=20
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H3 align=center>Горизонтальные линии</H3>
    Выберите значения параметров, которые требуется изменить,
    и нажмите кнопку <B>Выполнить</B>.
    <HR id="l" name="lin" size=20 width=200 align= left Color=red>
    <TABLE border=3>
      <TR>
        <TD>
         <FORM name="form1">
           <PRE>
Выравнивание
<input type="radio" name="m" value="LEFT" checked>LEFT
<input type="radio" name="m" value="CENTER">CENTER
<input type="radio" name="m" value="RIGHT">RIGHT
<input type="text" size=8 name= siz value=20> толщина линии (size)
<input type="text" size=8 name= wid value=200> длина линии (width)
</TD>
<TD valign="top">
Цвет линии (color)   <select name= col size=1
               onChange="gr(form1,form1.col.value)">
   <option value=0 >red
   <option value=1>green
   <option value=2>black
   <option value=3>maroon
   <option value=4>green
   <option value=5>olive
   <option value=6>navy
   <option value=7>purle
   <option value=8>teal
   <option value=9>gray
```

```
   <option value=10>silver
   <option value=11>yellow
   <option value=12>blue
   <option value=13>fuchsia
   <option value=14>white
 </select>
</TD></TR></TABLE><P>
<input type="button" value="Выполнить"  onclick="set(form1)">
<input type="reset" value="Обновить"  onclick="rset(form1)">
       </PRE>
    </FORM>
  </BODY>
</HTML>
```

Анкета "Преподаватель и студент"

Напишем сценарий обработки анкеты "Преподаватель и студент". Фамилии преподавателей задаются с помощью списка. Студент выбирает фамилию преподавателя и ставит оценки по указанным критериям.

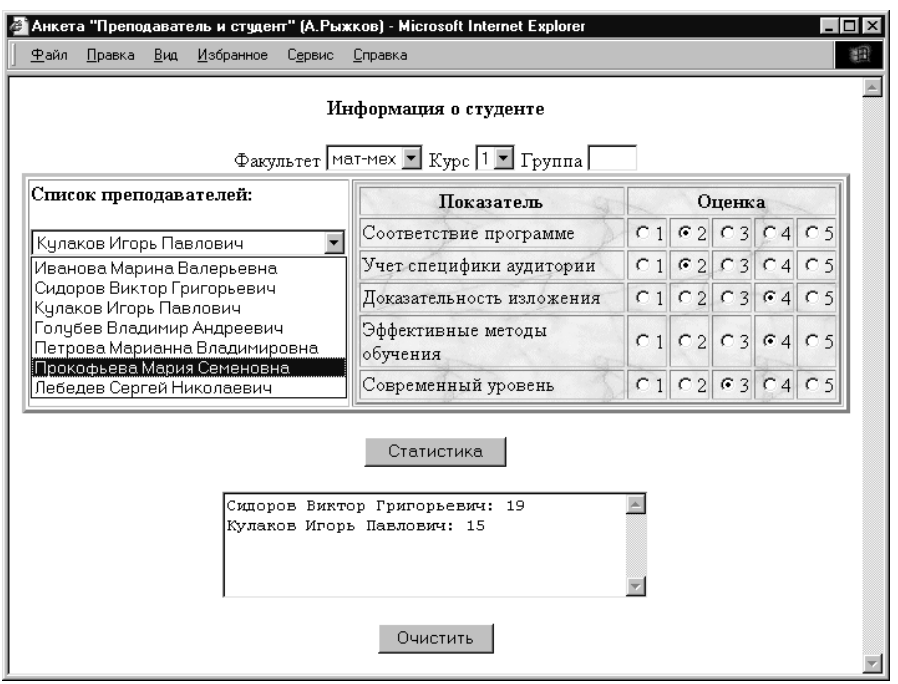


Рис. 6.6. Анкета "Преподаватель и студент"

После нажатия кнопки **Статистика** фамилия преподавателя и вычисленная сумма баллов появляются в области результата. После работы с анкетой ее вид может быть таким, как на рис. 6.6.

HTML-код документа обработки анкеты имеет вид, представленный в листинге 6.8.

Листинг 6.8. Анкета "Преподаватель и студент"

```
<HTML>
  <HEAD>
    <TITLE> Анкета "Преподаватель и студент" (А.Рыжков)</TITLE>
    <script language="JavaScript">
    <!--
      var p=""
      var s=""
      function  gr(obj,m)
       { p=((obj.elements[3])[m]).text }
      function statist(obj)
       { var res=0
         if(obj[0].checked) {res=res+1}
         if(obj[1].checked) {res=res+2}
         if(obj[2].checked) {res=res+3}
         if(obj[3].checked) {res=res+4}
         if(obj[4].checked) {res=res+5}
         return res
       }
      function stat(obj)
       { s=s+p+": "+Number(statist(obj.par1)+statist(obj.par2)+
           statist(obj.par3)+statist(obj.par4)+ statist(obj.par5))+"\r\n"
         obj.result.value=s
       }
    //-->
    </script>
  </HEAD>
  <BODY background="fon1.jpg">
    <CENTER>
    <H4 align="center"> Информация о студенте </H4>
    <FORM name="form1">
      Факультет <select name="faculty" size=1 maxlength=20>
```

```
            <option value="mm"> мат-мех
            <option value="ph"> физфак
            <option value="ch"> химфак
          </select>
Курс <select name="k" size=1 maxlength=1>
       <option value=1> 1
       <option value=2> 2
       <option value=3> 3
       <option value=4> 4
       <option value=5> 5
     </select>
 Группа <input type="text" name="group" size=4>
 <TABLE border=3>
   <TR>
     <TD valign="top">
       <H4>Список преподавателей:</H4>
       <select name="teacher" size=1 col=50
               onClick="gr(form1,form1.teacher.value)">
         <option value=0> Иванова Марина Валерьевна
         <option value=1> Сидоров Виктор Григорьевич
         <option value=2> Кулаков Игорь Павлович
         <option value=3> Голубев Владимир Андреевич
         <option value=4> Петрова Марианна Владимировна
         <option value=5> Прокофьева Мария Семеновна
         <option value=6> Лебедев Сергей Николаевич
       </select>
     </TD>
     <TD>
       <TABLE border=2 align="center" background="fon3.gif">
         <TR><TH align="center"> Показатель </TH>
             <TH colspan=5 align="center"> Оценка </TH></TR>
         <TR><TD>Соответствие программе
             <TD><input type="radio" name="par1" value=1>1
             <TD><input type="radio" name="par1" value=2>2
             <TD><input type="radio" name="par1" value=3>3
             <TD><input type="radio" name="par1" value=4>4
             <TD><input type="radio" name="par1" value=5>5
         </TR>
```

```
            <TR><TD>Учет специфики аудитории
                <TD><input type="radio" name="par2" value=1>1
                <TD><input type="radio" name="par2" value=2>2
                <TD><input type="radio" name="par2" value=3>3
                <TD><input type="radio" name="par2" value=4>4
                <TD><input type="radio" name="par2" value=5>5
                </TD></TR>
            <TR><TD>Доказательность изложения
                <TD><input type="radio" value=1 name="par3">1
                <TD><input type="radio" value=2 name="par3">2
                <TD><input type="radio" value=3 name="par3">3
                <TD><input type="radio" value=4 name="par3">4
                <TD><input type="radio" value=5 name="par3">5
                </TD></TR>
            <TR><TD>Эффективные методы обучения
                <TD><input type="radio" value=1 name="par4">1
                <TD><input type="radio" value=2 name="par4">2
                <TD><input type="radio" value=3 name="par4">3
                <TD><input type="radio" value=4 name="par4">4
                <TD><input type="radio" value=5 name="par4">5
                </TD></TR>
            <TR><TD>Современный уровень
                <TD><input type="radio" value=1 name="par5">1
                <TD><input type="radio" value=2 name="par5">2
                <TD><input type="radio" value=3 name="par5">3
                <TD><input type="radio" value=4 name="par5">4
                <TD><input type="radio" value=5 name="par5">5
                </TD></TR>
          </TABLE>
        </TD>
      </TR>
    </TABLE><P>
    <input type="button" value="Статистика" onClick="stat(form1)"><P>
    <textarea Cols=40 rows=5 name="result"></textarea><P>
    <input type="reset" value="Очистить" onClick="p=''; s=''">
  </FORM>
 </BODY>
</HTML>
```

Тест "Города и памятники"

Напишем сценарий обработки теста "Города и памятники". Названия городов и памятников задаются с помощью списков. Пользователь выбирает в левом перечне название города, а в правом — памятник, расположенный в этом городе. После нажатия кнопки **Результат** в текстовое поле выводится количество правильных ответов (рис. 6.7).

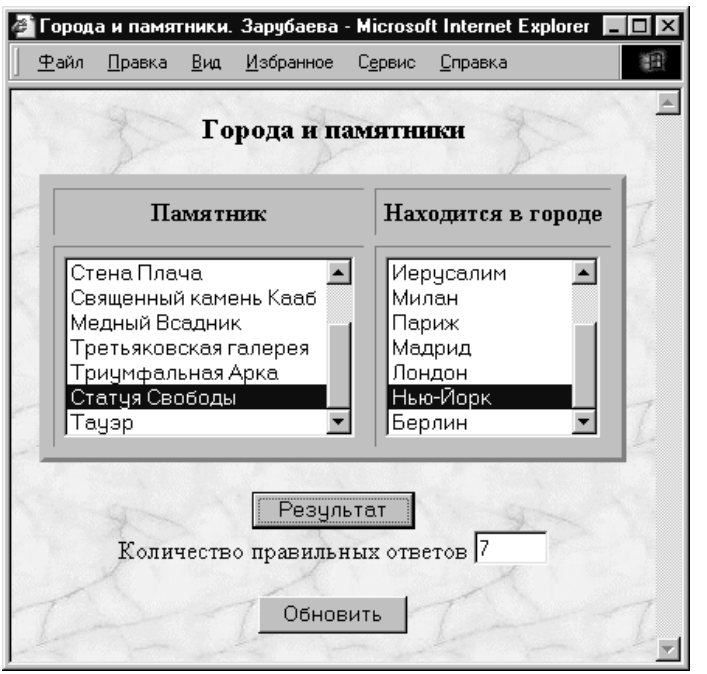


Рис. 6.7. Тест "Города и памятники"

В сценарии используются три глобальные переменные. Переменная `q` хранит последнее выбранное значение в левом столбце; переменная `a` — выбранное значение правого столбца; значение переменной `sum` содержит число правильных ответов. В двух списках для правильной пары "вопрос/ответ" совпадают соответствующие значения параметра `value`. Эти значения проверяются после выбора элемента списка правого столбца. Результат тестирования можно посмотреть, если нажать кнопку **Результат**.

Сценарий, реализующий простую обработку теста, представлен в листинге 6.9.

Листинг 6.9. Простая тестирующая программа

```
<HTML>
  <HEAD>
    <TITLE>Города и памятники. (Зарубаева)</TITLE>
    <script>
    <!--
      var sum=0
      var q
      var a
      function eq()
       { q=test.question.value
         a=test.answer.value;
         if (a==q) sum +=1
       }
      function result()
       { document.test.res.value=sum}
    //-->
    </script>
    <BODY background="fon3.gif">
      <h3 align=center>Города и памятники</h3>
      <FORM name="test">
      <TABLE border=3  align=center cellpadding=5
             cellspacing=6 bgcolor= silver>
        <TR><TH>Памятник</TH><TH>Находится в городе</TH>
        <TR><TD>
            <select size =7 name="question"
                    onChange="q=test.question.value">
              <option value="mad">Музей Прадо<br>
              <option value="ber" >Рейхстаг<br>
              <option value="mil">Оперный театр Ла Скала<br>
              <option value="ier">Стена Плача<br>
              <option value="mek">Священный камень Кааб<br>
              <option value="spb">Медный Всадник<br>
              <option value="mos">Третьяковская галерея<br>
              <option value="par">Триумфальная Арка<br>
              <option value="new">Статуя Свободы<br>
              <option value="lon">Тауэр<br>
            </select>
```

```
        </TD>
        <TD>
          <select size=7 name="answer" onChange="eq()">
            <option value="spb">Санкт-Петербург<br>
            <option value="mos">Москва<br>
            <option value="mek">Мекка<br>
            <option value="ier">Иерусалим<br>
            <option value="mil">Милан<br>
            <option value="par">Париж<br>
            <option value="mad">Мадрид<br>
            <option value="lon">Лондон<br>
            <option value="new">Нью-Йорк<br>
            <option value="ber">Берлин<br>
          </select>
        </TD></TR>
      </TABLE>
      <CENTER><P>
      <input type="button" value= "Результат" onclick="result()"><br>
      Количество правильных ответов
      <input type="text" name="res" size="5"><P>
      <input type="reset" value= "Обновить" onclick="sum=0">
    </FORM>
  </BODY>
</HTML>
```

Цветовое оформление таблицы и ячеек

Напишем сценарий, который позволяет выбирать цвет фона для всей таблицы и ячеек и продемонстрировать, как будет выглядеть документ при разном цветовом оформлении (рис. 6.8).

Предположим, что для размещения картин в галерее требуется задать цвет фона стены и цвет рамок картин. В рассматриваемом примере с помощью выбора цвета фона таблицы и цвета фона ее ячеек задается та цветовая гамма, которая более удачно соответствует изображениям или вкусам пользователя. Выбор цвета фона таблицы и ячеек выполняется с помощью списка. Не предусматривается возможность задания для каждой из ячеек отдельного цвета. HTML-код документа, содержащего функции, выполняющие необходимые преобразования, имеет вид, представленный в листинге 6.10.

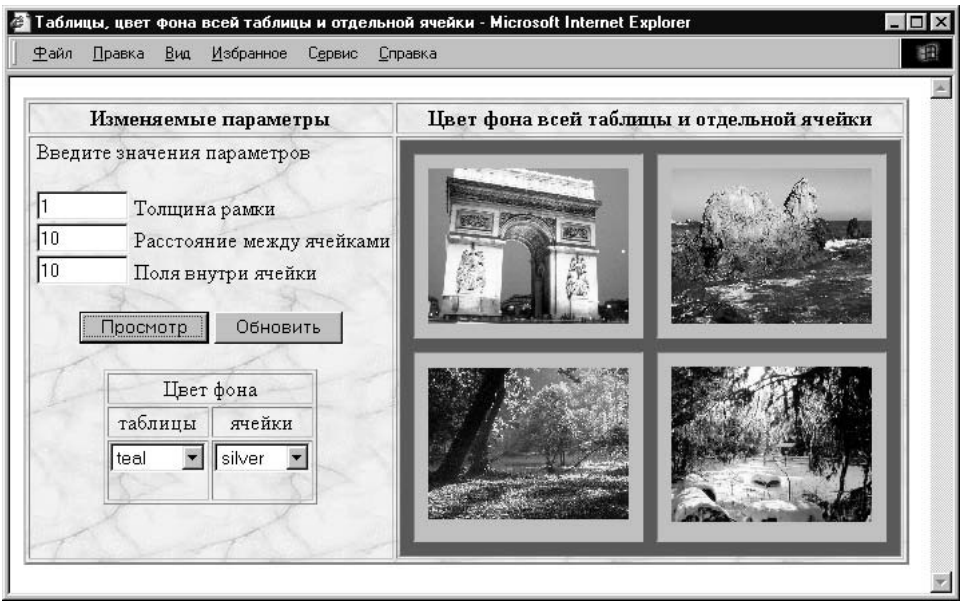


Рис. 6.8. Цветовое оформление таблиц и ячеек

Листинг 6.10. Цветовое оформление таблицы и ячеек

```
<HTML>
  <HEAD>
    <TITLE>Таблицы, цвет фона всей таблицы и отдельной ячейки</TITLE>
    <script>
    <!--
      function rset(obj)
       { document.all("itab").bgColor="silver"
         document.all("itab1").bgColor="gray"
         document.all("itab2").bgColor="gray"
         document.all("itab3").bgColor="gray"
         document.all("itab4").bgColor= "gray"
         document.all("itab").border=1
         document.all("itab").cellSpacing=10
         document.all("itab").cellPadding=20
         obj.bor.value=1
         obj.cells.value=10
         obj.cellp.value=20
       }
```

```
    function gr(obj,m)
     { document.all("itab").bgColor=((obj.elements[5])[m]).text }
    function  grcol(obj,m)
     { document.all("itab1").bgColor=((obj.elements[5])[m]).text
       document.all("itab2").bgColor=((obj.elements[5])[m]).text
       document.all("itab3").bgColor=((obj.elements[5])[m]).text
       document.all("itab4").bgColor=((obj.elements[5])[m]).text
     }
    function set(obj)
     { document.all("itab").border=Number(obj.bor.value)
       if (obj.cells.value != "")
         document.all("itab").cellSpacing=Number(obj.cells.value)
       if (obj.cellp.value != "")
         document.all("itab").cellPadding=Number(obj.cellp.value)
     }
  //-->
  </script>
</HEAD>
<BODY id="doc">
  <TABLE border=2 background="fon3.gif">
    <TR><TD align=center><B>Изменяемые параметры</B></TD>
        <TD align=center><B>Цвет фона всей таблицы
                           и отдельной ячейки</B></TD>
    </TR>
    <TR>
      <TD valign=top>
       <Введите значения параметров
      <FORM name="form1">
         <input type="text" size=8 name=bor value=1>
          Толщина рамки<br>
         <input type="text" size=8 name=cells value=10>
          Расстояние между ячейками<br>
         <input type="text" size=8 name=cellp value=20>
          Поля внутри ячейки<br><br>
        <CENTER>
          <input type="button" value="Просмотр"  onclick="set(form1)">
          <input type="reset"  value="Обновить"
                 onclick="rset(form1)"><br><br>
        </CENTER>
```

```
<TABLE border=1 align=center background="fon1.jpg">
<TR><TD colspan=2 align=center>Цвет фона</TD></TR>
<TR><TD align=center>таблицы</TD>
    <TD align=center>ячейки</TD></TR>
<TR>
  <TD align=center valign=top>
  <select name= col size=1
     onChange="gr(form1,form1.col.value)">
    <option value=0 >red
    <option value=1>green
    <option value=2>black
    <option value=3>maroon
    <option value=4>green
    <option value=5>olive
    <option value=6>navy
    <option value=7>purle
    <option value=8>teal
    <option value=9>gray
    <option value=10 selected>silver
    <option value=11>yellow
    <option value=12>blue
    <option value=13>fuchsia
    <option value=14>white
  </select>
  </TD>
  <TD align=center valign=top>
  <select name= colcol size=1
     onChange="grcol(form1,form1.colcol.value)">
    <option value=0>red
    <option value=1>green
    <option value=2>black
    <option value=3>maroon
    <option value=4 selected>green
    <option value=5>olive
    <option value=6>navy
    <option value=7>purle
    <option value=8>teal
    <option value=9>gray
    <option value=10>silver
```

```
            <option value=11>yellow
            <option value=12>blue
            <option value=13>fuchsia
            <option value=14>white
          </select>
      </FORM>
    </TD>
    </TR>
  </TABLE>
  </TD>
  <TD>
    <TABLE id="itab" border=1 cellspacing=10 cellpadding=20
           bgcolor=silver align=center>
      <TR>
        <TD id="itab1" bgcolor=gray>
          <A href="P1.jpg">
          <IMG src="P1.jpg" border="0" width="150"></A></TD>
        <TD id="itab2" bgcolor=gray >
          <A href="P2.jpg">
          <IMG src="P2.jpg" border="0"width="150"></A></TD>
      </TR>
      <TR>
        <TD id="itab3" bgcolor=gray>
          <A href="P3.jpg"><IMG src="P3.jpg"
                        border="0" width="150"></A>
        </TD>
        <TD id="itab4" bgcolor=gray>
          <A href="P4.jpg"><IMG src="P4.jpg"
                        border="0" width="150"></A>
        </TD></TR></TABLE></TD></TR>
  </TABLE>
</BODY>
</HTML>
```

Выравнивание изображений

Напишем сценарий, который позволяет указывать значения параметров горизонтального выравнивания для трех изображений, расположенных в документе. После выбора значения параметров следует нажать кнопку **Про-**

смотр, чтобы просмотреть документ при разных вариантах выравнивания. На рис. 6.9 представлен документ с выбранными параметрами горизонтального размещения.

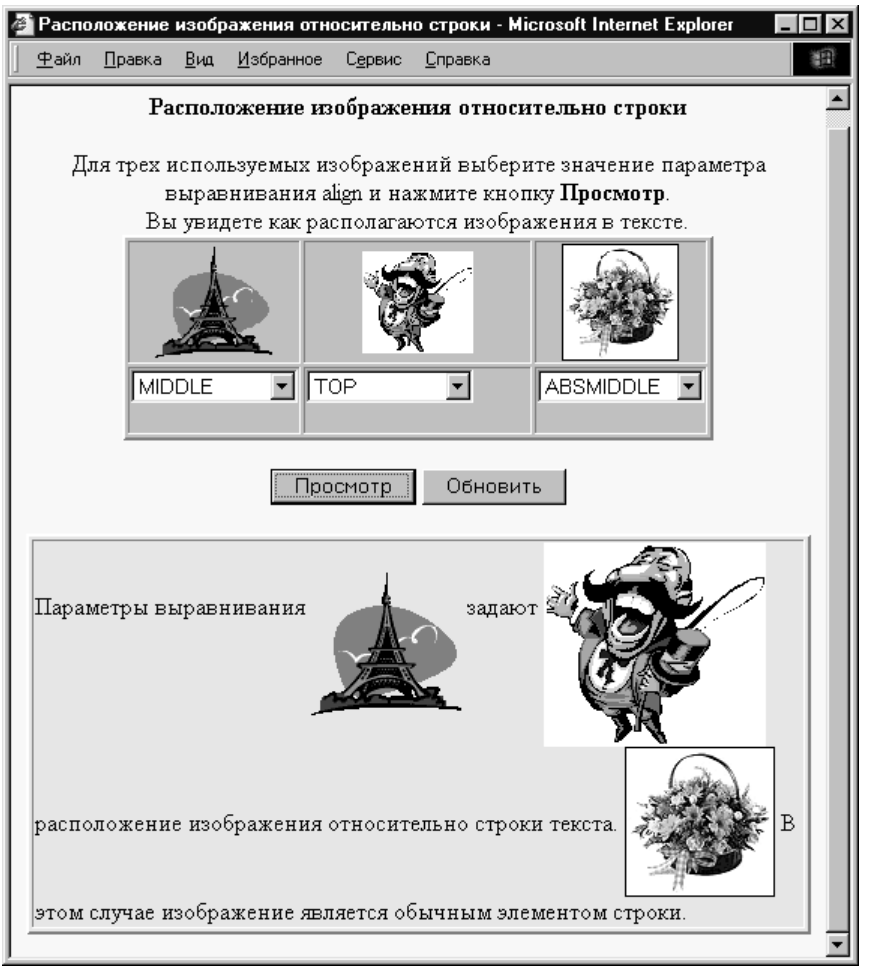


Рис. 6.9. Расположение нескольких изображений в тексте

В верхней части документов в таблице расположены рисунки и списки для выбора значений параметров выравнивания. При нажатии кнопки **Просмотр** изменяется расположение трех изображений относительно друг друга и текстовой строки. Код HTML для данного документа представлен в листинге 6.11.

Листинг 6.11. Горизонтальное выравнивание трех изображений в строке

```
<HTML>
  <HEAD>
    <TITLE>Расположение изображения относительно строки</TITLE>
    <script>
    <!--
      var k1="TOP"
      var k2="TOP"
      var k3="TOP"
      function setk(i)
       { var k
         var obj=eval("form"+i)
         if (((obj.elements[0])[0]).selected)
           k=((obj.elements[0])[0]).value
         if (((obj.elements[0])[1]).selected)
           k=((obj.elements[0])[1]).value
         if (((obj.elements[0])[2]).selected)
           k=((obj.elements[0])[2]).value
         if (((obj.elements[0])[3]).selected)
           k=((obj.elements[0])[3]).value
         if (((obj.elements[0])[4]).selected)
           k=((obj.elements[0])[4]).value
         if (((obj.elements[0])[5]).selected)
           k=((obj.elements[0])[5]).value
         if (i==1)
           k1=k
         else
           if (i==2)
             k2=k
           else
             k3=k
       }
      function set()
       { document.pict1.align=k1
         document.pict2.align=k2
         document.pict3.align=k3
       }
    //-->
```

```
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <CENTER>
  <H4 align=center>Расположение изображения относительно строки</H4>
  Для трех используемых изображений выберите значение параметра
  выравнивания align и нажмите кнопку <B>Просмотр</B>.<br>
  Вы увидете как располагаются изображения в тексте.
  <TABLE border=2 bgcolor=silver>
    <TR>
      <TD align=center><IMG src=p521.jpg width=50></TD>
      <TD align=center><IMG src=p522.gif></TD>
      <TD align=center><IMG src=p523.gif></TD>
    </TR>
    <TR>
      <TD><FORM name="form1">
          <select name=sel1 size=1 onchange="setk(1)">
            <option value="top" checked>TOP
            <option value="texttop">TEXTTOP
            <option value="middle">MIDDLE
            <option value="absmiddle">ABSMIDDLE
            <option value="bottom">BOTTOM
            <option value="absbottom">ABSBOTTOM
          </select>
      </FORM></TD>
      <TD><FORM name="form2">
          <select name=sel2 size=1 onchange="setk(2)">
            <option value="top" checked>TOP
            <option value="texttop">TEXTTOP
            <option value="middle">MIDDLE
            <option value="absmiddle">ABSMIDDLE
            <option value="bottom">BOTTOM
            <option value="absbottom">ABSBOTTOM
          </select>
      </FORM></TD>
      <TD><FORM name="form3">
          <select name=sel3 size=1 onchange="setk(3)">
            <option value="top" checked>TOP
            <option value="texttop">TEXTTOP
            <option value="middle">MIDDLE
```

```
            <option value="absmiddle">ABSMIDDLE
            <option value="bottom">BOTTOM
            <option value="absbottom">ABSBOTTOM
          </select>
        </FORM></TD>
    </TR></TABLE>
    <FORM>
      <input type="button" value= "Просмотр" onclick="set()">
      <input type="reset" value="Обновить">
    </FORM>
    </CENTER>
    <TABLE border=2 bgcolor="#FFDCDC">
      <TR><TD>
        Параметры выравнивания
        <IMG src=p521.jpg align=TOP name=pict1> задают
        <IMG src=p522.gif align=TOP name=pict2>
          расположение изображения относительно строки текста.
        <IMG src=p523.gif  align=TOP name=pict3>
        В этом случае изображение является обычным элементом строки.
    </TD></TR></TABLE>
  </BODY>
</HTML>
```

Упражнения

1. Напишите сценарий, который позволяет выбрать для таблицы и составляющих ее ячеек либо цвет фона, либо фоновое изображение, либо и то и другое. Предусмотрите возможность задания своего цвета фона для каждой ячейки.
2. Напишите сценарий, который позволяет посчитать стоимость предполагаемой покупки. Задается список продуктов, цена за единицу товара и количество экземпляров.
3. Напишите сценарий, обрабатывающий результаты ответа на вопросы по теме "Работа с архивами". Вопросы и возможные ответы представьте в виде списков.

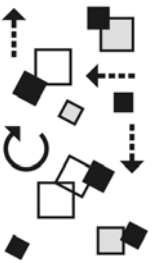
   Вопросы и варианты ответов таковы:

   - Выделите те программы, которые являются архиваторами.
     - ARJ
     - PKZIP

- ◊ LHA
- ◊ RAR
- ◊ WORD

- Для чего используются архивы?
  - ◊ Для сжатия информации
  - ◊ Для создания резервных копий
  - ◊ Для форматирования
  - ◊ Для хранения информации
- Что может архиватор?
  - ◊ Добавлять файлы в архив
  - ◊ Обновлять файлы в архиве
  - ◊ Удалять файлы из архива
- Какие архиваторы позволяют создавать многотомные архивы?
  - ◊ ARJ
  - ◊ PKZIP
  - ◊ LHA
  - ◊ RAR
- Основными характеристиками архиватора являются…
  - ◊ Скорость работы
  - ◊ Степень сжатия файлов
  - ◊ Сервисные функции
- На что указывает расширение в имени файла?
  - ◊ На обрабатываемый архивный файл
  - ◊ На используемую операционную систему
  - ◊ На используемый архиватор
  - ◊ На необходимую программу
- Архивация — это…
  - ◊ Создание архивной копии диска
  - ◊ Упаковка данных
  - ◊ Нанесение разметки на пустой диск
  - ◊ Работа в архиве
  - ◊ Извлечение файлов из архива
  - ◊ Ничто из перечисленного выше не верно

Глава 7



Фреймы

Окно просмотра браузера можно разбить на несколько прямоугольных областей, называемых *фреймами*. Области соприкасаются друг с другом и в каждую из областей можно загрузить отдельный HTML-документ, и работать с ним независимо от документов, загруженных в другие области окна или фрейма. Между фреймами можно организовать взаимодействие, например, выбор ссылки в одном из фреймов позволит изменить содержимое других фреймов. Фреймы часто используются в случаях, когда возникает необходимость загрузить документ в одну из областей при работе в другой области, или когда следует отобразить информацию, которая должна постоянно находиться на экране.

Простая фреймовая структура

Создадим документ, который разбивает область экрана на две части. Левая часть содержит оглавление разделов документа, который располагается в правой части. При выборе пункта оглавления в левой части появляется соответствующий раздел документа в правой части.

Разобьем область экрана на два фрейма. Левый фрейм занимает 25% ширины всего окна и будет содержать оглавление разделов документа, который загрузим в правый фрейм. Пусть имя файла, содержащего оглавление — contents.htm, а имя документа — ch.htm. Фреймовая структура задает способ организации экрана и определяет, какие документы должны быть первоначально загружены во фреймы. Создать описанную фреймовую структуру можно, если использовать документ, содержащий HTML-код, представленный в листинге 7.1, а.

Листинг 7.1, а. Создание простой фреймовой структуры

```
<HTML>
  <HEAD>
    <TITLE>Простая фреймовая структура</TITLE>
  </HEAD>
  <frameset cols="25%,75%">
    <frame src=contents0.htm name=left>
```

```
    <frame src= ch.htm name=right>
  </frameset>
</HTML>
```

Параметр `cols` тега `<frameset>` имеет вид `cols="`*список_значений*`"`. В списке через запятую перечисляются значения, которые определяют размеры фреймов. Список должен содержать не менее двух значений. Значения могут задаваться в процентах, в пикселах, в относительных единицах.

Тег `<frame>` определяет один фрейм. Он должен располагаться внутри парного тега `<frameset>` и `</frameset>`. Число тегов `<frame>` должно совпадать с количеством тегов, определенных при описании фреймовой структуры. В рассматриваемом примере в теге `<frameset cols="25%,75%">` определено два фрейма, поэтому в дальнейшем следует описание каждого из фреймов с помощью тега `<frame>`.

Значение параметра `src` тега `<frame>` определяет адрес документа, который первоначально загружается во фрейм. В рассматриваемом случае в левый фрейм загружается документ с именем contents0.htm, а в правый фрейм — документ с именем ch.htm. В теге параметр `name` определяет имя фрейма, необходимое для указания, в какой фрейм загрузить документ. Если имя фрейма не задавать, то будет создан фрейм без имени, но сослаться на него из других фреймов будет нельзя.

Связь между фреймами и документами для описанного примера изображена на рис. 7.1.

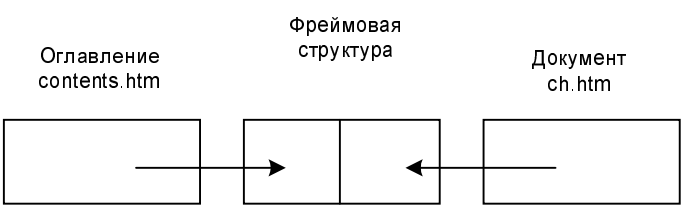


Рис. 7.1. Связь между фреймами и документами

Файл ch.htm содержит так называемые внутренние ссылки, т. е. разбит на четыре части. В документе описаны различные способы выравнивания текста, заголовка и горизонтальной линии. Вид документа приведен на рис. 7.2.

Перед заголовком, поясняющим действие параметра при выравнивании по левому краю, написан тег `<A name=chapter_1></A>`, помечающий начало раздела. К этому разделу можно осуществить переход, например, по ссылке **Возврат**, заданной следующим образом:

```
<A href="#chapter_1">Возврат</A>
```

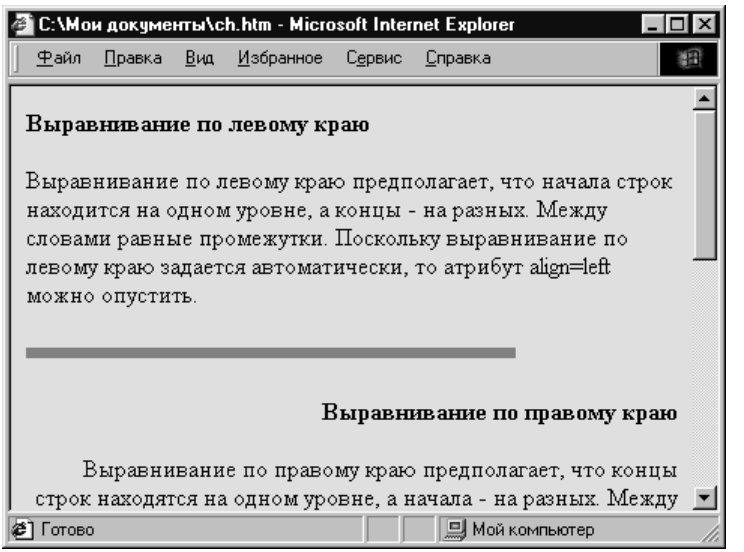


Рис. 7.2. Содержимое документа ch.htm

HTML-код документа, представляющего оглавление, которое загружается в левый фрейм, представлен в листинге 7.1, б.

Листинг 7.1, б. Содержимое документа contents0.htm

```
<HTML>
  <HEAD>
    <TITLE>Ссылки внутри документа</TITLE>
  </HEAD>
  <BODY background="decor.gif" bgcolor=silver>
    <h4 align=center>Выравнивание</h4>
    <A href="ch.htm#chapter_1" target=right>по левому краю</A><br>
    <A href="ch.htm#chapter_2" target=right>по правому краю</A><br>
    <A href="ch.htm#chapter_3" target=right>по центру</A><br>
    <A href="ch.htm#chapter_4" target=right>по ширине</A><br>
  </BODY>
</HTML>
```

Ссылка устанавливается на соответствующий раздел документа заданием параметра `href="ch.htm#chapter_1"`. К каждой из частей осуществляется переход по ссылке из оглавления, расположенного в левом фрейме.

Параметр `target` определяет имя фрейма, в который загружается документ, на который установлена ссылка. В рассматриваемом случае это фрейм с

именем `right`. По умолчанию или при отсутствии параметра `target` документ загружается в текущий фрейм или окно. На рис. 7.3 представлен документ, имеющий описанную фреймовую структуру.

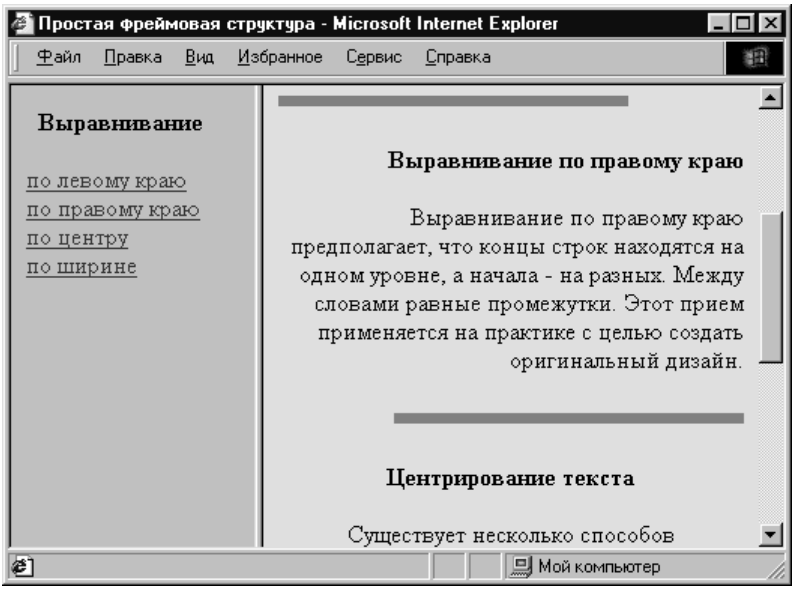


Рис. 7.3. Документ, имеющий фреймовую структуру

Для всех ссылок в документе с именем contents.htm указано одинаковое значение параметра `target`. Если все документы, загружаемые по ссылкам, должны загружаться в один и тот же фрейм, то можно задать параметр `target` в теге `<BASE>`. В этом случае HTML-код файла, содержащего оглавление можно описать так, как указано в листинге 7.1, в.

Листинг 7.1, в. Ссылки внутри документа

```
<HTML>
  <HEAD>
    <TITLE>Ссылки внутри документа</TITLE>
  </HEAD>
  <BODY background="decor.gif"  bgcolor=silver>
    <base target=right >
    <A name=chapter_0> </A>
    <h4 align=center>Выравнивание</h4>
    <A href="ch.htm#chapter_1">по левому краю</A><br>
```

```
    <A href="ch.htm#chapter_2">по правому краю</A><br>
    <A href="ch.htm#chapter_3">по центру</A><br>
    <A href="ch.htm#chapter_4">по ширине</A><br>
  </BODY>
</HTML>
```

Фреймовая структура с загружаемыми документами

Создадим документ, левая часть которого представляет оглавление, а в правую часть загружаются документы выбранного пункта оглавления. Документы, соответствующие пунктам оглавления, хранятся в разных файлах.

При решении задачи экран по-прежнему разбивается на два фрейма. Левый фрейм занимает 30% ширины всего окна и будет содержать оглавление документов, которые могут быть просмотрены пользователем при выборе соответствующего пункта. Правый фрейм занимает большую часть окна просмотра и предназначен для отображения самих документов. При первоначальной загрузке оба фрейма делят окно просмотра по вертикали в соотношении 30% и 70%. Данное соотношение может меняться при просмотре. Каждый из фреймов имеет свою полосу прокрутки, обеспечивающую просмотр всего документа. При выборе ссылки в левом фрейме соответствующий документ будет загружен в правый фрейм. Такая структура позволяет одновременно видеть на экране и оглавление документов, и сами документы.

Пусть оглавление документа содержит шесть пунктов и располагается в файле с именем contents.htm. Требуется, чтобы файл, содержащий оглавление, загружался в левый фрейм. Файлы с именами ch1.htm, ch2.htm, ..., ch6.htm содержат документы, соответствующие пунктам оглавления. Взаимодействие между фреймами и документами представлено на рис. 7.4.

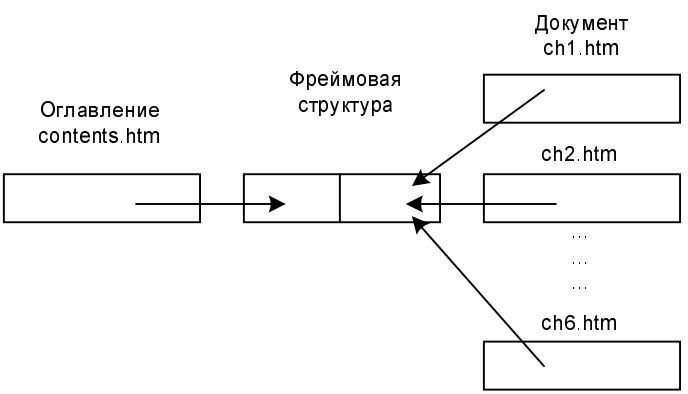


Рис. 7.4. Фреймовая структура с загружаемыми документами

Фреймовая структура мало отличается от той, какая была рассмотрена в предыдущем примере (листинг 7.2, а).

Листинг 7.2, а. Задание фреймовой структуры

```
<HTML>
  <HEAD>
    <TITLE>Простая фреймовая структура</TITLE>
  </HEAD>
  <frameset cols="30%,70%">
    <frame src=contents.htm name=left>
    <frame src=empty.htm name=right>
  </frameset>
</HTML>
```

В правый фрейм первоначально загружается файл с именем empty.htm. Если сразу неизвестно, какой файл загружать во фрейм, то можно использовать файл, содержащий HTML-код (листинг 7.2, б).

Листинг 7.2, б. Документ для первоначальной загрузки

```
<HTML>
  <HEAD>
    <TITLE>Пустой документ</TITLE>
  </HEAD>
  <BODY>
  </BODY>
</HTML>
```

В левый фрейм помещается оглавление, которое содержит ссылки на документы, расположенные в различных файлах. Оглавление может быть сформировано так, как указано в листинге 7.2, в.

Листинг 7.2, в. Оглавление, загружаемое в левый фрейм

```
<HTML>
  <HEAD>
    <TITLE>Оглавление</TITLE>
  </HEAD>
  <BODY background="decor.gif" bgcolor=silver>
```

```
    <base target=right>
    <h3>Оглавление</h3>
    <OL>
      <LI><A href="ch1.htm">Основы языка HTML </A>
      <LI><A href="ch2.htm">Графика </A>
      <LI><A href="ch3.htm">Изображение-карта </A>
      <LI><A href="ch4.htm">Списки </A>
      <LI><A href="ch5.htm">Таблицы </A>
      <LI><A href="ch6.htm">Фреймы </A>
    </OL>
  </BODY>
</HTML>
```

Во многих случаях при начальной загрузке в файл помещают страницу с титульным листом, как в случае, представленном на рис. 7.5

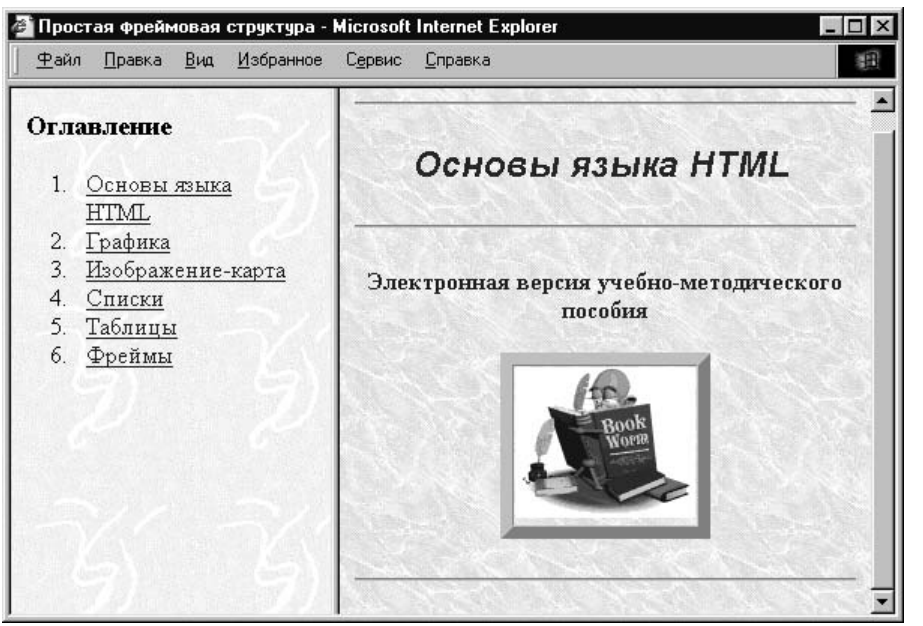


Рис. 7.5. Документ со стартовой страницей

При описании фреймовой структуры в этом случае следует для правого фрейма задать адрес файла с титульной страницей. При выборе пункта меню в левом фрейме соответствующий документ появляется в правом фрейме.

Фреймовая структура с раскрывающимся оглавлением одиночного пункта

Создайте документ, левая часть которого представляет оглавление, а в правую часть загружаются соответствующие оглавлению документы. Предусмотрите возможность "раскрытия" пунктов оглавления и перехода к документам по подпунктам.

Предположим, что в левом фрейме представлено оглавление, содержащее пункты верхнего уровня. Требуется при щелчке по пункту оглавления раскрыть список подпунктов и предусмотреть возможность перехода к соответствующему подпункту разделу документа. Такого рода задачи решаются разными способами. Рассмотрим простые решения.

Для каждого пункта меню создадим документ, в котором соответствующий пункт меню "раскрыт". Имена документов cont1.htm, cont2.htm, ..., cont6.htm. Документ с именем cont5.htm содержит оглавление с подпунктами пятого пункта основного оглавления. При выборе соответствующего подпункта происходит переход к соответствующему разделу документа с именем ch5.htm. HTML-код документа cont5.htm имеет вид, представленный в листинге 7.3.

Листинг 7.3. Оглавление с раскрытым пунктом 5

```
<HTML>
  <HEAD>
    <TITLE>Оглавление с раскрытым пунктом 5</TITLE>
  </HEAD>
  <BODY background="decor.gif" bgcolor=silver>
    <base target=right>
    <h3>Оглавление</h3>
    <OL>
      <LI><A href="ch1.htm">Основы языка HTML </A>
      <LI><A href="ch2.htm">Графика </A>
      <LI><A href="ch3.htm">Карта-изображение </A>
      <LI><A href="ch4.htm">Списки </A>
      <LI><A href="ch5.htm"
             onclick="top.left.location='contentsR.htm'">Таблицы </A>
      <UL>
        <LI><A href="ch5.htm#5ch_1">Простые таблицы</A>
        <LI><A href="ch5.htm#5ch_2">Данные внутри таблицы</A>
        <LI><A href="ch5.htm#5ch_3">Вложенные таблицы</A>
```

```
      </UL>
      <LI><A href="ch6.htm">Фреймы </A>
    </OL>
  </BODY>
</HTML>
```

В документе ch5.htm расположены теги, отмечающие те разделы, к которым в дальнейшем можно обратиться по соответствующему подпункту:

```
<A name=5ch_1></A>
<A name=5ch_2></A>
<A name=5ch_3></A>
```

В рассмотренных примерах страница разбивалась на два вертикальных фрейма: левый с именем `left` и правый с именем `right`. Фреймы образуют иерархическую модель объектов `Frame`. На верхнем уровне расположен объект `top`, являющийся родителем двух описанных фреймов. Для ссылки на фреймы можно использовать имена. Для того чтобы сослаться на левый фрейм, следует указать конструкцию `top.left`. Аналогично, для ссылки на правый фрейм — `top.right`. Свойство фрейма `location` определяет URL-адрес загруженного во фрейм документа. При выполнении присваивания `top.left.location='contentsR.htm'` во фрейм с именем `left` загружается файл с именем `'contentsR.htm'` из текущей папки.

Вид документа с "раскрытым" пунктом меню после перехода по ссылке изображен на рис. 7.6.

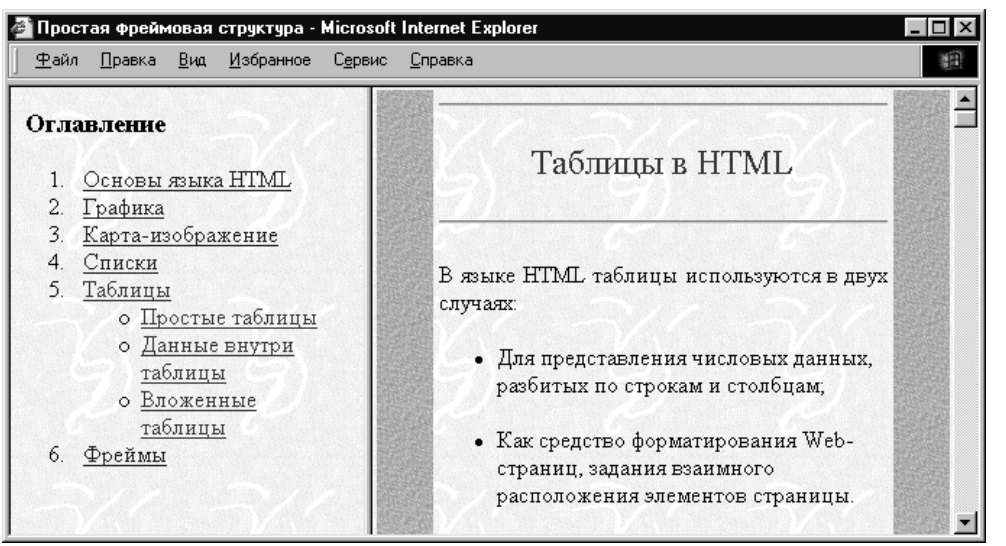


Рис. 7.6. Пример оглавления с раскрытым пунктом меню

После щелчка мышью по основному пункту меню, подменю "сворачивается". Реакция на событие `onClick` — загрузка в левый фрейм файла, содержащего основное меню, т. е. меню с именем contentsR.htm. При щелчке по пункту меню **Таблицы** в основном меню происходит загрузка в левый фрейм файла cont5.chm. Когда пункт меню раскрыт, можно осуществлять переходы по подпунктам к соответствующим разделам документа.

Для ссылки на фреймы страницы можно использовать свойство-массив `frames` объекта `top`, в котором содержатся ссылки на все фреймы страницы. В рассмотренных примерах на фреймы можно было сослаться следующим образом:

```
top.frames[0]
top.frames[1]
```

Фреймовая структура с раскрывающимся оглавлением всех пунктов

Создадим документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: левую и правую. В левой области поместите оглавление в виде списка, в правом — документ, загруженный в соответствии с выбранным пунктом оглавления. Предусмотрите возможность работы в режиме, когда пользователь может "раскрыть" или "свернуть" одновременно все пункты оглавления.

При решении задачи поступим следующим образом: создадим два оглавления, одно содержит только пункты верхнего уровня, т. е. "свернутое" оглавление, а во втором оглавлении все пункты раскрыты. В обоих документах предусмотрены кнопки, при нажатии которых меню "сворачивается" или "раскрывается". Приведем пример основного оглавления (листинг 7.4).

Листинг 7.4. Основное оглавление

```
<HTML>
  <HEAD>
    <TITLE>Основное оглавление</TITLE>
    <script>
    <!--//
      function chcont(n)
      { if (n==1)
          top.left.location="contents.htm"
        else
          top.left.location="contall.htm"
      }
```

```
    //-->
    </script>
  </HEAD>
  <BODY background="decor.gif" bgcolor=silver>
    <base target=right >
    <FORM name=form1>
      <input type="button" value= раскрыть onclick="chcont(0)">
      <input type="button" value= свернуть onclick="chcont(1)">
    </FORM>
    <H3>Оглавление</H3>
    <OL>
      <LI><A href="ch1.htm">Основы языка HTML </A>
      <LI><A href="ch2.htm">Графика </A>
      <LI><A href="ch3.htm">Карта-изображение </A>
      <LI><A href="ch4.htm">Списки </A>
      <LI><A href="ch5.htm">Таблицы </A>
      <LI><A href="ch6.htm">Фреймы </A>
    </OL>
  </BODY>
</HTML>
```

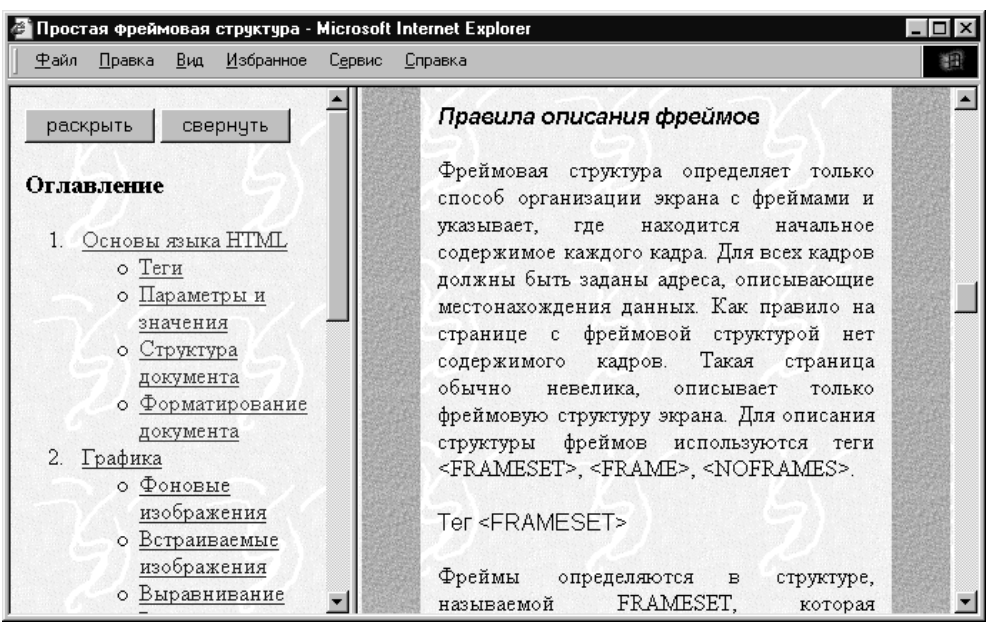


Рис. 7.7. Оглавление в "раскрытом" виде

Функция `chcont()` в зависимости от значения параметра загружает в левый фрейм одно из двух оглавлений. На рис. 7.7 приведен пример, когда оглавление раскрыто и в правый фрейм загружен соответствующий выбранному подпункту документ.

Функция `chcont()` присутствует в двух документах, содержащих оглавление.

Фреймовая структура из трех фреймов

Создадим документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Левая область состоит из двух частей: в верхней части должны быть расположены кнопки, управляющие видом оглавления, в нижней части — само оглавление в свернутом или развернутом виде. Правая часть обязана содержать собственно документ, соответствующий выбранному пункту оглавления.

При решении этой задачи будет сформирована фреймовая структура более сложного вида, чем та, которую рассматривали ранее. Документ делится по вертикали на две области. Левая область, в свою очередь, разбивается с помощью фреймов по горизонтали еще на две части. Параметр `rows="`*`список_значений`*`"` задает количество и размеры фреймов по горизонтали. При решении нашей задачи определим для верхнего фрейма 60% высоты окна просмотра и 40% — для нижнего. Фреймовая структура описана в листинге 7.5.

Листинг 7.5. Задание фреймовой структуры из трех фреймов

```
<HTML>
  <HEAD>
    <TITLE>Три фрейма и раскрывающееся оглавление</TITLE>
  </HEAD>
  <frameset cols="32%,*">
    <frameset rows="60,*">
      <frame src=but.htm name=topleft>
      <frame src=contents.htm name=left>
    </frameset>
    <frame src=start.htm name=right marginwidth=1>
  </frameset>
</HTML>
```

Рис. 7.8. Документ с кнопками управления видом оглавления

В левый верхний фрейм загружается документ, содержащий описание кнопок и действий при щелчке по этим кнопкам. HTML- код документа имеет вид:

```
<HTML>
  <HEAD>
    <TITLE>Кнопки раскрывающегося меню</TITLE>
    <script>
    <!--//
      function chcont(n)
        { if (n==1)
            top.left.location="contents.htm"
          else
            top.left.location="contall.htm"
        }
    //-->
    </script>
  </HEAD>
  <BODY>
  <BODY background="decor.gif" bgcolor=silver>
```

```
    <FORM name=form1>
      <input type="button" value= раскрыть onclick="chcont(0)">
      <input type="button" value= свернуть onclick="chcont(1)">
    </FORM>
  </BODY>
</HTML>
```

В левый нижний фрейм загружается оглавление. Первоначально это файл, содержащий нераскрытое оглавление. В правый фрейм загружается файл, содержащий титульный лист. На рис. 7.8 представлен документ, соответствующий описанной структуре.

Обмен содержимым фреймов

Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область содержит два фрейма, нижняя область состоит из одного фрейма. В нижней области располагается кнопка, которая обменивает содержимое верхних фреймов.

Область окна просмотра сначала разбивается на два фрейма по горизонтали. При описании тега `<frameset rows="*,50">` определяется, что размер нижнего фрейма 50 пикселов, остальное пространство отводится под верхний фрейм. Верхний фрейм в свою очередь делится по вертикали на два фрейма, размеры которых заданы в процентах к области просмотра. HTML-код документа, создающий описанную фреймовую структуру, выглядит так, как указано в листинге 7.6, а.

Листинг 7.6, а. Три фрейма с кнопкой для обмена содержимого фреймов

```
<HTML>
  <HEAD>
    <TITLE>Три фрейма с кнопкой для обмена</TITLE>
  </HEAD>
  <frameset rows="*,50">
    <frameset cols="55%,45%">
      <frame src=lpict.htm name=left>
       <frame src=rtext.htm name=right>
    </frameset>
    <frame src=but1.htm name=butt>
  </frameset>
</HTML>
```

Для ссылки на фреймы будем, как и ранее, использовать имена. Сначала следует запомнить имя файла, загруженного в левый фрейм. Для этого используется переменная `l`. Затем в левый фрейм загружается документ, расположенный в правом фрейме. Это достигается выполнением оператора присваивания

```
top.left.location=top.right.location
```

И, наконец, в правый фрейм загружается тот документ, адрес которого запомнили в переменной `l`.

HTML-код для нижнего фрейма представлен в листинге 7.6, б.

Листинг 7.6, б. Сценарий для нижнего фрейма

```
<HTML>
  <HEAD>
    <TITLE>Кнопка для смены содержимого фреймов</TITLE>
    <script>
    <!--//
      function chframe()
       { var l=top.left.location
         top.left.location=top.right.location
         top.right.location=l
       }
    //-->
    </script>
  </HEAD>
  <BODY background="decor.gif" bgcolor=silver>
    <CENTER>
    <FORM name=form1>
      <input type="button" value=Обмен onclick="chframe()">
    </FORM>
    </CENTER>
  </BODY>
</HTML>
```

На рис. 7.9 приведен документ с описанной фреймовой структурой.

При щелчке по кнопке **Обмен** содержимое верхних фреймов поменяется местами.

Рис. 7.9. Обмен содержимым двух фреймов

Функция, осуществляющая обмен содержимым фреймов, может быть описана следующим образом:

```
function chframe()
 { var l=top.frames[0].location
   top.frames[0].location=top.frames[1].location
   top.frames [1].location=l
 }
```

Простой пример использования плавающих фреймов

Браузер Microsoft Internet Explorer разрешает использование так называемых *плавающих фреймов*, описание которых может быть расположено в тексте обычного HTML-документа. Для определения плавающего фрейма используется тег `<iframe>`. В указанном теге могут встречаться те же параметры, что и в теге описания обычных фреймов, кроме параметра `noresize`, т. к. размер плавающего фрейма не может быть изменен пользователем. Браузе-

ры, не поддерживающие тег <iframe>, отображают записанную между тегами <iframe> и </iframe> информацию. В следующем примере HTML-документа используется описание плавающего фрейма, которое напоминает синтаксис тега встраиваемого в документ изображения. Параметр src определяет адрес загружаемого во фрейм изображения. Параметр name задает имя фрейма для того, чтобы можно было получить доступ к фрейму, например, из сценария.

Сценарий представлен в листинге 7.7, а.

Листинг 7.7, а. Использование плавающих фреймов

```
<HTML>
  <HEAD>
    <TITLE> Использование плавающих фреймов</TITLE>
  </HEAD>
  <BODY bgcolor="FFFFCC">
    <H4 align=center>Шишкин Иван Иванович</H4>
    <iframe src=textsh.htm name="flframe">
      Отображение плавающих фреймов в Вашем браузере не предусмотрено
    </iframe>
    <OL>
      <LI><A href=pictsh.htm target="flframe">Картина</A></LI>
      <LI><A href=textsh.htm target="flframe">Описание</A></LI>
    </OL>
  </BODY>
</HTML>
```

В плавающий фрейм можно загрузить другие документы, используя определенные на странице гиперссылки. Результат отображения примера приведен на рис. 7.10.

Для плавающих фреймов с помощью параметров можно задавать размеры фрейма, горизонтальное выравнивание, размер отступа содержимого фрейма от границ. В следующем примере приведено описание плавающего фрейма и заданы параметры. Фрейм имеет высоту 320 пикселов, занимает по ширине 60% окна, расположен справа от текста, полосы прокрутки будут установлены в случае, если документ не станет помещаться во фрейм. Содержимое фрейма должно быть отделено от границы по горизонтали и вертикали на заданное число пикселов.

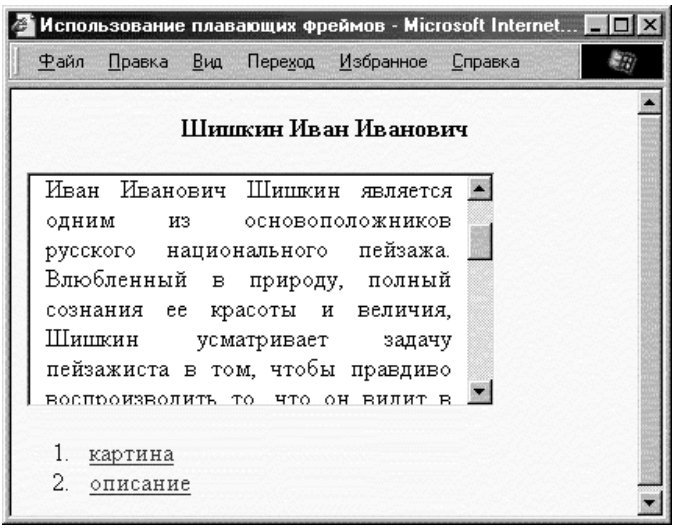


Рис. 7.10. Плавающие фреймы

Сценарий представлен в листинге 7.7, б.

Листинг 7.7, б. Плавающие фреймы и их параметры

```
<HTML>
  <HEAD>
    <TITLE> Плавающие фреймы и их параметры</TITLE>
  </HEAD>
  <BODY bgcolor="FFFFCC">
    <h4 align=center>Творчество русских художников</h4>
    <iframe src=pictsh.htm name="flframe" height=320 width=60% hspace=50
            vspace=10 scrolling=auto align=right>
      Ваш браузер не позволяет отображать плавающие фреймы
    </iframe>
    <OL>
      <LI><A href=pictsh.htm target="flframe">Картина</A></LI>
      <LI><A href=textsh.htm target="flframe">Описание</A></LI>
    </OL>
  </BODY>
</HTML>
```

Приведенный пример браузер Microsoft Internet Explorer отобразит так, как на рис. 7.11.

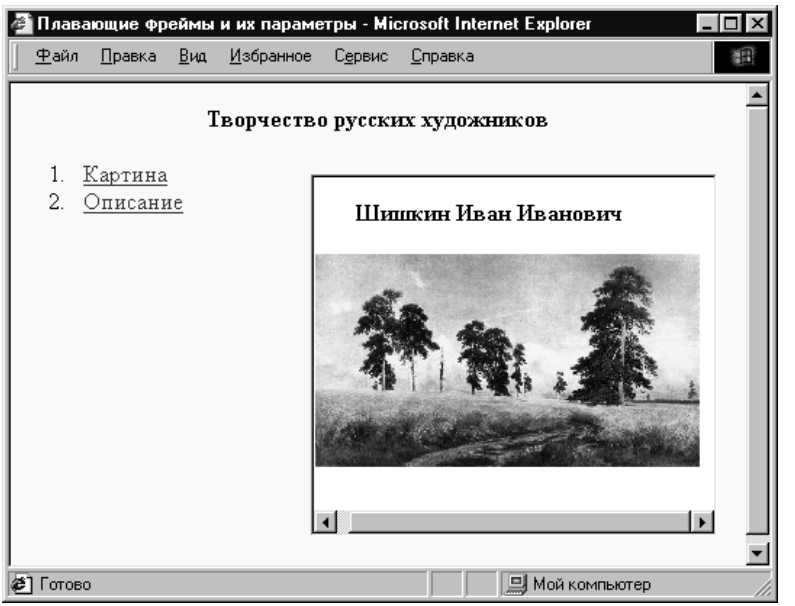


Рис. 7.11. Изменение содержимого плавающего фрейма

Плавающие фреймы и организация гиперссылок

Создадим сценарий, в котором с левой стороны хранится список задач. При выборе задачи из списка ее условие появляется в плавающем фрейме, который расположен в окне просмотра справа.

В документе с условиями задач перед условием первой задачи расположим два тега `<A name="1"></A>`, а далее перед условием задачи с номером `i` расположим тег `<A>`, параметр `name` которого равен `i`. Таким образом, мы подготовим текст условий задач для ссылок на задачи с соответствующим номером. Пусть имя файла с условиями задач indiv.htm. HTML-код документа, содержащего список задач, и плавающий фрейм, представлен в листинге 7.8.

Листинг 7.8. Использование плавающих фреймов

```
<HTML>
  <HEAD>
    <TITLE> Использование плавающих фреймов</TITLE>
  </HEAD>
  <BODY background="decor.gif">
    <base target="flframe">
    <H4 align=center>Для выполнения  задания выберите задачу </H4>
    <iframe src=indiv.htm name="flframe" height=350 width=70% hspace=10
            scrolling=auto align=right>
      Ваш браузер не позволяет отображать плавающие термы
    </iframe>
    <H4>Темы заданий </H4>
    <OL>
      <LI><A href="indiv.htm#1">экспорт товаров</A></LI>
      <LI><A href="indiv.htm#2">курсовые работы</A></LI>
      <LI><A href="indiv.htm#3">график обследования</A></LI>
      <LI><A href="indiv.htm#4">отгрузка со склада</A></LI>
      <LI><A href="indiv.htm#5">технический осмотр</A></LI>
      <LI><A href="indiv.htm#6">пансионат</A></LI>
      <LI><A href="indiv.htm#7">регистрация переговоров</A></LI>
      <LI><A href="indiv.htm#8">расписание экзаменов</A></LI>
      <LI><A href="indiv.htm#9">учет товаров</A></LI>
      <LI><A href="indiv.htm#10">выдача ключей</A></LI>
      <LI><A href="indiv.htm#11">абонементы</A></LI>
      <LI><A href="indiv.htm#12">тестирование</A></LI>
      <LI><A href="indiv.htm#13">соревнование</A></LI>
      <LI><A href="indiv.htm#14">семейный бюджет</A></LI>
      <LI><A href="indiv.htm#15">экзамены</A></LI>
    </OL>
  </BODY>
</HTML>
```

Результат отображения описанной страницы приведен на рис. 7.12.

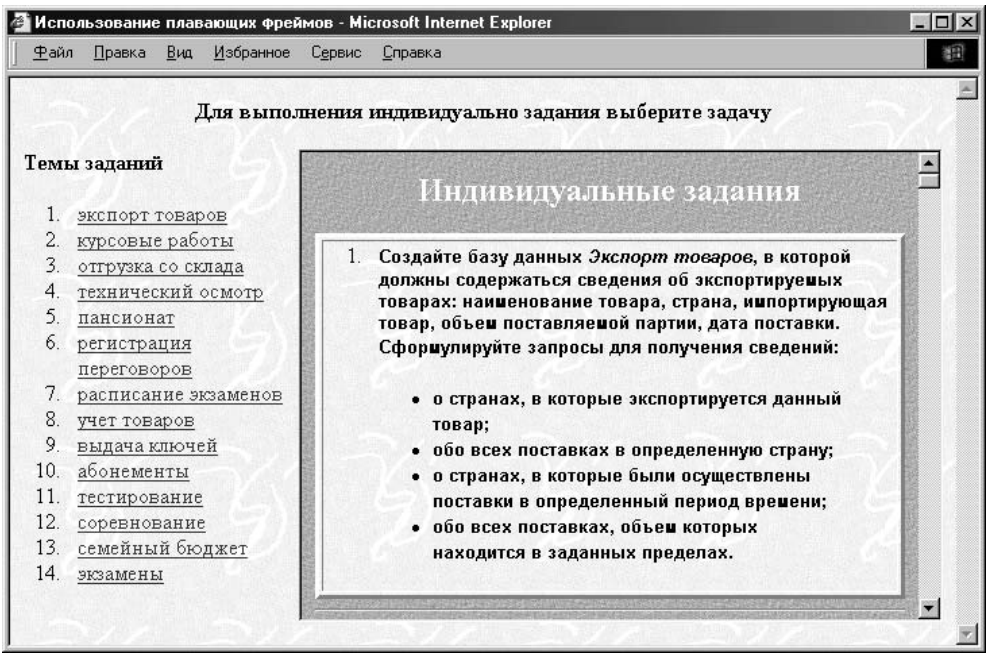


Рис. 7.12. Оглавление плавающего фрейма

Упражнения

1. Создайте документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: верхнюю и нижнюю. В верхней области поместите оглавление в виде списка, при выборе пунктов которого соответствующий раздел должен появляться в нижней части окна.
2. Создайте документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: верхнюю и нижнюю. В верхней области поместите оглавление, представленное с помощью графического горизонтального меню.
3. Создайте документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: верхнюю и нижнюю. В верхней области поместите оглавление. Оглавление представьте с помощью изображения-карты. При выборе активной области в нижнюю область окна должен быть загружен соответствующий пункту раздел.
4. Выполните предыдущие три задания при условии, что пункты меню соответствуют разделам одного документа.
5. Выполните первые три задания при условии, что пункты меню соответствуют разделам, хранящимся в разных документах.

6. Создайте документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: левую и правую. В левой области поместите оглавление, при выборе пунктов которого соответствующий пункту раздел должен появляться в правой части окна. Оглавление представьте с помощью графического вертикального меню.
7. Создайте документ, разбивающий окно просмотра с помощью фреймов на две прямоугольные области: левую и правую. В левой области поместите оглавление, при выборе пунктов которого соответствующий пункту раздел должен появляться в правой части окна. Оглавление представьте с помощью изображения-карты.
8. Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область (A) занимает по ширине весь экран, нижняя область окна разбивается на две части: левую (B) и правую (C). В верхнюю область A поместите графическое изображение, в область B — оглавление, при выборе пунктов которого соответствующий раздел должен появляться в области C. Оглавление может быть представлено:
   - списком;
   - графическим меню;
   - изображением-картой.

   Пунктам меню могут соответствовать разделы:
   - одного документа;
   - разных документов.

   Представьте материалы в виде HTML-документов с описанной структурой.
9. Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область (A) занимает по ширине весь экран, нижняя область окна разбивается на две части: левую (B) и правую (C). В область B поместите оглавление, представленное графическим меню. Разделы, соответствующие пунктам меню, должны появляться в области C. При попадании курсора мыши на соответствующий пункт меню, в области A должна появляться краткая аннотация выбранного пункта. Представьте материалы в виде HTML-документов с описанной структурой.
10. Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область (A) занимает по ширине весь экран, нижняя область окна разбивается на две части: левую (B) и правую (C). В верхнюю область A поместите изображение-карту, дублирующую меню области B.
11. Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область окна делится на две

части: левую (B) и правую (C). Нижняя область (A) занимает по ширине весь экран. В область A поместите оглавление для некоторых художественных шедевров. При выборе пункта оглавления в части B должна появиться репродукция картины, в части — C сведения о художнике или о картине.

12. Создайте документ, разбивающий окно просмотра с помощью фреймов на три прямоугольные области. Верхняя область окна разбивается на две части: левую (B) и правую (C). Нижняя область (A) занимает по ширине весь экран. В область B поместите оглавление, представленное графическим меню. Разделы, соответствующие пунктам меню и содержащие теоретический материал, должны появляться в области C. В область A поместите документ, содержащий ссылки на примеры, практическую работу, контрольные вопросы, список литературы по выбранному разделу.

13. Создайте документ, разбивающий окно просмотра с помощью фреймов на четыре прямоугольные области, как показано на рис. 7.13. В области A поместите оглавление. При выборе пункта оглавления в область B должен помещаться теоретический материал, соответствующий пункту меню, в область C — список контрольных вопросов по рассматриваемой теме, в область D — практическая работа. Представьте материалы в виде HTML-документов описанной структуры.

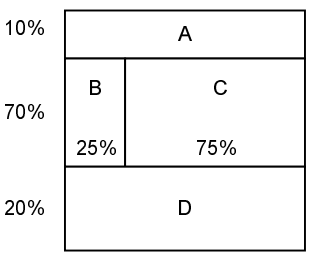


Рис. 7.13. Фреймовая структура к упражнению 13

14. Создайте документ, разбивающий окно просмотра с помощью фреймов на четыре прямоугольные области. Первая область (A) занимает по ширине все окно и 10% высоты окна. В эту область следует помещать графические изображения, которые чередуются через каждые десять секунд. Две области — B и C — занимают по высоте 70% окна и по ширине 20% и 80% соответственно. В область B поместите оглавление в виде списка, в области C должны появляться разделы, соответствующие пунктам оглавления. Четвертая часть D занимает по ширине всю область окна и 20% высоты окна. В области D разместите четыре кнопки, обеспечивающие навигацию по пунктам оглавления. Одна кнопка позволяет осуществить переход к началу документа (первый пункт оглавления), вторая — к концу (последний пункт оглавления), две другие — к

следующему и предыдущему разделу по отношению к текущему. Текущий раздел содержится в области C:

- разделы, соответствующие пунктам меню, расположены в одном документе;
- разделы, соответствующие пунктам меню, расположены в разных документах.

15. Создайте документ, разбивающий окно просмотра с помощью фреймов на четыре прямоугольные области, как показано на рис. 7.14. В область A помещается список тем. При выборе темы в области C отображаются разделы соответствующей темы. Выбор раздела области C приводит к отображению области B связанного документа. При этом в области D появляется соответствующее теме изображение. Оформите материал в виде HTML-документов так, чтобы он удовлетворял описанной схеме.

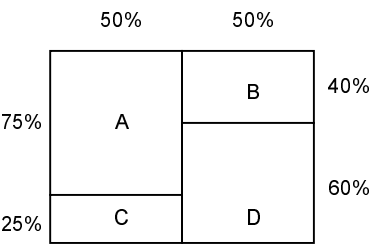
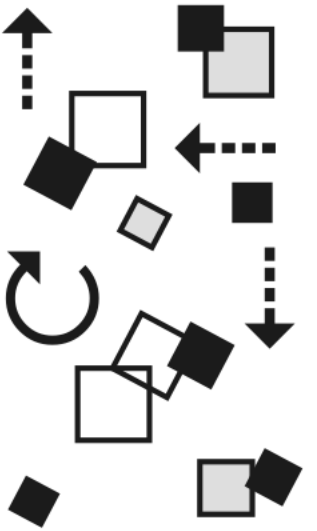


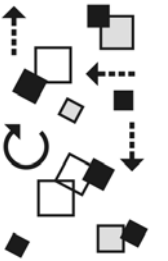
Рис. 7.14. Фреймовая структура к упражнению 15



ЧАСТЬ II

Основные объекты JavaScript и методы работы с ним

Глава 8



Повторяющиеся вычисления

Для успешного решения широкого круга задач требуется многократно повторить некоторую последовательность действий, записанную в программе один раз. В том случае, когда число повторений последовательности действий нам неизвестно, либо число повторений зависит от некоторых условий, можно воспользоваться *оператором цикла* вида:

```
while (B) {S}
```

где `B` — выражение логического типа; `S` — операторы, называемые *телом цикла*. Операторы `S` в фигурных скобках выполняются до тех пор, пока условие `B` не станет ложным.

Нахождение общего делителя

Напишем программу, которая для двух заданных чисел определяет наибольший общий делитель.

При решении задачи воспользуемся алгоритмом Евклида. Если значение m равно нулю, то наибольший общий делитель чисел n и m равен n:

НОД(n, 0) = n.

В остальных случаях верно следующее соотношение:

НОД(n, m) = НОД(m, $n\%m$).

В функции `nod` переменная `p` используется для получения остатка от деления чисел `n` и `m` (листинг 8.1). Выполнение цикла продолжается до тех пор, пока значение `p` не станет равным нулю. Последнее вычисленное значение `m` равно наибольшему общему делителю.

Листинг 8.1. Наибольший общий делитель двух чисел

```
<HTML>
  <HEAD>
    <TITLE>Наибольший общий делитель двух чисел</TITLE>
    <script language="JavaScript">
    <!-- //
      function nod(obj)
```

```
        { var n=obj.num1.value
          var m=obj.num2.value
          var p = n%m
          while (p!=0)
           { n=m
             m=p
             p=n%m
           }
          obj.res.value=m
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    Наибольший общий делитель двух заданных чисел
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить" onClick="nod(form1)"><br>
      Наибольший общий делитель  <input type="text" name="res"
                                        size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

На рис. 8.1 приведен результат выполнения сценария для заданных натуральных чисел.

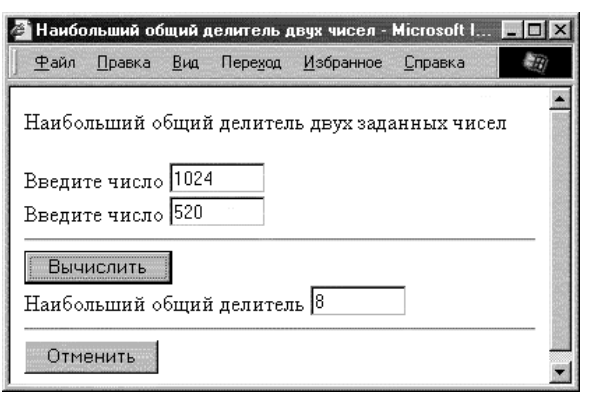


Рис. 8.1. Наибольший общий делитель двух натуральных чисел

Определение наименьшего общего кратного

Напишем программу, определяющую наименьшее общее кратное двух заданных натуральных чисел.

Для того чтобы определить наименьшее общее кратное, вычислим наибольший общий делитель и воспользуемся следующим равенством:

НОК(n, m) = $n \times m$/НОД(n, m).

В отличие от предыдущего варианта опишем функцию `nod` с двумя параметрами. Вычисленное значение наибольшего общего делителя будет выдаваться в качестве результата функции. Функция `nok` простая, ее действия состоят в вызове функции `nod` и выдаче в качестве результата функции значения выражения. Значением параметра обработки события в данном случае является оператор присваивания, обеспечивающий запись вычисленного значения в соответствующее поле формы (листинг 8.2).

Листинг 8.2. Наименьшее общее кратное двух натуральных чисел

```
<HTML>
  <HEAD>
    <TITLE>Наименьшее общее кратное двух натуральных чисел</TITLE>
    <script language="JavaScript">
    <!-- //
      function nod(n,m)
       { var p=n%m
         while (p!=0)
          { n=m
            m=p
            p=n%m
          }
         return m
       }
      function nok(n,m)
       { return n*m/nod(n,m) }
    //-->
    </script>
  </HEAD>
  <BODY>
    Наименьшее общее кратное двух заданных натуральных чисел
```

```
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить"
      onClick="form1.res.value=nok(form1.num1.value,form1.num2.value)">
      <br>
      Наименьшее общее кратное  <input type="text" name="res"
                                     size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Определение взаимно простых чисел

Необходимо написать программу, определяющую, являются ли два заданных натуральных числа взаимно простыми. Взаимно простые числа не имеют общих делителей кроме 1.

При рассмотрении предыдущих примеров описаны функции, которые помогают решить поставленную задачу. Для двух данных чисел можно найти наибольший общий делитель и проверить, равен ли он нулю. Если наибольший общий делитель равен нулю, то числа являются взаимно простыми (листинг 8.3, а).

Листинг 8.3, а. Взаимно простые числа

```
<HTML>
  <HEAD>
    <TITLE>Взаимно простые числа</TITLE>
    <script language="JavaScript">
    <!-- //
      function nod(n,m)
       { var p=n%m
         while (p!=0)
          { n=m
            m=p
            p=n%m
          }
         return m
       }
```

```
      function numsim(n,m)
       { return (nod(n,m)==1) }
      function ans(n,m)
       { if (numsim(n,m))
          { form1.res.value='Да'}
         else
          { form1.res.value='Нет'}
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    Являются ли заданные натуральные числа взаимно простыми?
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить"
             onClick="ans(form1.num1.value,form1.num2.value)"><br>
      Числа взаимно-просты?  <input type="text" name="res" size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

В трех предыдущих примерах при решении задачи использовалась функция `nod`. В каждом из документов приводилось ее описание. Можно описание функции `nod` вынести во внешний файл, а там, где требуется ее использование, подключать внешний файл к документу. Пусть в файле nnod.js содержится следующее описание функции `nod`:

```
function nod(n,m)
 { var p=n%m
   while (p!=0)
    { n=m
      m=p
      p=n%m
    }
   return m
 }
```

Для того чтобы данный файл связать с документами, содержащими вызов функции nod, требуется указать имя внешнего файла в качестве значения параметра src тега <script>. Документ, использующий внешний файл для рассматриваемой задачи, представлен в листинге 8.3, б.

Листинг 8.3, б. Описание функции nod во внешнем файле

```
<HTML>
  <HEAD>
    <TITLE>Наибольший общий делитель двух чисел</TITLE>
    <script language="JavaScript" src=nnod.js>
    <!--//
      document.alert("Проверьте файл со сценарием")
    //-->
    </script>
  </HEAD>
  <BODY>
    Наибольший общий делитель двух заданных чисел
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><hr>
      <input type="button" value="Вычислить"
      onClick="form1.res.value=nod(form1.num1.value,form1.num2.value)">
      <br>
      Наибольший общий делитель  <input type="text" name="res"
                                      size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Строка <script language="JavaScript" src=nnod.js> обеспечивает подключение внешнего файла к документу.

При решении задачи нахождения наименьшего общего кратного использовали две функции nod и nok. Пусть функция nod хранится во внешнем файле, а функция nok — в том же документе, в котором производится ее вызов. При описании первого тега <script> указывался источник с помощью значения параметра src, второй тег содержал описание функции nok. HTML-код документа представлен в листинге 8.3, в.

Листинг 8.3, в. Наименьшее общее кратное. Функция nod во внешнем файле

```
<HTML>
  <HEAD>
    <TITLE>Наименьшее общее кратное двух натуральных чисел</TITLE>
    <script language="JavaScript" src=nnod.js>
    <!-- //
      document.alert("Проверьте файл со сценарием")
    //-->
    </script>
    <script language="JavaScript">
    <!-- //
      function nok(n,m)
       { return n*m/ nod(n,m) }
    //-->
    </script>
  </HEAD>
  <BODY>
    Наименьшее общее кратное двух заданных натуральных чисел
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить"
      onClick="form1.res.value=nok(form1.num1.value,form1.num2.value)">
      <br>
      Наименьшее общее кратное  <input type="text" name="res"
                                      size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

При определении, являются ли два числа взаимно простыми, используются две функции, хранящиеся в отдельных внешних файлах. Для каждой из функций в документе используется свой тег `<script>`, в котором задаются имена внешних файлов (листинг 8.3, г).

Листинг 8.3, г. Определение взаимно простых чисел. Java Script-функции хранятся во внешних файлах

```
<HTML>
  <HEAD>
    <TITLE>Взаимно простые числа</TITLE>
    <script language="JavaScript" src=nnod.js>
    <!-- //
      document.alert("Проверьте файл со сценарием")
    //-->
    </script>
    <script language="JavaScript" src=nnumsim.js>
    <!-- //
      document.alert("Проверьте файл со сценарием")
    //-->
    </script>
  </HEAD>
  <BODY>
    Являются ли заданные натуральные числа взаимно простыми?
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить"
             onClick="ans(form1.num1.value,form1.num2.value)"><br>
      Числа взаимно просты?   <input type="text" name="res" size="8"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Принятие решения о простом и составном числе

Напишем программу, которая определяет, является ли заданное число простым или составным.

В функции `simple` (листинг 8.4) для того, чтобы выяснить, является ли число `n` простым, проверяется остаток от деления `n` на `i`. Переменная `i` может изменяться от 2 до $r = n/2$. Если при каком-либо значении остаток от деления `n` на `i` равен нулю, то отсюда следует, что `i` — делитель `n`, а, следова-

тельно, число n — составное. Выполнение цикла тогда следует прекратить. С этой целью переменной p будет присвоено значение false. Если же переменная i "пробежала" все значения от 2 до r, и при каждом значении i остаток от деления n на i был отличен от нуля, то исполнение цикла завершится, как только значение i станет больше r. Значение переменной p не изменится. В этом случае n — простое число.

Листинг 8.4. Число простое или составное

```
<HTML>
  <HEAD>
    <TITLE>Число простое или составное</TITLE>
    <script language="JavaScript">
    <!-- //
      function simple(obj)
       { var n=obj.num.value
         var p = True
         if (n > 3)
           { var i=2;
             var r=n/2
             while ((i<=r) && p)
               { p=(n%i!=0);  i +=1}
           }
         if (p) obj.result.value="простое"
         else obj.result.value="составное"
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    Является введенное число простым или составным?
    <FORM name="form1">
      <input type="text" name="num" size="8">
      <input type="button" value="Определить" onClick="simple(form1)">
      <input type="text" name="result" size="10"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Функцию `simple` можно улучшить, если делители числа искать из интервала $\left[2, \sqrt{n}\right]$. Кроме того, имеет смысл искать делители только для нечетных чисел.

Числа-близнецы

Напишем программу, определяющую, являются ли два заданных натуральных числа числами-близнецами.

Два натуральных числа называются *близнецами*, если они оба просты и разница между ними равна 2, например, числа 11 и 13 и 17 и 19 являются близнецами. При решении данной задачи воспользуемся функцией `simple`, определяющей, является ли число простым. Функция будет иметь один параметр-число и выдавать логический результат `true`, если число простое, и `false`, если число составное. Функция `numeq` для заданных двух чисел проверяет, являются ли числа простыми и верно ли, что разность между числами равна 2. Результат работы функции `numeq` служит для формирования ответа, который помещается в соответствующее поле формы (листинг 8.5).

Листинг 8.5. Числа-близнецы

```
<HTML>
  <HEAD>
    <TITLE>Числа-близнецы</TITLE>
    <script language="JavaScript">
    <!-- //
      function simple(n)
       { var p=true
         if (n>3)
         { var i=2;
           var r=n/2
           while ((i<=r) && p)
            { p=(n%i!=0);  i += 1}
         }
         return p
       }
      function numeq (n,m)
       { return (simple(n) && simple(m) && Math.abs(n-m)==2) }
      function ans(n,m)
       { if (numeq(n,m))
```

```
          { form1.res.value="близнецы" }
         else
          { form1.res.value="не являются близнецами" }
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    Являются ли заданные натуральные числа близнецами?
    <FORM name="form1">
      Введите число <input type="text" name="num1" size="8"><br>
      Введите число <input type="text" name="num2" size="8"><br>
      <input type="button" value="Вычислить"
             onClick="ans(form1.num1.value,form1.num2.value)"><br>
      Заданные числа <input type="text" name="res" size="25"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Числа Фибоначчи

Напишем программу, определяющую, является ли заданное число числом Фибоначчи.

Элементы числовой последовательности 1, 1, 2, 3, 5, 8, 13, 21, ..., в которой первые два элемента равны 1, а каждый следующий элемент представляет собой сумму двух предыдущих, называются *числами Фибоначчи*. В функции `fib` в результате работы цикла формируется очередное число последовательности до тех пор, пока сформированное число не станет больше или равным заданному числу. После выполнения цикла определяется, принадлежит ли заданное число последовательности чисел Фибоначчи (листинг 8.6).

Листинг 8.6. Числа Фибоначчи

```
<HTML>
  <HEAD>
    <TITLE> </TITLE>
    <script language="JavaScript">
    <!-- //
      function fib(obj)
```

```
      { var num=obj.data.value
        var f1=1
        var f2=1
        var f3=1
        while (num>f3)
         { f3=f1+f2;    f1=f2;    f2=f3 }
        if (num==f3) obj.result.value="Да"
        else obj.result.value="Нет"
      }
    //-->
    </script>
  </HEAD>
  <BODY>
    Является ли введенное значение числом Фибоначчи?
    <FORM name="form1">
      <input type="text" name="data" size="8">
      <input type="button" value="Определить" onClick="fib(form1)">
      <input type="text" name="result" size="5"><hr>
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Решение уравнения методом итераций

Необходимо написать программу, которая находит корень уравнения $\sin(x^2+1)$ методом итераций при заданном начальном приближении t и точности *eps*.

Методом итераций решается уравнение вида

$$x = f(x), \left|f(x) < 1\right|.$$

Последовательные приближения к корню задаются по формуле $x_r = f(x_{k-1})$, $k = 1, 2, \ldots, x_0 = t$. Если $\left|x_k - x_{k-1}\right| < eps$, то значение x_{k-1} можно считать приближенным корнем уравнения, т. к. выполнено:

$x_r = f(x_{k-1})$ и $\left|f(x_{k-1}) - x_{k-1}\right| < eps$.

Два последних вычисленных значения отличаются по модулю на величину, меньшую заданной точности *eps*.

В функции `iter` используются переменные `x` и `y` для хранения двух последовательных значений (листинг 8.7).

Листинг 8.7. Решение уравнения методом итераций

```
<HTML>
  <HEAD>
    <TITLE>Решение уравнения методом итераций</TITLE>
    <script language="JavaScript">
    <!-- //
      function iter(obj)
       { var t=obj.prib.value
         var eps=obj.teps.value
         var x=t
         var y=Math.sin(x*x+1)
         while (Math.abs (y-x)>=eps)
           { x=y;  y= Math.sin(x*x+1) }
         obj.result.value=y
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    Решение уравнения sin(x*x+1)=0 методом итераций
    <FORM name="form1">
      Введите начальное приближение  <input type="text" name="prib"
                                          size="8"><br>
      Введите требуемую точность     <input type="text" name="teps"
                                          size="8"><br>
      <input type="button" value="Найти корень" onClick="iter(form)">
      Корень уравнения <input type="text" name="result" size="18"><hr>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
<HTML>
```

Свойства пар натуральных чисел

Напишем сценарий, который позволяет для двух заданных натуральных чисел найти наибольший общий делитель или наименьшее общее кратное, выяснить, являются ли числа взаимно простыми или близнецами. Свойства, которые требуется определить, задаются с помощью флажка так, как показано на рис. 8.2.

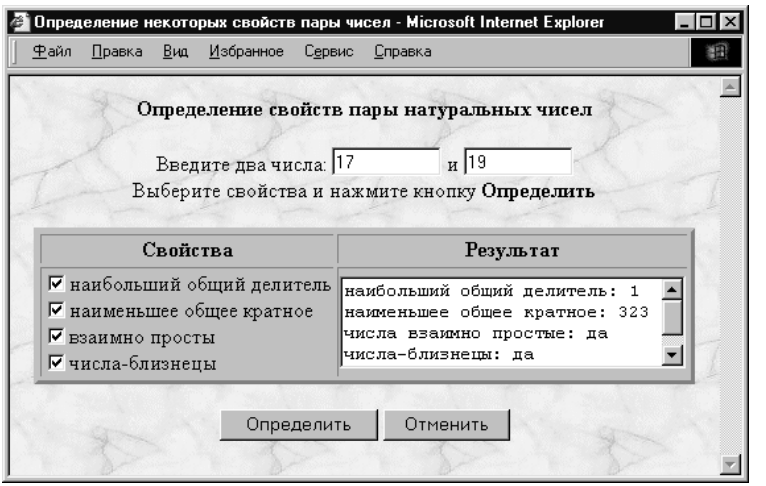


Рис. 8.2. Свойства пар натуральных чисел

Две функции: вычисляющая наименьшее общее кратное и проверяющая, являются ли два числа взаимно простыми, используют функцию определения наибольшего общего делителя. Функция принятия решения, являются ли два значения числами-близнецами, использует функцию определения простого числа. Все функции поместим в отдельный файл. Этот файл не будет содержать тегов HTML и должен иметь расширение js. В нашем случае во внешнем файле хранятся следующие описания переменных и функций:

```
var s1="наибольший общий делитель: "
var s2="наименьшее общее кратное: "
var s3="числа взаимно простые: "
var s4="числа-близнецы: "
function nod(n,m)
 { var p = n%m
   while (p !=0)
     { n=m
       m=p
       p=n%m
     }
   return m
 }
function nok(n,m)
 { return n*m/nod(n,m) }
function simple(n)
 { var p=true
     if (n>3)
```

```
      { var i=2;
        var r=n/2
        while ((i<=r) && p)
          { p=(n%i != 0);  i+=1}
      }
    return p
 }
function numeq(n,m)
 { var r="нет"
    if ( simple(n) && simple(m) && Math.abs(n-m)==2 )
      r="да"
    return r
 }
function numsim(n,m)
 { var r= "нет"
    if (nod(n,m)==1)
      r="да"
    return r
 }
function grant(obj)
{ var s=""
  var n=obj.num1.value
  var m=obj.num2.value
  if ((obj.elements[2]).checked) {s=s+s1+nod(n,m)+"\r\n"; ans(s)}
  if ((obj.elements[3]).checked) {s=s+s2+nok(n,m)+"\r\n"; ans(s)}
  if ((obj.elements[4]).checked) {s=s+s3+numsim(n,m)+"\r\n"; ans(s)}
  if ((obj.elements[5]).checked) {s=s+s4+numeq(n,m)+"\r\n"; ans(s)}
}
function ans(s)
 { document.form1.res.value =s }
```

Для того чтобы этот файл связать с документом, содержащим вызов функций, описанных во внешнем файле, требуется имя файла задать в качестве значения параметра `src` тега `<script>`. Документ, использующий внешний файл для рассматриваемой задачи, представлен в листинге 8.8.

Листинг 8.8. Определение некоторых свойств пары натуральных чисел

```
<HTML>
  <HEAD>
    <TITLE>Определение некоторых свойств пары чисел</TITLE>
    <script language="JavaScript" src="8ex8func.js">
```

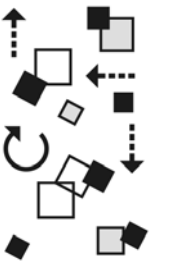
```
    <!--
      document.alert("Проверьте файл со сценариями!")
    //-->
    </script>
  </HEAD>
  <BODY background="fon3.gif">
    <H4 align=center>Определение свойств пары натуральных чисел </H4>
    <CENTER>
    <FORM name="form1">
      Введите два числа:
      <input type="text" name="num1" size=10>
      и <input type="text" name="num2" size=10><br>
      Выберите свойства и нажмите кнопку <B>Определить</B><P>
      <TABLE border=3 bgcolor=silver align=center>
        <TR><TH>Свойства</TH><TH>Результат</TH></TR>
        <TR>
          <TD>
            <input type="checkbox" name="n" value=1>
               наибольший общий делитель<br>
            <input type="checkbox" name="n" value=2>
               наименьшее общее кратное<br>
            <input type="checkbox" name="n" value=3>взаимно просты<br>
            <input type="checkbox" name="n" value=4>числа-близнецы<br>
          </TD>
          <TD><textarea name=res cols=30 rows=4></textarea><br></TD>
        </TR>
      </TABLE><P>
      <input type="button" value=Определить onClick="grant(form1)">
      <input type="reset"  value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите программу, которая "переворачивает" заданное натуральное число.
2. Напишите сценарий, который определяет все числа-палиндромы[1] из заданного пользователем диапазона чисел.

[1] Палиндром — число или слово, которые читаются одинаково как справа налево, так и слева направо. Например, потоп или 454. — *Ред.*

Глава 9



Оператор цикла арифметического типа

Если число повторений заранее известно, то можно воспользоваться следующим оператором цикла, который часто называют оператором *цикла арифметического типа*. Синтаксис этого оператора таков:

```
for (A; B; I){S}
```

Выражение `A` служит для инициализации параметра цикла и вычисляется один раз в начале выполнения цикла. Выражение `B` (условие продолжения) управляет работой цикла. Если значение выражения ложно, то выполнение цикла завершается, если истинно, то выполняется оператор `S`, составляющий тело цикла. Выражение `I` служит для изменения значения параметра цикла. После выполнения тела цикла `S` вычисляется значение выражения `I`, затем опять вычисляется значение выражения `B` и т. д. Цикл может прекратить свою работу в результате выполнения оператора `break` в теле цикла. Опишем функцию `sumdel`, которая находит сумму делителей числа `n`, не считая самого числа.

```
function sumdel (n)
 { var s=1;
   for (var i=2; i <= n/2; i++)
     { if (n % i == 0)  s += i }
   return s
 }
```

Параметр цикла `i` описывается с помощью оператора `var i=2`. Условие продолжения выполнения цикла `i<=n/2`. Все делители натуральных чисел находятся в интервале [1; *n*]. Параметр цикла `i` увеличивается на 1 при выполнении `i++`. Тело цикла состоит из условного оператора. Если очередное число является делителем, то оно добавляется к переменной `s`, служащей для "накапливания" суммы делителей числа `n`. До цикла переменной `s` присваивается значение 1, т. к. 1 — делитель любого натурального числа.

Совершенные числа

Напишем программу, определяющую, является ли заданное число *n* совершенным.

Совершенным называется число *n*, равное сумме своих делителей, не считая самого числа. Например, число 6 является совершенным, т. к. верно 6 = 1 + 2 + 3, где 1, 2, 3 — делители числа 6. Число 28 также является совершенным, справедливо равенство 28 = 1 + 2 + 4 + 7 + 14. При решении задачи воспользуемся только что описанной функцией `sumdel` (листинг 9.1).

Листинг 9.1. Итерационные методы. Совершенные числа

```
<HTML>
  <HEAD>
    <TITLE>Итерационные методы. Совершенные числа</TITLE>
    <script language="JavaScript">
    <!-- //
      function  sumdel(n)
       { var s=1;
         for (var i=2;  i<=n/2; i++)
          { if (n % i == 0)    s += i }
         return s
       }
      function sov(obj)
       { var n=obj.numb.value;
         var s=""
         if (n==sumdel(n)) s="совершенное"
         else s="не является совершенным"
         return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P> Итерационные методы. Совершенные числа</P>
    <FORM name="form0">
      Введите натуральное число: <input type="text" size=8 name="numb">
      <input type="button" value=Выполнить
                  onClick="this.form.res.value=sov(form0)"><hr>
```

```
      Данное число: <input type="text" size=24 name="res"><hr>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Обратите внимание на значение параметра обработки события. В данном случае это оператор присваивания, в правой части которого вызов функции `sov`.

Дружественные числа

Необходимо написать сценарий, который для двух заданных натуральных чисел *n* и *m* определяет, являются ли они дружественными. Числа называются *дружественными*, если каждое из них равно сумме делителей другого, не считая самого числа.

Очевидно, что если число *n* является совершенным, то пара (*n*, *n*) является дружественной. Интересно рассмотреть случаи, когда числа *n* и *m* различны.

При решении задачи удобно использовать функцию `sumdel`, которая для каждого натурального числа находит сумму его делителей без самого числа. Так как функция `sumdel` используется для решения разных задач, то разумно ее описание вынести во внешний файл. Пусть fsumdel.js — имя файла, в котором хранится описание функции `sumdel` на языке JavaScript. Описание функции `dr`, определяющей, являются ли два числа дружественными, может быть таким:

```
function  dr (n,m)
 { var s = ""
   if (n==sumdel(m) && m==sumdel(n)) s="дружественные"
   else  s="не являются дружественными"
   return s
 }
```

Полностью создать программу читателю предлагается в качестве упражнения.

Нахождение дружественных чисел из заданного диапазона

Напишем сценарий, позволяющий найти все числа из заданного диапазона, которые образуют пару дружественных чисел.

Параметр цикла `j` "пробегает" все числа от `a` до `b` из заданного диапазона. Для каждого числа `j` находится число, являющееся суммой делителей `j`. Если это число попадает в рассматриваемый диапазон, то рассчитывается его

сумма делителей и сравнивается с числом `j`. В текстовой переменной `st` формируется ответ в виде пар дружественных чисел. Сформированная строка `st` выводится в текстовое поле формы, предназначенное для отображения результата, как показано на рис. 9.1.

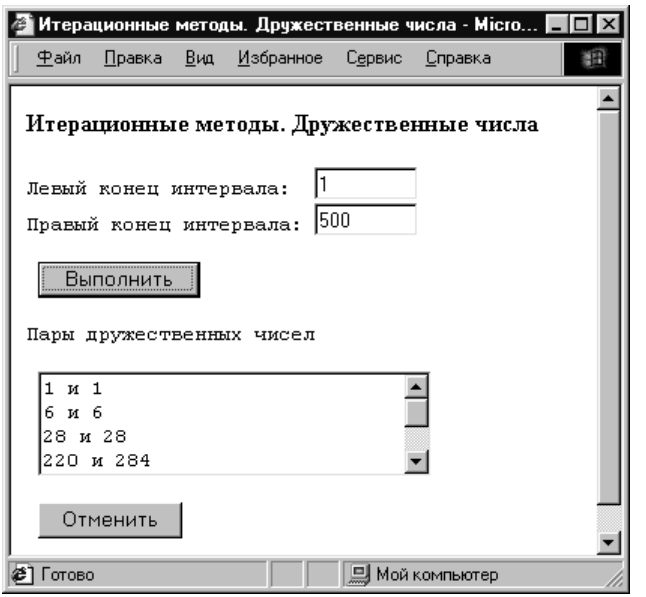


Рис. 9.1. Дружественные числа из заданного диапазона

HTML-код документа представлен в листинге 9.2.

Листинг 9.2. Итерационные методы. Дружественные числа

```
<HTML>
  <HEAD>
    <TITLE> Итерационные методы. Дружественные числа</TITLE>
    <script LANGUAG="JavaScript">
    <!-- //
      function  sumdel (number)
       { var s=1;
         var i=1;
         for (i=2; i <= number/2; i++)
          { if ( number % i == 0)    s += i }
         return s
       }
```

```
      var st=""
      function dr(obj)
       { var a= obj.num1.value
         var b= obj.num2.value
         var k
         for (var j=a; j <=b; j++)
          { k=sumdel (j)
            if (k >=a && k <= b)
             { if (sumdel (k) == j)
               { st+= j+" и " +k+"\r\n" }
             }
          }
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4> Итерационные методы. Дружественные числа</H4>
    <FORM name="form0">
    <pre>
Левый конец интервала:  <input type="text" size=8 name="num1">
Правый конец интервала: <input type="text" size=8 name="num2"><br>
 <input type="button" value=Выполнить
      onClick="dr(form0); if (st==''){st='нет'};
this.form.res.value=st"><br>
Пары дружественных чисел<br>
 <textarea  cols=30 rows=4  name="res"> </textarea><P>
 <input type="reset" value=Отменить onClick="st=''">
    </pre>
    </FORM>
  </BODY>
</HTML>
```

При выполнения сценария в качестве ответа поступают все совершенные числа, и каждая из пар дружественных чисел встречается два раза. Для того чтобы исключить такие ситуации, изменим функцию `dr`, добавив в нее дополнительную проверку: `k>j`. Теперь в случае, когда задан интервал от 1 до 500, будет выведена лишь одна дружественная пара: 220 и 284.

Вычисление факториала: вариант 1

Напишем программу, определяющую значение факториала заданного натурального числа.

Напомним, что $n! = 1 \times 2 \times \ldots \times (n-1) \times n$. Будем использовать переменную `f` для накапливания сомножителей. До выполнения цикла переменной `f` присваивается значение 1, затем в теле цикла полученное значение `f` умножается на `i`. После того как параметр цикла `i` "пробежит" все значения от 2 до n в переменной `f` будет вычислено значение $n!$.

HTML-код представлен в листинге 9.3.

Листинг 9.3. Итерационные методы. Вычисление факториала

```
<HTML>
  <HEAD>
    <TITLE>Вычисление факториала</TITLE>
    <script language="JavaScript">
    <!-- //
      function  fact1 (obj )
       { n=obj.num.value
         var f=1
         for (var i=2; i <= n; i++)  {f = f*i}
         obj.res.value=f
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P> Итерационные методы. Вычисление факториала</P>
    <FORM name="form1">
      Введите натуральное: <input type="text" size=8 name="num">
      Вычисленное значение: <input type="text" size=8 name="res"><hr>
      <input type="button" value=Выполнить onClick="fact1(form1)"><hr>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Вычисление факториала: вариант 2

Приведем еще один пример вычисления значения факториала. Опишем функцию `fact1`, в которой организовано накапливание значения факториала способом, отличным от того, что был рассмотрен в предыдущем примере. Обратите внимание на оператор цикла:

```
function  fact2 (n)
 { var f=1
   for (var i=n; i > 1; i--)  {f = f*i};
   return f
 }
```

В функцию `fact2` передается значение параметра (а не имя формы, как в предыдущем примере), полученный после выполнения функции результат записывается в соответствующее поле формы. Значение параметра обработки события получается с помощью оператора присваивания, при помощи которого осуществляется запись в поле результата. Полностью программа выглядит так, как представлено в листинге 9.4.

Листинг 9.4. Вычисление *n*!

```
<HTML>
  <HEAD>
    <TITLE>Вычисление n!</TITLE>
    <script language="JavaScript">
    <!-- //
      function  fact2 (n)
       { var f=1
         for (var i=n; i >= 1; i-=1)
          { f = f*i }
         return f
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P> Итерационные методы. Вычисление факториала n!</P>
    <FORM name="form1">
      Введите натуральное: <input type="text" size=8 name="num">
      Вычисленное значение: <input type="text" size=8 name="res"><hr>
```

```
      <input type="button" value=Выполнить
             onClick="form1.res.value=fact2(form1.num.value)"><hr>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Вычисление *n*!!

Напишем сценарий, который по заданному натуральному числу *n* вычисляет *n*!!

Напомним, что *n*!! = 1×3×...×*n*, если *n* нечетно, и *n*!! = 2×4×...×*n*, если *n* четно. Опишем функцию `fact3`, решающую задачу:

```
function fact3 (n)
 { var f=1
   for (var i=n; i>1; i-=2)   {f = f*i};
   return f
 }
```

Обратите внимание на выражение, в результате выполнения которого должно меняться значение параметра цикла. На каждом шаге итерации значение параметра уменьшается на 2. Полностью сценарий представлен в листинге 9.5.

Листинг 9.5. Вычисление значения *n*!!

```
<HTML>
  <HEAD>
    <TITLE>Вычисление значения n!!</TITLE>
    <script language="JavaScript">
    <!-- //
      function  fact3(n)
        { var f=1
          for (var i=n; i >= 1; i-=2)  {f = f*i}
          return f
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P> Итерационные методы. Вычисление n!!</P>
    <FORM name="form1">
```

```
      Введите натуральное: <input type="text" size=10 name="num">
      Вычисленное значение: <input type="text" size=10 name="res"><hr>
      <input type="button" value=Выполнить
             onClick="form1.res.value=fact3(form1.num.value)"><hr>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Движение точки вдоль ломаной

Точка движется по линии, изображенной на рис. 9.2, перемещаясь за один шаг на одну клетку. Известны целочисленные координаты (x, y) точки в некоторый момент времени. Напишем программу, определяющую координаты точки через заданное число шагов.

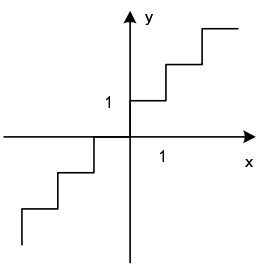


Рис. 9.2. Траектория движения точки вдоль ломаной

Проведем прямые, соединяющие точки ломаной. Если значения координат x и y совпадают, следовательно, точка лежит на прямой, описываемой уравнением $y = x$, и следующий шаг вдоль ломаной должен быть вправо. Если же значения координат x и y связаны соотношением $y = x-1$, то точка вдоль ломаной должна сместиться вверх. Мы описали только один шаг передвижения. Чтобы узнать координаты точки через несколько шагов, в программе используется цикл. В функции `step` вычисляются и записываются в нужные поля формы конечные координаты (листинг 9.6).

Листинг 9.6. Движение точки вдоль ломаной

```
<HTML>
  <HEAD>
    <TITLE>Движение точки вдоль ломаной</TITLE>
    <script language="JavaScript">
    <!-- //
```

```
    function step(obj)
     { var x=Number(obj.x.value)
       var y=Number(obj.y.value)
       var n=obj.n.value
       for (var i=1; i <= n; i++)
        { if (y==x) x += 1
          else if (y==x-1) y+=1 }
       obj.newx.value=x
       obj.newy.value=y
     }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Движение точки вдоль ломаной</P>
    <pre>
<FORM name="form1">
Введите координату по оси x:   <input type="text" size=8 name="x"><hr>
Введите координату по оси y:   <input type="text" size=8 name="y"><hr>
Введите число шагов:           <input type="text" size=8 name="n"><hr>
 <input type="button" value=Выполнить onClick="step(form1)"><hr>
<P>Координаты точки в конце шага</P>
 По оси x: <input type="text" size=8 name="newx"><hr>
 По оси y: <input type="text" size=8 name="newy"><hr>
 <input type="reset" value=Отменить>
</FORM></pre>
  </BODY>
</HTML>
```

Вычисление суммы чисел, кратных 7

Напишите функцию, которая определяет сумму всех чисел, кратных 7 в заданном интервале [*m*; *n*].

В качестве начального значения параметра цикла в функции `step` выбирается ближайшее целое к значению правого конца промежутка. На каждом шаге выполнения тела цикла значение параметра уменьшается на 7. Цикл завершает свою работу, когда значение параметра цикла меньше значения левого промежутка. Полностью программа описана в листинге 9.7.

Листинг 9.7. Сумма чисел, кратных 7, в заданном интервале

```
<HTML>
  <HEAD>
    <TITLE>Сумма чисел, кратных 7, в заданном интервале </TITLE>
    <script language="JavaScript">
    <!-- //
      function  sum7(m,n)
       { var s=0
         for (var i=n-n%7; i >= m; i-=7)
          { s += i }
         return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Определение суммы чисел, кратных 7, в заданном интервале</P>
    <pre>
<FORM name="form1">
Введите левую границу интервала:   <input type="text" size=8
name="lef"><hr>
Введите правую границу интервала:  <input type="text" size=8
name="rig"><hr>
Вычисленная сумма:               <input type="text" size=8
name="res"><hr>
 <input type="button" value=Выполнить
onClick="form1.res.value=sum7(form1.lef.value,form1.rig.value)"><hr>
 <input type="reset" value=Отменить>
</FORM>
</pre></BODY></HTML>
```

Сумма элементов последовательности

Напишем программу, которая по заданному целому значению n и вещественному x находит сумму $s = \sum_{i=1}^{n} \frac{x^i}{i!}$. Предположим, что уже найдена сумма первых $n-1$ элементов, последнее слагаемое обозначим через a. Следующий элемент последовательности определяется по формуле $a \times x / n$.

HTML-код представлен в листинге 9.8.

Листинг 9.8. Сумма элементов последовательности

```
<HTML>
  <HEAD>
    <TITLE>Сумма элементов последовательности</TITLE>
    <script language="JavaScript">
    <!-- //
      function sum(x,n)
       { var s=0
         var a=1
         for (var i=1; i <= n; i++)
          { a=a*x/i;  s += a }
            return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Сумма элементов последовательности </P>
<pre>
<FORM name="form1">
Введите значение x:       <input type="text" size=8 name="lef"><hr>
Введите число элементов:  <input type="text" size=8 name="rig"><hr>
Вычисленная сумма:        <input type="text" size=12 name="res"><hr>
 <input type="button" value=Выполнить

onClick="form1.res.value=sum(form1.lef.value,form1.rig.value)"><hr>
 <input type="reset" value=Отменить>
</FORM>
</pre>
  </BODY>
</HTML>
```

Выбор и размещение изображений

Напишем сценарий, в результате выполнения которого осуществляется выбор изображений и размещение их в заданных окнах в порядке предпочтения.

После щелчка мышью по названию изображения в левой части экрана в свободном окне появляется соответствующее изображение. Если все места заняты как на рис. 9.3, а пользователь выбирает название изображение, то происходит замена в текущем окне.

Рис. 9.3. Выбор изображения из заданного списка

Названия изображений задаются с помощью тега `<select>`. При выборе варианта соответствующее изображение загружается в свободную область просмотра. Значение глобальной переменной `k` определяет номер свободной области. Если заполнены все области просмотра, то очередное изображение опять помещается в первую область, замещая находящееся там изображение. Переменная `len` используется при вычислении числа названий изображений, т. е. элементов в теге `<select>`.

Функция `ref` очищает все окна. Сценарий решения задачи приведен в листинге 9.9.

Листинг 9.9. Выбор и размещение изображений

```
<HTML>
  <HEAD>
    <TITLE>Выбор и размещение изображений</TITLE>
    <script>
    <!--
      var k=0
      var len
```

```
    function ch_pict(num)
     { len=((document.form1).elements[0]).length-1
       var m=num*100
       document.images[k].src="picture"+num+".gif"
       document.forms[0].elements[k+1].value=m
       if (k < len-1) k+=1
       else k=0
     }
    function ref()
     { len=((document.form1).elements[0]).length-1
       k=0
       for (var i=0; i < len; i++)
        { document.images[i].src="picture0.gif"
          document.forms[0].elements[i+1].value=''}
     }
  //-->
  </script>
</HEAD>
<BODY bgcolor="#eaf5ef">
  <H4 align=center>Выберите рисунки в порядке предпочтения</H4>
  <FORM name="form1">
    <TABLE border=1>
     <TR>
       <TD>
         <select name="pict" size="7"
                 onChange="ch_pict(form1.pict.value)">
          <option value="0">Нет        </option>
          <option value="1">Дом        </option>
          <option value="2">Книги      </option>
          <option value="3">Письмо     </option>
          <option value="4">Краски     </option>
          <option value="5">Вопрос     </option>
          <option value="6">Дерево     </option>
        </select>
      </TD>
      <TD align=center><img src="picture0.gif" width=70></TD>
      <TD align=center><img src="picture0.gif" width=70></TD>
      <TD align=center><img src="picture0.gif" width=70></TD>
      <TD align=center><img src="picture0.gif" width=70></TD>
      <TD align=center><img src="picture0.gif" width=70></TD>
```

```
        <TD align=center><img src="picture0.gif" width=70></TD>
      </TR>
      <TR>
        <TD width=50>стоимость</TD>
        <TD width=50><input type="text" name="res1" size=8></TD>
        <TD width=50><input type="text" name="res2" size=8></TD>
        <TD width=50><input type="text" name="res3" size=8></TD>
        <TD width=50><input type="text" name="res4" size=8></TD>
        <TD width=50><input type="text" name="res5" size=8></TD>
        <TD width=50><input type="text" name="res6" size=8></TD>
      </TR>
    </TABLE><br>
    <input type="button" value="Обновить" onClick="ref()">
  </BODY>
</HTML>
```

Выбор критериев качества чтения лекций

Пусть качество чтения лекций оценивается по различным критериям. Требуется написать сценарий, который обеспечивает пользователю выбор пяти наиболее важных с точки зрения пользователя критериев из заданного списка.

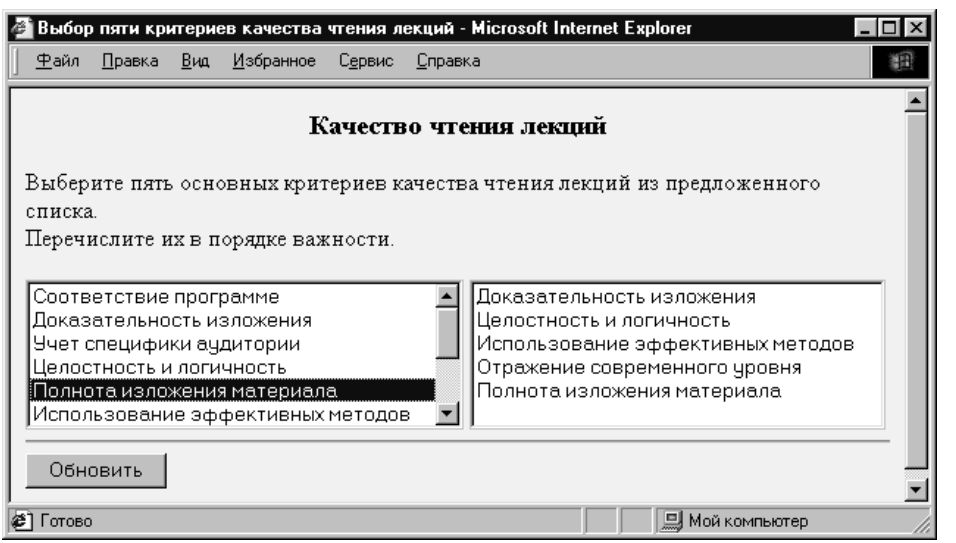


Рис. 9.4. Качество чтения лекций

Критерии оценки качества задаются с помощью тега <select>. Как только пользователь выбирает очередное значение качества, оно переносится в правое поле, как на рис. 9.4.

Разрешено выбрать не более пяти значений качеств, при попытке указать шестое выводится предупреждающее сообщение. Кроме того, в сценарии предусматривается, что все выбранные качества различны. Если делается попытка повторно выбрать некоторый критерий, то пользователь об этом будет предупрежден. HTML-код со сценарием представлен в листинге 9.10.

Листинг 9.10. Выбор пяти критериев качества чтения лекций

```
<HTML>
  <HEAD>
    <TITLE>Выбор пяти критериев качества чтения лекций</TITLE>
    <script language="JavaScript">
    <!-- //
      var n=0
      function mov(obj,m)
       { var s=((obj.elements[0])[m]).text
         var s1
         var p= false
         if (n<=(obj.elements[1]).length-1)
          { for (var i=0; i <= n-1; i++)
             { s1= ((obj.elements[1])[i]).text
               if (s==s1)
              { alert ("Вы уже выбирали этот критерий"); p= TRue; break }
             }
            if(!p)
          ((obj.elements[1])[n]).text=((obj.elements[0])[m]).text; n= n+1
          }
         else alert ("Вы уже выбрали необходимое число критериев")
       }
      function  ref(obj)
       { n=0
         for (var i=0; i <= (obj.elements[1]).length-1; i++)
          { ((obj.elements[1])[i]).text="                          "}
       }
    //-->
    </script>
  </HEAD>
```

```
  <BODY bgcolor="#eaf5ef" onLoad="ref(form1)">
    <FORM name="form1">
      <H3 align=center>Качество чтения лекций</H3>
      <P>Выберите пять основных критериев качества чтения
         лекций из предложенного списка.
      <br>Перечислите их  в порядке важности.</p>
      <select name="formdata" size=6
              onChange=mov(form1,form1.formdata.value)>
        <option value=0>Соответствие программе
        <option value=1>Доказательность изложения
        <option value=2>Учет специфики аудитории
        <option value=3>Целостность и логичность
        <option value=4>Полнота изложения материала
        <option value=5>Использование эффективных методов
        <option value=6>Отражение современного уровня
        <option value=7>Владение терминологией
        <option value=8>Полнота использования времени
        <option value=9>Тематическая завершенность
        <option value=10>Использование современных технологий
       </select>
       <select name="formres" size=6>
        <option   value="n1">
        <option   value="n2">
        <option  value="n3">
        <option  value="n4">
        <option  value="n5">
      </select><hr>
      <input type="reset" value=Обновить onClick="ref(form1)">
    </FORM>
  </BODY>
</HTML>
```

Анкета читателя

Напишем сценарий обработки анкеты читателя. Исследуемые характеристики журнала задаются в левом столбце анкеты. Для работы с анкетой читатель должен выделить характеристику и нажать кнопку **Добавить**. В правом столбце должен появиться соответствующий текст. Если из правого столбца требуется удалить текст, то его надо выделить и нажать кнопку **Удалить**. Ес-

ли в правом столбце нет выделенного текста, то при нажатии кнопки **Удалить** из списка убирается первый элемент. Если правый столбец пустой, то при нажатии кнопки **Удалить** выдается сообщение о том, что список пуст. На рис. 9.5 изображен вид анкеты читателя с уже выбранными характеристиками.

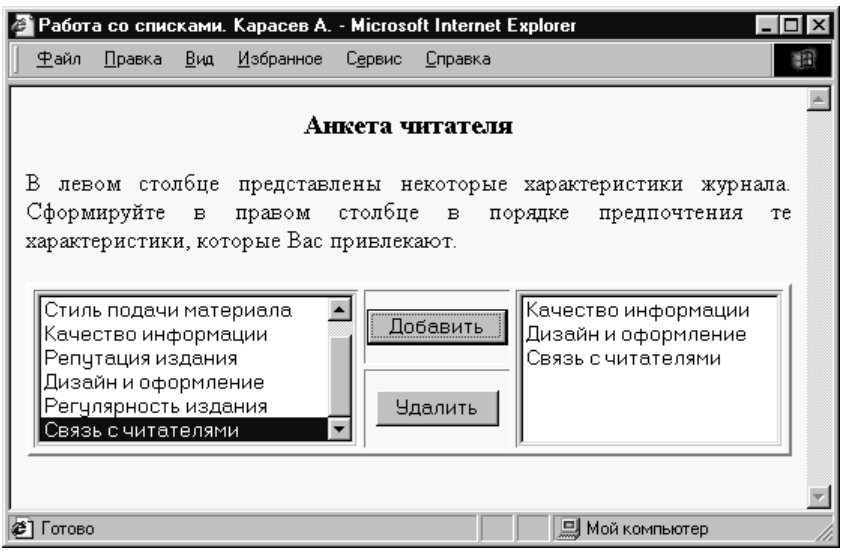


Рис. 9.5. Работа с анкетой читателя

Функция `add` помещает в правый список на свободное место выбранный элемент левого списка. Функция `del` удаляет из второго списка выбранный элемент. При этом с помощью цикла все остальные элементы списка сдвигаются вверх, и изменяется длина второго списка.

Листинг 9.11. Работа со списками: добавление и удаление элементов

```
<HTML>
  <HEAD>
    <TITLE>Работа со списками. Карасев А.</TITLE>
    <script>
    <!--
      function add(ob)
       { ob.List2.length++;
  ob.List2[ob.List2.length-1].text=ob.List1[Number(ob.List1.value)].text;
  ob.List2[ob.List2.length-1].value=ob.List2.length-1}
      function del(ob)
```

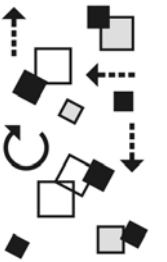
```
      { if (!(ob.List2.length==0))
        { for(i=Number(ob.List2.value);i+1<ob.List2.length;i++)
            ob.List2[i].text=ob.List2[i+1].text;
            ob.List2.length--}
        else
          alert("В списке нет элементов для удаления")
      }
  //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <FORM name=List>
    <h3 align=center>Анкета читателя</h3>
    <P align=justify>
    В левом столбце представлены некоторые характеристики журнала.
    Сформируйте в правом столбце в порядке предпочтения
    те характеристики, которые Вас привлекают.
    </P>
    <TABLE border=2 cellspacing=3>
      <TR>
        <TD rowspan=2>
          <select name=List1 size=6>
            <option value=0>Достоверность информации
            <option value=1>Стиль подачи материала
            <option value=2>Качество информации
            <option value=3>Репутация издания
            <option value=4>Дизайн и оформление
            <option value=5>Регулярность издания
            <option value=6>Связь с читателями</option>
          </select>
        <TD align=middle><input type=button value="Добавить"
                                onClick=add(List)>
        <TD rowspan=2>
          <select name=List2 size=6>
          </select>
        <TR>
          <TD align=middle><input type=button value="Удалить"
                                  onClick=del(List)></TD></TR>
```

```
      </TABLE>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите программу, в которой определяется, является ли номер шестизначного автобусного билета "счастливым", т. е. таким, в котором сумма цифр на нечетных местах равна сумме цифр, расположенных на четных местах.
2. Напишите сценарий, в котором определяется количество "счастливых" шестизначных автобусных билетов, т. е. таких, в номерах которых сумма первых трех цифр равна сумме трех последних.
3. Напишите сценарий, в котором определяется, можно ли натуральное число *n* представить в виде суммы двух квадратов.

Глава 10



Оператор *for...in*

Оператор `for...in` используется для анализа свойств объекта. Синтаксис оператора:

```
for (i in t) {s}
```

где `i` — переменная цикла; `t` — объект; `S` — последовательность операторов.

В результате выполнения оператора цикла производится перебор свойств объекта. Переменная цикла при каждом повторении содержит значение свойства объекта. Количество повторений тела цикла `S` равно числу свойств, определенных для объекта `t`.

Определение свойств элемента формы

Напишем сценарий, с помощью которого можно определить свойства элемента формы "поле ввода многострочного текста".

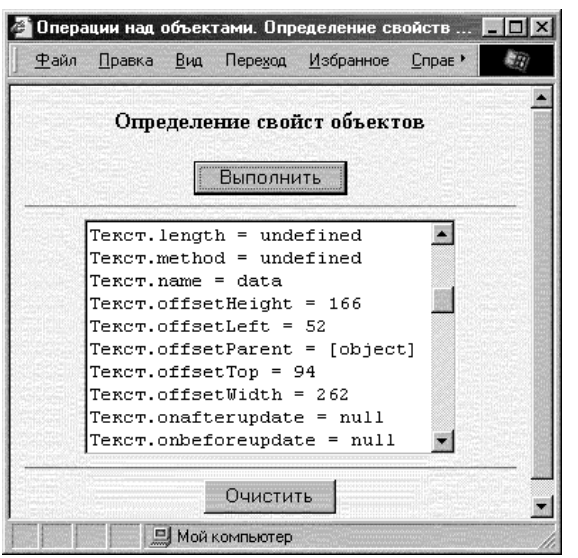


Рис. 10.1. Свойства поля для ввода многострочного текста

Свойства объекта с помощью оператора цикла формируются в строке `result`, затем после просмотра всех свойств значение строки `result` помещается в поле ввода многострочного текста (рис. 10.1).

Сценарий определения свойств текстового поля приведен в листинге 10.1.

Листинг 10.1. Операции над объектами. Свойства текстового поля

```
<HTML>
  <HEAD>
    <TITLE>Операции над объектами. Свойства текстового поля</TITLE>
    <script language="JavaScript">
    <!-- //
      function  propobj (obj)
      { var result = ""
        for (var i in obj)
         { result += obj.data.value +"." +i+ " = " +(obj.data)[i]+"\r\n"}
        result += "\n\r"
        form1.data.value=result
    }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=F8F8FF>
    <CENTER>
    <H4>Определение свойств объектов</H4>
    <FORM name="form1">
    <input type="button" value="Выполнить" onClick="propobj (form1)"><hr>
    <textarea name="data" cols=30 rows=10 id=1>Текст</textarea><hr>
    <input type="reset" value=Очистить>
    </CENTER>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите программу, определяющую все делители заданного натурального числа.
2. Напишите программу, которая выводит первые *n* чисел Фибоначчи.
3. Напишите программу, которая определяет корень уравнения вида $f(x) = 0$ методом Ньютона.
4. Напишите программу вычисления числа размещений из *m* элементов по *n*.
5. Напишите программу, вычисляющую число сочетаний из *m* элементов по *n*.
6. Напишите сценарий, при работе которого вводятся координаты точки на плоскости, и определяется число точек, попадающих в заштрихованную область (рис. 10.2).

Рис. 10.2. Область и точка

7. Напишите сценарии, определяющие свойства элементов: флажка, переключателя, списка, кнопки.
8. Точка движется вдоль ломаной. Напишите сценарий, определяющий координаты точки через заданное число шагов. Вид документа приведен на рис. 10.3.

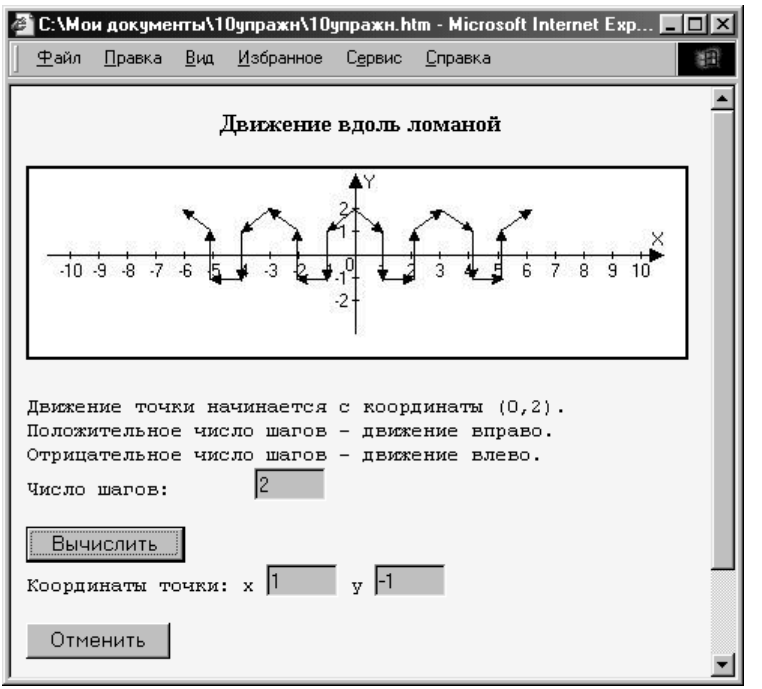
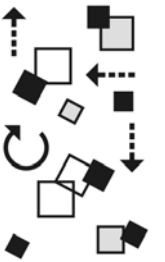


Рис. 10.3. Движение точки в заданном направлении

Глава 11



Представление и обработка дат

Встроенный объект `Data` применяется для представления и обработки даты и времени. Он не имеет свойств, но обладает несколькими методами, позволяющими устанавливать и изменять дату и время. В языке JavaScript дата определяется числом миллисекунд, прошедших с 1 января 1970 года.

Объект `Data` создается оператором `new` с помощью конструктора `Data`. Если в конструкторе отсутствуют параметры, то значением `new Data()` будет текущая дата и время. Значением переменной `my_date1`, определенной следующим образом:

```
var my_data1 = new Data()
```

будет объект, соответствующий текущей дате и времени.

Параметром конструктора `new Data` может быть строка формата "`месяц, день, год часы:минуты:секунды`". Опишем переменную `my_data2` и присвоим ей начальное значение:

```
var my_data2 = new Data("Fv, 12, 1978 16:45:10")
```

Переменная `my_data2` определяет дату 12 февраля 1978 года и время 16 часов 45 минут и 10 секунд. Значения часов, минут, секунд можно опустить, в этом случае они будут равны нулю:

```
var my_data3 = new Data("Feb, 12, 1978")
```

Параметры конструктора `new Data` могут определять год, месяц, число, время, минуты, секунды с помощью чисел. Дату 12 февраля 1978 года и время 16 часов 45 минут и 10 секунд можно задать так:

```
var my_data4 = new Data(78, 1, 12, 16, 45, 10)
```

Если время опустить, то описание будет следующим:

```
var my_data5 = new Data(78, 1, 12)
```

Все числовые представления даты нумеруются с нуля, кроме номера дня в месяце. Месяцы представляются числами от 0 (январь) до 11 (декабрь), поэтому второй параметр при задании переменных `my_data4` и `my_data5` равен 1.

Методами объекта `Date` можно получать и устанавливать отдельно значения месяца, дня недели, часов, минут и секунд.

- Метод `getDate` возвращает число в диапазоне от 1 до 31, представляющее число месяца.
- Метод `getHours` возвращает час суток. Значение возвращается в 24-часовом формате от 0 (полночь) до 23.
- Метод `getMinutes` возвращает минуты как целое от 0 до 59.
- Метод `getSeconds` возвращает число секунд как целое от 0 до 59.

В следующем примере эти методы используются для формирования текущего времени.

Определение текущего времени

Напишем сценарий, который определяет текущее время и выводит его в текстовое поле в формате "чч:мм:сс", как на рис. 11.1.

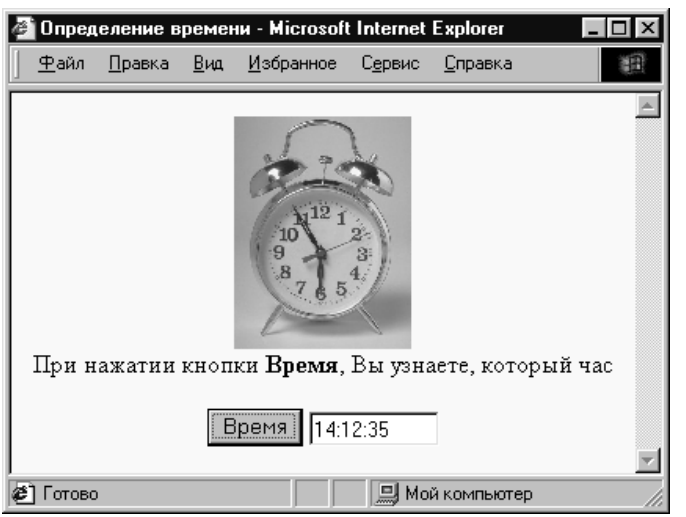


Рис. 11.1. Определение времени

В переменной `res` формируется строка, которая затем будет отображена в поле `rest` формы с именем `form1`. Для того чтобы уточнить время, следует еще раз нажать кнопку **Время** и т. д. Полностью сценарий приведен в листинге 11.1.

Листинг 11.1. Определение времени

```
<HTML>
  <HEAD>
    <TITLE>Определение времени</TITLE>
    <script language="JavaScript">
    <!-- //
      function cl()
       { var d=document
         var t=new Date()
         var h=t.getHours()
         var m=t.getMinutes()
         var s=t.getSeconds()
         var res=""
         if (h < 10)
           res += "0" + h
         else  res += h
         if (m < 10) res += ":0"+m
         else res += ":"+m
         if (s < 10) res += ":0"+s
         else res += ":"+s
         d.form1.rest.value = res
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFFFCC">
    <CENTER>
    <IMG src=alarmWHT.gif><br>
    При нажатии кнопки <B>Время</B>, Вы узнаете, который час
    <FORM name="form1">
      <input type="button" value=Время onClick="cl()">
      <input type="text" size=10 name="rest"><br>
    </FORM>
  </BODY>
</HTML>
```

Определение года, месяца, числа, дня недели и времени

Можно сделать так, что через некоторый заданный период значение времени будет обновляться. Для этого можно использовать функцию `setTimeout ("cl()", 3000)`. Напомним, что функция `setTimeout` выполняет указанные в первом параметре действия по истечении интервала времени, задаваемого вторым параметром. В приведенном примере через три секунды будет снова осуществлен вызов функции `cl`, результат отобразится в поле `rest` формы `form1`.

Объект `Data` обладает методами, которые позволяют определить день недели, номер месяца и год.

- Метод `getDay` возвращает день недели как целое число от 0 (воскресенье) до 6 (суббота).
- Метод `getMonth` возвращает номер месяца в году как целое число в интервале между 0 (январь) и 11 (декабрь). Обратите внимание, что номер месяца не соответствует стандартному способу нумерации месяцев.
- Метод `getYear` выдает год объекта.

Использование описанных методов потребуется для решения следующей задачи.

Напишем программу, которая определяет год, название месяца, число, день недели и время текущей даты (рис. 11.2).

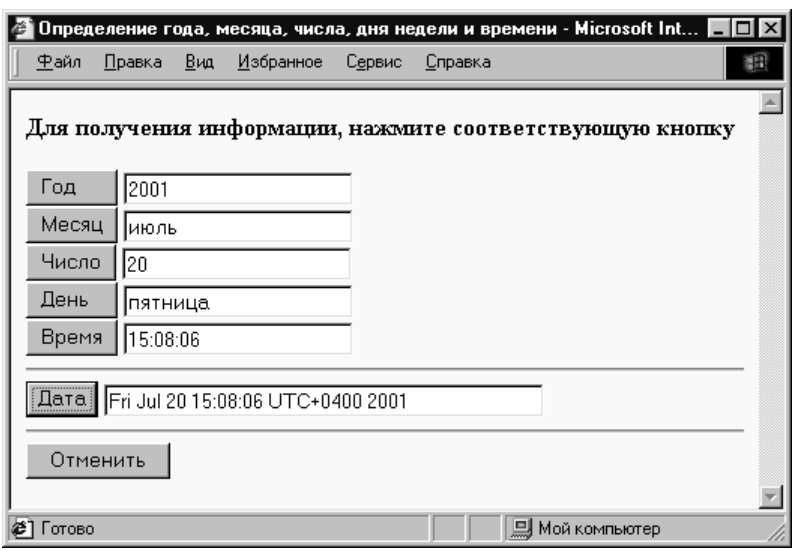


Рис. 11.2. Дата и время

Для того чтобы по номеру дня, определяемому методом `getDay`, вывести название дня, можно воспользоваться оператором выбора. Аналогично поступаем и в случае вывода названия месяца. При щелчке по соответствующей кнопке получаем требуемое значение. HTML-код с описанным сценарием представлен в листинге 11.2.

Листинг 11.2. Определение года, названия месяца, числа, дня недели и времени

```
<HTML>
  <HEAD>
    <TITLE>Определение года, месяца, числа, дня недели и времени</TITLE>
    <script language="JavaScript">
    <!-- //
      var d=document
      var t = new Date()
      function fyear (obj)
       { var year = t.getYear()
         obj.fye.value = year
       }
      function fmon (obj)
       { var s
         var mont = t.getMonth()
         switch (mont)
          { case 0: s="январь";   break;
            case 1: s="февраль";  break;
            case 2: s="март";     break;
            case 3: s="апрель";   break;
            case 4: s="май";      break;
            case 5: s="июнь";     break;
            case 6: s="июль";     break;
            case 7: s="август";   break;
            case 8: s="сентябрь"; break;
            case 9: s="октябрь";  break;
            case 10: s="ноябрь";  break;
            case 11: s="декабрь"; break;
          }
         obj.fm.value = s
       }
```

```
    function fdate(obj)
     { var d = t.getDate()
       obj.fdat.value = d
     }
    function fday(obj)
     { var day = t.getDay()
       var s
       switch (day)
        { case 0: s="воскресенье"; break;
          case 1: s="понедельник"; break;
          case 2: s="вторник";     break;
          case 3: s="среда";       break;
          case 4: s="четверг";     break;
          case 5: s="пятница";     break;
          case 6: s="суббота";     break;
        }
       obj.fd.value = s
     }
    function cl(obj)
     { var h = t.getHours()
       var m = t.getMinutes()
       var s = t.getSeconds()
       var res = "" + h
       if (m < 10)  res += ":0"+m
       else res += ":"+m
       if (s  < 10)  res += ":0"+s
       else res += ":"+s
       obj.clo.value = res
     }
    function aldate(obj)
     { obj.aldat.value = t }
  //-->
  </script>
</HEAD>
<BODY bgcolor="#FFFFCC">
  <H4>Для получения информации нажмите соответствующую кнопку</H4>
  <FORM name="form1">
    <input type="button" value="Год      " onClick="fyear(form1)">
```

```
      <input type="text" size=20 name="fye"><br>
      <input type="button" value=Месяц   onClick="fmon(form1)">
      <input type="text" size=20 name="fm"><br>
      <input type="button" value=Число   onClick="fdate(form1)">
      <input type="text" size=20 name="fdat"><br>
      <input type="button" value="День   "   onClick="fday(form1)">
      <input type="text" size=20 name="fd"><br>
      <input type="button" value=Время   onClick="cl(form1)">
      <input type="text" size=20 name="clo"><HR>
      <input type="button" value=Дата  onClick="aldate(form1)">
      <input type="text" size=40 name="aldat"><HR>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Определение рабочего и выходного дня

Перечисленные ниже методы позволяют устанавливать различные значения для объекта `Date`.

- Метод `setYear` устанавливает значение года для объекта `Date`. Метод `setDate` устанавливает день месяца. Параметр должен быть числом в диапазоне от 1 до 31.
- Метод `setMonth` устанавливает значение месяца. Параметр должен быть числом в диапазоне от 0 (январь) до 11 (декабрь).
- Метод `setHours` устанавливает час для текущего времени, использует целое число от 0 (полночь) до 23 для установки даты по 24-часовой шкале.
- Метод `setMinuts` устанавливает минуты для текущего времени, использует целое число от 0 до 59.
- Метод `setSeconds` устанавливает секунды для текущего времени, использует целое число от 0 до 59.
- Метод `setTime` устанавливает значение объекта `Date` и возвращает количество миллисекунд, прошедших с 1 января 1970 года.

Предположим, пользователь вводит год, название месяца и число. Требуется написать сценарий, в результате работы которого определяется, на какой день (рабочий или выходной) приходится заданная дата. В зависимости от дня недели в документе должен появляться соответствующий рисунок. Например, для рабочего дня документ будет иметь вид, как на рис. 11.3.

Рис. 11.3. Рабочие и выходные дни

В сценарии создается новый объект `Data` с именем `t`, для которого три значения определяются пользователем и берутся из соответствующих полей формы. Причем месяц определяется с помощью тега `<select>`. В теге `<option>` значение параметра `value` задает номер месяца в году. После того, как дата определена, с помощью метода `getDay()` выясняется номер дня недели, соответствующий этой дате. Функция `fday(day)` по номеру дня определяет его название, которое затем поступает в поле формы для формирования ответа на вопрос.

HTML-код со сценарием приведен в листинге 11.3.

Листинг 11.3. Рабочий или выходной день заданной даты

```
<HTML>
  <HEAD>
    <TITLE>Рабочий или выходной день заданной даты</TITLE>
    <script language="JavaScript">
    <!-- //
      var d=document
      var t = new Date()
      function fday(day)
```

```
      { var s
        switch (day)
         { case 0: s="воскресенье";  break;
           case 1: s="понедельник";  break;
           case 2: s="вторник";      break;
           case 3: s="среда";        break;
           case 4: s="четверг";      break;
           case 5: s="пятница";      break;
           case 6: s="суббота";      break;
         }
        return s
      }
     function def (obj)
      { var s
        t.setYear(Number(obj.fye.value))
        t.setDate(Number(obj.fdat.value))
        var m=Number(obj.fm.value)
        t.setMonth(m)
        var nday= t.getDay()
        var idt= fday (nday)
        if (nday==0 || nday==6)
         { s= "выходной день- "+ idt
           document.mypict.src="burger.gif"
         }
        else
         { s="рабочий день- "+ idt
           document.mypict.src="comp.gif"
         }
        obj.fd.value =s
      }
     function aldate(obj)
      { obj.aldat.value = t}
  //-->
  </script>
</HEAD>
<BODY>
  <CENTER>
  <H4>На какой день недели (рабочий или выходной)
      попадает заданная дата?</H4>
```

```
    <TABLE align= center border=3 cellpadding=4 cellspacing=4
           bgcolor=silver background="tile2.jpg">
      <TR><TH>Интересующая дата</TH><TH> попадает на</TH></TR>
      <TR><TD>
            <FORM name="form1">
            <pre>
год:    <input type="text" size=10 name="fye" >
месяц:  <select name="fm" size=1>
                <option value=0>январь
                <option value=1>февраль
                <option value=2>март
                <option value=3>апрель
                <option value=4>май
                <option value=5>июнь
                <option value=6>июль
                <option value=7>август
                <option value=8>сентябрь
                <option value=9>октябрь
                <option value=10>ноябрь
                <option value=11>декабрь
</select>
число:  <input type="text" size=10 name="fdat" >
           </pre><FORM></TD>
          <TD align=center><input type="text" size=27 name="fd"><br>
            <IMG src=comp.gif name=mypict height=100>
          </TD>
      </TR>
    </TABLE>
    <input type="button" value="Определить" onClick="def (form1)">
    <input type="reset" value="Отменить"><HR>
    <input type="button" value=Тест  onClick="aldate (form1)">
    <input type="text" size=30 name="aldat" ><HR>
  </BODY>
</HTML>
```

Если выбранная дата приходится на выходной день, то изменится рисунок, вместо изображения компьютера в документе появится праздничный пирог, как на рис. 11.4.

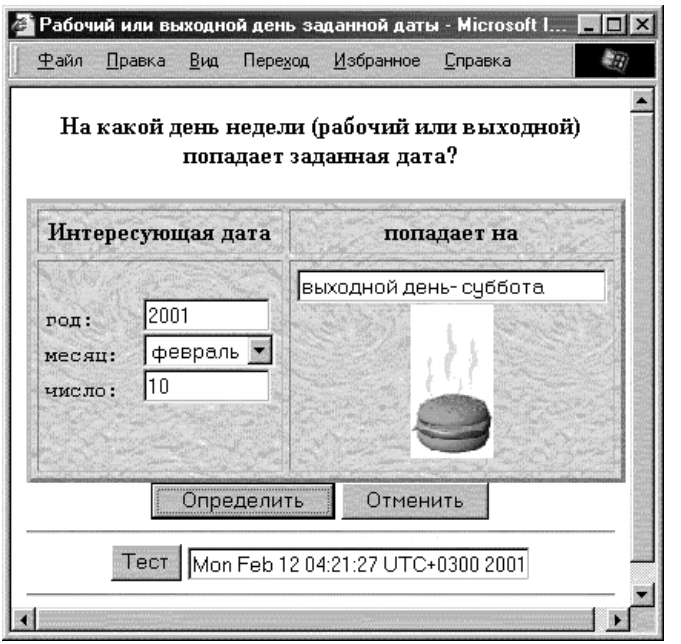


Рис. 11.4. Выходной день

Пятница 13

Напишем сценарий, с помощью которого определяются все даты в указанном году, приходящиеся на пятницу, 13 число.

При написании сценария будем поступать следующим образом: перебирать с помощью цикла месяцы и в каждом месяце устанавливать номер дня 13. Установка требуемой даты выполняется использованием методов работы с датой:

```
t.setYear (y)
t.setMonth (i)
t.setDate (13)
```

Далее следует проверить, какой номер дня соответствует этой дате. Если номер равен пяти (`(t.getDay())==5`), то день недели — пятница, найденный месяц следует запомнить. Для формирования ответа используется строковая переменная `c`, после запоминания названия месяца добавляется символ перевода строки. HTML-код со сценарием приведен в листинге 11.4.

Листинг 11.4. В какие месяцы года 13 число попадает на пятницу?

```
<HTML>
  <HEAD>
    <TITLE>В какие месяцы года 13 число попадает на пятницу?</TITLE>
    <script language="JavaScript">
    <!-- //
      function def13(obj)
       { var t= new Date()
         var c=""
         var y=Number(obj.fye.value)
         for (var i=0; i <=11; i++)
          { t.setYear(y)
            t.setMonth(i)
            t.setDate(13)
            if ((t.getDay())==5)
            c = c+ fmon(i)+ "\r\n"
          }
         obj.res.value = c
       }
      function fmon(mont)
       { var s
         switch (mont)
         { case 0: s="январь";   break;
           case 1: s="февраль";  break;
           case 2: s="март";     break;
           case 3: s="апрель";   break;
           case 4: s="май";      break;
           case 5: s="июнь";     break;
           case 6: s="июль";     break;
           case 7: s="август";   break;
           case 8: s="сентябрь"; break;
           case 9: s="октябрь";  break;
           case 10: s="ноябрь";  break;
           case 11: s="декабрь"; break;
         }
         return s
       }
```

```
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFFFCC">
    <H4>В какие месяцы заданного года число 13 попадает на пятницу?</h4>
    <FORM name="form1">
      Введите год:  <input type="text" size=8 name="fye" >
      <input type="button" value=Найти onClick="def13 (form1)"><br>
      <textarea Cols=30 rows=4 name=res></textarea><br>
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Результат работы сценария приведен на рис. 11.5.

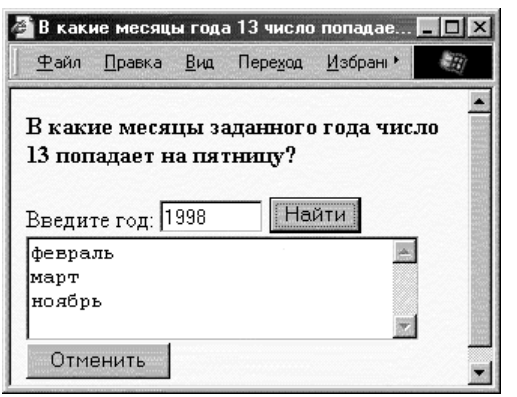


Рис. 11.5. Пятницы, попадающие на 13 число

Дата, время и день посещения Web-страницы

Напишем сценарий, который помещает в документ дату, время и день посещения страницы, как показано на рис. 11.6.

Основные действия выполняет функция `DateTime()`, в которой формируются необходимые компоненты помещаемой на странице информации. Обратите внимание, как теги HTML используются при формировании текстовой переменной. На рис. 11.6 некоторые слова выделены полужирным начертанием.

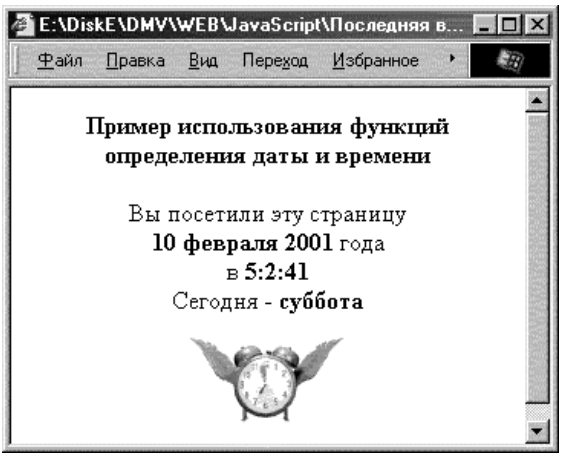


Рис. 11.6. Время посещения

HTML-код представлен в листинге 11.5.

Листинг 11.5. Дата посещения Web-страницы

```
<HTML>
  <HEAD>
    <script language="JavaScript">
      function NDays(r)
       { var s=""
         switch (r)
          { case 0: s="воскресенье";  break;
            case 1: s="понедельник";  break;
            case 2: s="вторник";      break;
            case 3: s="среда";        break;
            case 4: s="четверг";      break;
            case 5: s="пятница";      break;
            case 6: s="суббота";      break;
          }
         return s
        }
      function NMonths (r)
       { var s=""
         switch (r)
          { case 0: s="января";    break;
```

```
            case 1: s="февраля";   break;
            case 2: s="марта";     break;
            case 3: s="апреля";    break;
            case 4: s="мая";       break;
            case 5: s="июня";      break;
            case 6: s="июля";      break;
            case 7: s="августа";   break;
            case 8: s="сентября";  break;
            case 9: s="октября";   break;
            case 10: s="ноября";   break;
            case 11: s="декабря";  break;
          }
          return  s
        }
       function DateTime()
        { var now = new Date()
          var str= "Вы посетили эту страницу<br><b>"
          year = now.getYear() + ""
          str += now.getDate() + " " + NMonths(now.getMonth()) + " " +
                 now.getYear()  + "</b> года<br>"
          str += "в <b>" + now.getHours() + ":" + now.getMinutes() + ":" +
                 now.getSeconds() + "</b><br>"
          str += "Сегодня — <b>" + NDays(now.getDay()) + "</b>"
          document.write(str)
        }
    </script>
  </HEAD>
  <BODY>
    <H4 align=center>Пример использования функций определения
                     даты и времени</H4>
    <CENTER>
      <script language = "JavaScript">
        DateTime()
      </script>
      <br><img src=Time.gif align=center>
    </CENTER>
  </BODY>
</HTML>
```

Определение даты для заданного дня недели

Напишем сценарий, который определяет все даты заданного года, приходящиеся на определенный день недели, выбранный пользователем.

На рис. 11.7 приведен результат работы сценария, определяющий все даты 2001 года, приходящиеся на среду.

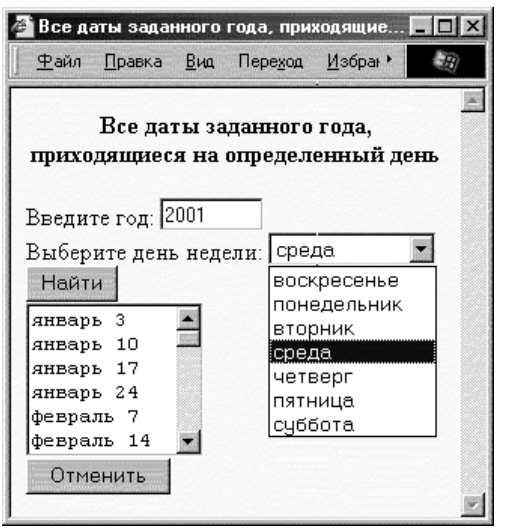


Рис. 11.7. Даты заданного дня недели

Сценарий, осуществляющий основную обработку, может быть описан так, как представлено в листинге 11.6.

Листинг 11.6. Даты, приходящиеся на выбранный день недели

```
function defdate(obj)
 {var t= new Date()
  var c=""
  var  y=Number(obj.fye.value)
  var a=obj.dayn.value
  t.setYear (y)
  for (var i=0; i <=11; i++)
   { t.setMonth (i)
     for (var k=1; k< 31; k++ )
```

```
      { t.setDate(k)
        if (t.getDay() == a)
          c = c+ fmon(i)+" " +k+ "\r\n"
       }
   }
   obj.res.value = c
}
```

Предыдущие примеры были рассмотрены подробно, поэтому полностью создать документ читателю рекомендуется в качестве упражнения.

Составление расписания занятий

Предположим, что занятия в некотором учебном заведении начинаются по мере комплектования групп. В анкете заполняется дата начала и конца работы группы. Для проведения занятий по определенной дисциплине выбирается день занятий. Требуется написать сценарий, который формирует расписание занятий.

После того как пользователь заполнил поля формы, указав дату начала и конца занятий группы и день недели для проведения занятий по некоторой дисциплине, определяется дата первого занятия. Предположим, группа начинает работу 15 февраля 2001 года (четверг), а занятия по заданной дисциплине должны проводиться по средам. Тогда дата первого занятия — 21 февраля.

Далее формируется дата следующего занятия `t` и сопоставляется с датой окончания работы курсов `w`. Сначала сравниваются года. Продолжать работу следует лишь в случае, когда год предполагаемого текущего занятия предшествует или равен году окончания занятий. Если года совпадают, то сравниваются месяцы, если и месяцы совпадают, то сравниваются даты. Другими словами, если текущая дата предшествует дате окончания, то она добавляется в расписание, формируется дата следующего предполагаемого занятия и т. д.

Для того чтобы получить расписание занятий, надо ввести дату начала и окончания занятий в анкете, представленной на рис. 17.8.

Сформированное расписание помещается в текстовое поле. Если даты выбраны неверно, т. е. дата окончания курсов предшествует дате начала работы курсов, то выдается сообщение о том, что требуется проверить введенные даты. Возможна ситуация, когда даты введены верно, а выбранный день не попадает в интервал времени, определенный для занятий. В этом случае также будет выдано предупреждающее сообщение. HTML-код документа со сценарием приведен в листинге 11.7.

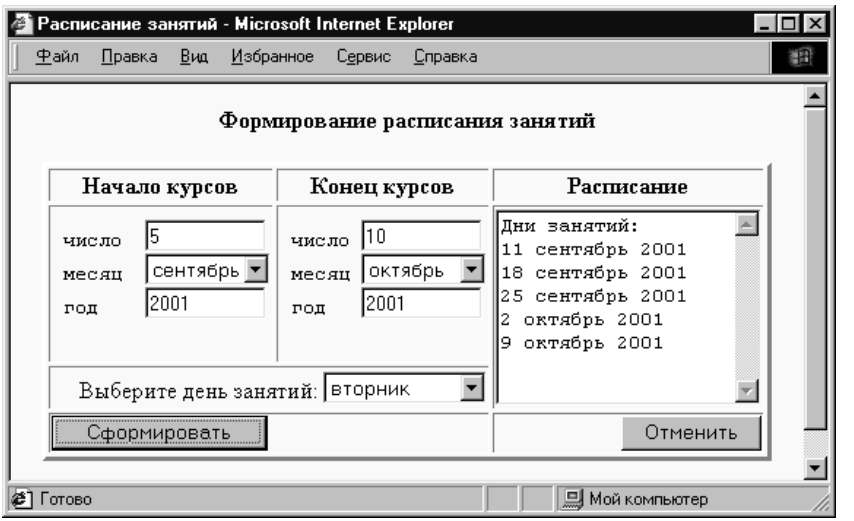


Рис. 11.8. Расписание занятий

Листинг 11.7. Расписание занятий

```
<HTML>
  <HEAD>
    <TITLE>Расписание занятий</TITLE>
    <script language="JavaScript">
    <!-- //
      function ident (mont)
       { var s
         switch (mont)
          { case 0: s="январь";  break;
            case 1: s="февраль";  break;
            case 2: s="март";     break;
            case 3: s="апрель";   break;
            case 4: s="май";      break;
            case 5: s="июнь";     break;
            case 6: s="июль";     break;
            case 7: s="август";   break;
            case 8: s="сентябрь"; break;
            case 9: s="октябрь";  break;
            case 10: s="ноябрь";  break;
            case 11: s="декабрь"; break;
```

```
    }
   return s
 }
function rasp(obj)
 { // дата начала работы группы
   var d= new Date(obj.begy.value, obj.begm.value, obj.begd.value)
   // дата окончания работы группы
   var w= new Date(obj.endy.value, obj.endm.value, obj.endd.value)
   // дата текущего занятия
   var t= new Date(obj.begy.value, obj.begm.value, obj.begd.value)
   var s=""
   // выбранный пользователем день работы
   var n=obj.dt.value
   // сформированная дата дня первого занятия
   var k=d.getDate()+Number(n)-Number(d.getDay())
   if (n < d.getDay())
     k += 7
   t.setDate(k)
   var s1 = "Дни занятий: "+"\n"
   // scur - дата предполагаемого текущего занятия
   var scur
   var s=s1
   // поиск дат занятий
   while (t.getYear() <= w.getYear())
    { // формирование даты очередного занятия
      scur=+ t.getDate()+" "+ident(Number(t.getMonth()))+
             " "+t.getYear()+"\n"
      if (t.getYear() < w.getYear())
        { s+= scur }
      else
       { if ( t.getMonth() < w.getMonth())
          {s +=scur}
         else
          { if (t.getMonth () == w.getMonth())
            { if (t.getDate()<w.getDate())
                {s +=scur}
              else
                { if (t.getDate()==w.getDate())
                   {s+= scur; break }
```

```
                        }
                      }
                    }
                  }
                k=t.getDate()+7
                t.setDate(k)
              }
            if (s==s1)
              alert ("Проверьте даты начала и конца занятий")
            else
              obj.res.value = s
        }
      //-->
      </script>
    </HEAD>
    <BODY bgcolor="#FFFFCC">
      <H4 align=center>Формирование расписания занятий</H4>
      <FORM name="form1">
        <TABLE border=3 align=center>
          <TR><TH>Начало курсов</TH><TH>Конец курсов</TH>
              <TH>Расписание</TH></TR>
          <TR><TD><pre>
 число  <input type="text" name="begd" size=10)>
 месяц  <select name="begm" size=1 >
                  <option value=0>январь
                  <option value=1>февраль
                  <option value=2>март
                  <option value=3>апрель
                  <option value=4>май
                  <option value=5>июнь
                  <option value=6>июль
                  <option value=7>август
                  <option value=8>сентябрь
                  <option value=9>октябрь
                  <option value=10>ноябрь
                  <option value=11>декабрь
                </select>
 год    <input type="text" name="begy" size=10 value=2001>
</pre></TD>
```

```
<TD><pre>
 число <input type="text" name="endd" size=10>
 месяц <select name="endm" size=1>
                <option value=0>январь
                <option value=1>февраль
                <option value=2>март
                <option value=3>апрель
                <option value=4>май
                <option value=5>июнь
                <option value=6>июль
                <option value=7>август
                <option value=8>сентябрь
                <option value=9>октябрь
                <option value=10>ноябрь
                <option value=11>декабрь
              </select>
 год   <input type="text" name="endy" size=8 value=2001>
</pre></TD>
      <TD rowspan=2>
        <textarea Cols=20 rows=8 name=res></textarea><br>
      </TD></TR>
      <TR><TD colspan=2 align=right>
          Выберите день занятий:
          <select name= dt size=1>
            <option value=0 >воскресенье
            <option value=1>понедельник
            <option value=2>вторник
            <option value=3>среда
            <option value=4>четверг
            <option value=5>пятница
            <option value=6>суббота
          </select>
      </TD></TR>
      <TR><TD colspan=2 align=left><br>
          <input type="button" value=Сформировать
                 onClick="rasp (form1)"><br></TD>
        <TD align=right><input type="reset" value=Отменить>
        </TD>
      </TR>
```

```
    </TABLE><br>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите сценарий, который по заданному времени определяет количество минут, прошедших от начала текущего занятия.
2. Напишите сценарий, который для заданной даты определяет, сколько дней прошло после некоторого события.
3. Напишите сценарий, с помощью которого можно выяснить количество дней между заданными датами.
4. Напишите сценарий, который рассчитывает, сколько рабочих и выходных дней между двумя заданными датами.
5. Напишите сценарий, который по заданной дате определяет номер недели в году.
6. Напишите сценарий, который по дате рождения человека определяет, под каким знаком зодиака родился человек.
7. В анкете приводятся данные о семи сотрудниках: фамилия и дата приема на работу. Напишите сценарий вычисления стажа работы (в годах) и определения максимального числа сотрудников с одинаковым стажем.
8. В анкете приводятся данные о шести сотрудниках: фамилия и дата приема на работу. Все сотрудники в зависимости от стажа работы разделяются на категории: проработавшие меньше года относятся к категории "молодой", от года до пяти лет — "опытный", более пяти лет — "ветеран". Напишите сценарий определения стажа работы каждого сотрудника и числа сотрудников в каждой категории. Постройте диаграмму, отражающую число сотрудников в категориях.
9. В анкете приводятся данные о семи сотрудниках: фамилия, дата рождения и оклад. Все сотрудники в зависимости от возраста разделяются на категории: менее 30 лет — "молодой", от 30 до 55 — "средний возраст", более 55 — "старший возраст". Напишите сценарий вычисления возраста каждого сотрудника и размер средней зарплаты для любой категории. Постройте диаграмму, отражающую среднюю зарплату в каждой из трех категорий.
10. В анкете заполняется информация о десяти сотрудниках: фамилия, дата заключения контракта и срок (в годах), на какой заключен контракт. Создайте форму для ввода данных. Напишите сценарий, определяющий:
    - дату окончания контракта;
    - месяц, в который закончится контракт;

- день недели, в который закончится контракт;
- сотрудников, контракт с которыми заканчивается в определенном году (год выбирается пользователем);
- сотрудников, контракт с которыми заканчивается в определенном месяце (месяц выбирается пользователем);
- сотрудников, контракт с которыми заканчивается в определенный день недели (день выбирается пользователем);
- сотрудников, контракт с которыми заканчивается в выходной день (субботу или воскресенье);
- сотрудников, контракт с которыми заканчивается в определенное время года (время года выбирается);
- сотрудников, контракт с которыми заканчивается в определенный квартал (номер квартала выбирается пользователем);
- в зависимости от введенной даты тех сотрудников, с которыми заключен контракт на заданный момент;
- сотрудников, контракт с которыми завершен на заданную дату.

11. В анкете заполняется информация о семи сотрудниках: фамилия, дата заключения контракта. Ровно через одиннадцать месяцев после начала работы сотрудникам предоставляется отпуск продолжительностью 24 рабочих дня. Создайте форму для ввода данных. Напишите сценарий, определяющий:
    - дату начала и окончания отпуска для каждого из сотрудников;
    - фамилии сотрудников, отпуск которым будет предоставлен в заданном месяце (месяц выбирается пользователем);
    - фамилии сотрудников, отпуск которых будет завершен в заданном квартале (номер квартала выбирается пользователем);
    - фамилии сотрудников, отпуск которым будет предоставлен в следующем календарном году;
    - фамилии сотрудников, отпуск которых будет завершен в следующем календарном году;
    - фамилии сотрудников, которые в данный момент работают (не находятся в отпуске), в зависимости от введенной пользователем даты.
12. В анкете заполняется информация о семи сотрудниках: фамилия, дата заключения контракта, продолжительность отпуска. Ровно через одиннадцать месяцев после начала работы сотрудникам предоставляется отпуск, продолжительность которого оговорена в контракте (12, 24, 30, 48, 60 дней). Напишите сценарий, определяющий:
    - дату начала и окончания отпуска для каждого из сотрудников;

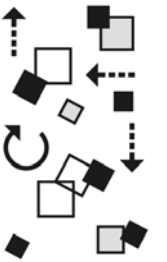
- фамилии сотрудников, начала отпусков которых приходятся на заданный месяц (месяц выбирается пользователем);
- в зависимости от введенной пользователем даты фамилии сотрудников, которым предоставлен отпуск;
- в зависимости от введенной пользователем даты фамилии сотрудников, которые в данный момент работают (не находятся в отпуске).

13. В анкете заполняется информация о семи сотрудниках, принятых на работу на основе почасовой оплаты: фамилия, оплата за час, дата начала работы, дата окончания. Напишите сценарий, при работе которого определяется:
    - количество рабочих дней для каждого сотрудника;
    - количество выходных дней между заданными датами;
    - размер гонорара, учитывая, что каждый рабочий день длится 8 часов (стоимость одного часа работы указана в договоре и содержится в анкете).
14. В анкету вносится информация о дате начала семестра, дате конца семестра. Требуется написать сценарий, который после выбора пользователем дня недели определяет количество часов, отведенных на данную дисциплину в определенном семестре. Напишите сценарий, определяющий:
    - даты проведения занятий;
    - по номеру занятия дату его проведения.
15. Предположим, что занятия в некоторой обучающей организации начинаются по мере комплектования групп. В анкете заполняется дата начала работы группы, количество занятий, отведенное на курс. Напишите сценарий, который определяет:
    - дату проведения занятия с заданным номером;
    - дату проведения контрольной работы (после половины занятий полагается контрольная работа);
    - дату проведения зачета (предпоследнее занятие отведено для сдачи зачета);
    - дату ближайшего занятия и его номер;
    - сколько занятий уже прошло и сколько предстоит провести.
16. В документе определяется дата начала и конца семестра. Преподаватель заполняет анкету, содержащую название дисциплины, и выбирает день недели для занятий. Вводятся сведения о трех дисциплинах. Напишите сценарий, определяющий:
    - расписание занятий (указывается дата, день недели и дисциплины);

- сколько занятий в семестре будет проводиться по каждой из дисциплин;
- неделю с максимальной загрузкой;
- неделю с минимальной загрузкой;
- расписание контрольных работ, в предположении, что контрольные проводятся примерно в середине курса, и не разрешено проводить две контрольные в один день;
- расписание зачетов, в предположении, что зачет проводится на последнем занятии.

17. Напишите сценарий, формирующий расписание занятий. Требуется предусмотреть ввод времени начала занятий. На каждое занятие отводится одна пара, состоящая из двух часов. Продолжительность часа занятий (30, 45, 50 минут) в паре определяется пользователем. Кроме того, требуется задать продолжительность перерыва между часами в паре и между парами, длительность обеденного перерыва, количество пар в день. В расписании требуется указать номер пары, время ее начала и окончания, время начала и окончания обеденного перерыва.
18. В старояпонском календаре был принят 60-летний цикл, состоящий из пяти 12-летних подциклов. Подциклы обозначались названиями цвета: зеленый, красный, желтый, белый, черный. Внутри каждого подцикла годы носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, курицы, собаки и свиньи. Например, 1984 год (год зеленой крысы) был началом очередного цикла. Напишите сценарий, который по заданной дате определяет название года по старояпонскому календарю.

Глава 12



Строки и методы работы с ними

Многие языки программирования позволяют обрабатывать строковые данные. Прежде, чем рассмотреть принципы работы со строками в языке JavaScript, уточним некоторые понятия, относящиеся к строкам. *Алфавит* — это конечное множество символов. *Строка* — конечная последовательность символов некоторого алфавита. Если рассматриваем алфавит A, состоящий из двух символов {0, 1}, то строками в алфавите A являются: 1001, 1, 01, 10, 1111 и т. д. *Пустая строка* — это строка, не содержащая ни одного символа. В строке важен порядок символов, так, строки 01 и 10 различны. Длина строки равна числу символов в строке. Длина пустой строки равна нулю, длина строки 11 равна двум, длина строки 1001 — четырем. Если X и Y — строки, то их *сцеплением* или *конкатенацией* называется строка XY, полученная приписыванием символов строки Y за символами строки X. В предыдущих примерах неоднократно использовались строковые литералы, например, при выводе некоторой информации в документ `Document.write ("Площадь равна", s)`. Напомним, что строковый литерал представляет собой последовательность символов, заключенную в одинарные или двойные кавычки. Строковые литералы или строковые переменные являются в языке JavaScript объектом типа `string`, к которому могут быть применены методы, определенные в языке. Создание нового объекта требует вызова функции-конструктора объекта. Для того чтобы создать строковый объект, надо применить конструктор `newString`, например:

```
s=newString("результат=")
```

Объект `string` имеет единственное свойство `length (длина_строки)`. Выражение `s.length` выдает значение 10, равное длине строки, содержащейся в строковом объекте `s`. Объект `string` имеет два типа методов. С методами, непосредственно влияющими на саму строку, мы сейчас и познакомимся, рассматривая примеры обработки текстовой информации.

Одним из часто используемых методов является метод выделения из строки отдельного символа. Метод `charAt(n1)` возвращает символ, позицию которого определяет параметр `n1`. Символы в строке перенумерованы, начиная с 0.

Вывод символов строки в "столбик"

Напишем сценарий, при выполнении которого заданный текст выводится в "столбик", т. е. на каждой строке размещается по одному символу.

При решении задачи из заданной строки последовательно выбираются символы. Формируется новая строка, в которой за каждым символом ставится последовательность символов, обеспечивающая переход на новую строку. Когда строка результата сформирована, то она размещается в текстовом поле формы, тем самым для исходной строки осуществляется вывод в "столбик". Сценарий, осуществляющий обработку строки, приведен в листинге 12.1.

Листинг 12.1. Вывод символов строки в "столбик"

```
<HTML>
  <HEAD>
    <TITLE>Вывод символов строки в "столбик"</TITLE>
    <script language="JavaScript">
    <!-- //
      function ttest(s)
       { var sres="Прочитанный текст:"+" \r\n"+s+"\r\n"+
                  'Текст в "столбик":'+"\r\n"
         var cur=""
         for ( var i=0; i <= s.length-1; i += 1)
           {c=s.charAt(i);  cur +=c+"\r\n" }
         sres+=cur
         return sres
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFFFCC">
    <H4>Символы текущей строки в столбик</H4>
    <FORM name="form1">
      Введите строку:   <input type="text" size=20 name="st1"><hr>
      <input type="button" value=Выполнить
                  onClick="form1.res.value=ttest(form1.st1.value)">
      <input type="reset" value=Очистить><hr>
      <textarea cols=20 rows=7 name= res></textarea>
```

```
    </FORM>
  </BODY>
</HTML>
```

На рис. 12.1 изображен документ, соответствующий приведенному коду.

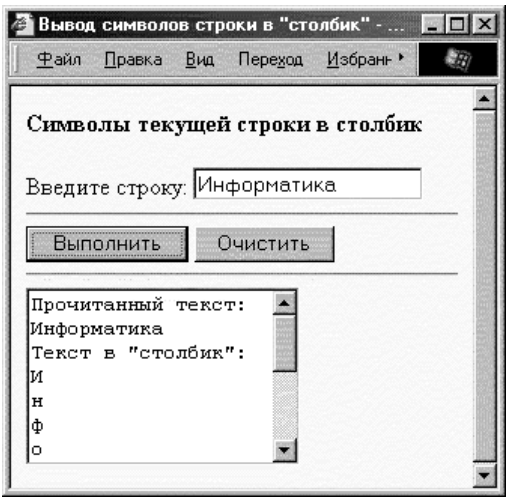


Рис. 12.1. Вывод текста в столбик

Строка является универсальным способом представления информации. Часто пользователь вводит информацию в виде строки, а программа должна уметь распознавать, какого рода данные были введены. В следующей задаче вводимая строка представляет собой последовательность оценок, и требуется эту строку проанализировать и извлечь из нее нужные данные.

Сводка по результатам проведения экзамена

Напишем программу, которая по последовательности оценок, полученных на экзамене, формирует сводку. В сводке содержатся сведения о числе студентов, сдавших экзамен на "отлично", "хорошо", "удовлетворительно" и "неудовлетворительно". Кроме того, подсчитывается средний балл, на который сдан экзамен. Для вычисления среднего балла суммируются все оценки, полученные на экзамене, и рассчитанная сумма делится на число студентов.

В функцию `svod` поступает строка. С помощью цикла просматриваются символы строки, выбираются те, которые соответствуют оценке, остальные игнорируются. В функции используются четыре переменные `t2`, `t3`, `t4`, `t5`

для хранения числа оценок. В теле цикла все оценки суммируются для того, чтобы затем получить средний бал по предмету. После анализа всей строки в соответствующих полях формы выводится требуемая информация так, как изображено на рис. 12.2.

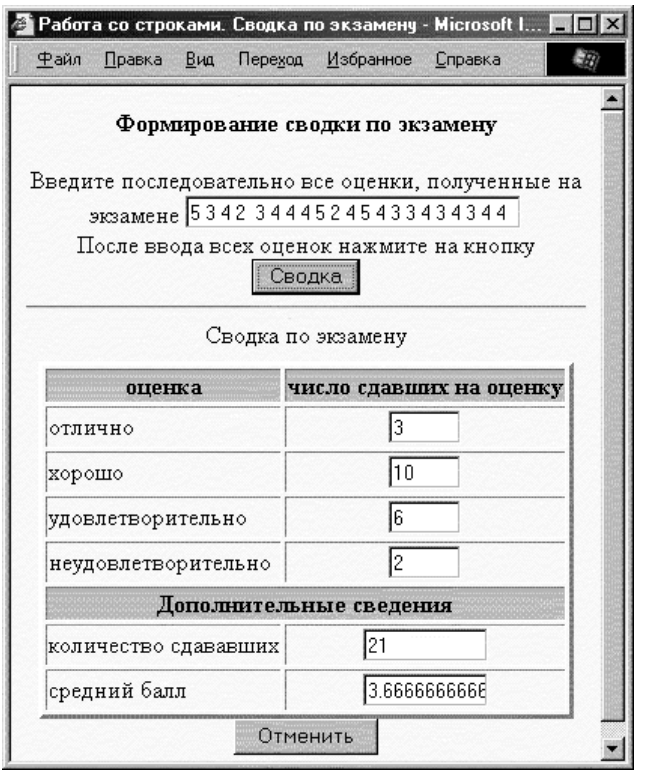


Рис. 12.2. Сводка по экзамену

HTML-код представленного на рисунке документа со сценарием, формирующим сводку по экзамену, представлен в листинге 12.2.

Листинг 12.2. Формирование сводки по экзамену

```
<HTML>
  <HEAD>
    <TITLE>Работа со строками. Сводка по экзамену</TITLE>
    <script language="JavaScript">
    <!-- //
      function svod(obj)
```

```
      { var t2=0;   var t3 =0;   var t4 =0;   var t5=0
        var s =0;   var n =0;    var i;       var r
        var st=obj.data.value
        for ( i=0; i< st.length; i++)
         { r = st.charAt(i);
           if (r>="2" && r<="5")
             { n=n+1
               s=s+Number(r)
               switch (r)
                { case "2":  {t2++;  break};
                  case "3":  {t3++;  break};
                  case "4":  {t4++;  break};
                  case "5":  {t5++;  break};
                }
             }
         }
        obj.res2.value=t2;    obj.res3.value=t3
        obj.res4.value=t4;    obj.res5.value=t5
        obj.num.value=n;      obj.sr.value=s/n
      }
  //-->
  </script>
</HEAD>
<BODY bgcolor="#FFFFCC">
  <H4 align=center>Формирование сводки по экзамену</H4>
  <FORM name="form1"><CENTER>
    Введите последовательно все оценки, полученные на экзамене
    <input type="text" name="data" size="30"> <br>
    После ввода всех оценок нажмите на кнопку
    <input type="button" value="Сводка" onClick="svod(form1)"><hr>
    <TABLE border=3><caption>Сводка по экзамену</caption>
      <TR><TH bgcolor=silver>оценка</TH>
          <TH bgcolor=silver>число сдавших на оценку</TH></TR>
      <TR><TD>отлично</TD>
        <TD align=center><input type="text" name="res5"
                                size="5"></TD></TR>
      <TR><TD>хорошо</TD>
        <TD align=center><input type="text" name="res4"
                               size="5"></TD></TR>
```

```
        <TR><TD>удовлетворительно</TD>
          <TD align=center><input type="text"
                                  name="res3" size="5"></TD></TR>
        <TR><TD>неудовлетворительно</TD>
          <TD align=center><input type="text" name="res2"
                                  size="5"></TD></TR>
        <TR><TH colspan=2 bgcolor=silver align=center>
          Дополнительные сведения</TH></TR>
        <TR><TD>количество сдававших</TD>
          <TD align=center><input type="text" name="num"
                                  size="10"></TD></TR>
        <TR><TD>средний балл</TD>
          <TD align=center><input type="text" name="sr" size="10">
          </TD></TR>
      </TABLE>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
<HTML>
```

Проверка идентификатора

Напишем программу, определяющую, является ли последовательность введенных символов идентификатором. Под *идентификатором* будем понимать последовательность букв латинского алфавита (как прописных, так и строчных) и цифр. Последовательность символов, являющихся идентификатором, должна начинаться с буквы.

При решении задачи проверим первый символ введенной строки. Если символ не является буквой, то задача решена: введенная последовательность символов не является идентификатором. Если же первый символ — буква, то анализ строки следует продолжить. Поочередно просматриваются все символы последовательности. Если какой-либо элемент не является буквой или цифрой, то анализ строки следует прекратить, а работу цикла прервать. Если просмотрели все символы строки, и каждый из них удовлетворяет нашему условию, то введенная строка представляет собой идентификатор.

HTML-код приведен в листинге 12.3.

Листинг 12.3. Проверка, является ли последовательность символов идентификатором

```
<HTML>
  <HEAD>
    <TITLE>Последовательность символов - идентификатор</TITLE>
    <script language="JavaScript">
    <!-- //
      function idt(obj)
       { var s=obj.data.value
         var res=1
         var ch=s.charAt(0)
         if (!(ch>='a' && ch<='z' || ch>='A' && ch<='Z')) res=0;
         if (res!=0)
          { for (var i=1; i <= s.length-1; i++)
            { ch=s.charAt(i)
              if (!(ch>='a' && ch<='z' || ch>='A' && ch<='Z' ||
                    ch>='0' &&  ch<='9'))
                {res=0; break}
            }
          }
         if (res==1) obj.result.value="идентификатор"
         else obj.result.value="не является идентификатором"
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Является ли последовательность символов идентификатором?</H4>
    <FORM name="form1">
      <input type="text" name="data" size="8">
      <input type="button" value="Определить" onClick="idt(form1)">
      <input type="text" name="result" size="28"><hr>
      <input type="reset" value="Отменить">
    </FORM></BODY></HTML>
```

Вычисление количества повторений символа в строке

Необходимо написать программу, которая определяет, сколько раз некоторый символ встречается в заданном тексте.

Исследуемый текст определяется в форме с помощью тега <textarea>. Алгоритм решения прост. Просматривается строка слева направо, и каждый символ строки сравнивается с заданным символом. После просмотра всего текста будет определено число символов в тексте, совпадающих с данным. Полностью программа представлена в листинге 12.4.

Листинг 12.4. Количество заданных символов в тексте

```
<HTML>
  <HEAD>
    <TITLE>Количество заданных символов в тексте</TITLE>
    <script language="JavaScript">
    <!-- //
      function numsym(obj)
       { var h=obj.data.value
         var s=obj.textin.value
         var res=0
         for (var i=0; i <= s.length-1; i++)
          { ch=s.charAt(i)
            if (ch==h)
              {res+=1}
          }
         obj.result.value=res
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    Количество заданных символов в тексте
    <FORM name="form1">
      Введите текст:<br>
      <textarea name="textin" rows=4 cols=20></textarea><hr>
      Введите символ: <input type="text" name="data" size="8"><hr>
      <input type="button" value="Определить"
```

```
            onClick="numsym(form1)"><hr>
      Количество символов в тексте:
      <input type="text" name="result" size=8><hr>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Одним из часто используемых является метод выделения из строки требуемой подстроки. Метод `substring(n1,n2)` возвращает подстроку, заданную индексами `min(n1,n2)` и `max(n1,n2)-1`. Если `n1=n2`, то результатом будет пустая строка. Если строка `s="Решение"`, то значением `s.substring(0,1)` будет первый символ строки `"Р"`. Результатом выполнения `s.substring(0,7)` будет вся строка.

Вывод префиксов строки

Напишем программу, которая выводит все префиксы заданной строки.

Строка `X` называется *префиксом* строки `S`, если строка `S` представима как `XY`. На рис. 12.3 показано, как может быть организован вывод префиксов введенной строки.

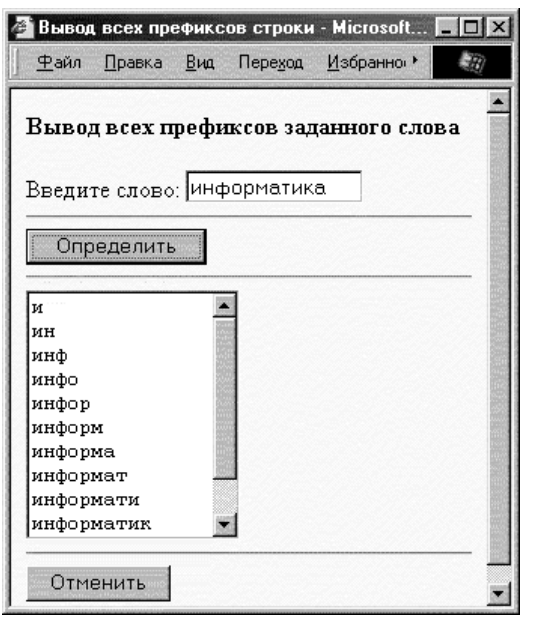


Рис. 12.3. Определение префиксов строки

При решении задачи будем использовать метод `substring`, "наращивая" на каждом шаге итерации строку, являющуюся префиксом заданной (листинг 12.5).

Листинг 12.5. Вывод всех префиксов строки

```
<HTML>
  <HEAD>
    <TITLE>Вывод всех префиксов строки</TITLE>
    <script language="JavaScript">
    <!-- //
      function pref(obj)
       { var s=obj.data.value
         var res=""
         for (var i=1; i <= s.length; i++)
          { res +=s.substr(0,i)+"\r\n" }
            obj.textpref.value = res
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFFFCC">
    <H4>Вывод всех префиксов заданного слова</H4>
    <FORM name="form1">
      Введите слово: <input type="text" name="data" size="15"><hr>
      <input type="button" value="Определить" onClick="pref(form1)"><hr>
      <textarea name="textpref" rows=10 cols=15></textarea><hr>
      <input type="reset" value="Отменить">
    </FORM>
  </BODY>
</HTML>
```

Вывод суффиксов строки

Необходимо написать программу, которая выводит все суффиксы заданной строки. Строка `Y` называется *суффиксом* строки `S`, если строка `S` представима как `XY`.

При решении этой задачи следует сначала выделить последний символ строки, затем предпоследний и последний и т. д. Функция `suf` осуществляет

формирование строки, которая затем будет помещена в соответствующее поле формы, в строке хранятся суффиксы введенного слова:

```
function suf(obj)
 { var s=obj.data.value
   var n=s.length
   var res =""
   for (var i=n-1; i >=0 ; i--)
     { res +=s.substr(i,n)+"\r\n" }
   obj.textsuf.value = res
 }
```

Метод `substr(n1,n2)` также позволяет выделять из строки подстроку. Параметр `n1` задает позицию первого символа подстроки; параметр `n2` определяет количество символов в подстроке. Например, если строка `s="сборник"`, то в результате выполнения `substr(0,4)` будет выделена подстрока `"сбор"`.

Вычисление количества повторений строки в тексте

Напишем программу, которая определяет, сколько раз заданное слово встречается в определенном тексте.

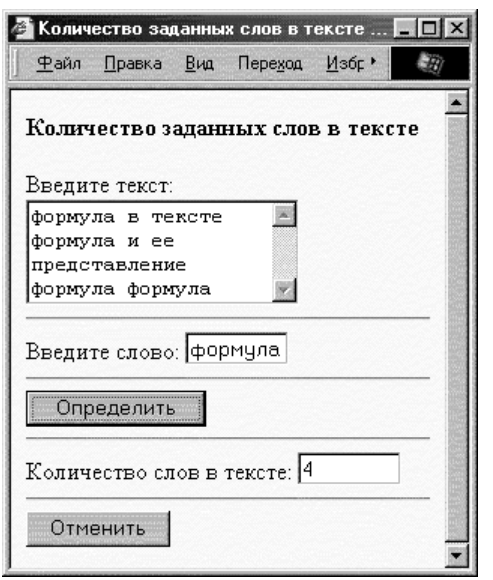


Рис. 12.4. Число заданных слов в тексте

В тексте слова разделяются пробелами. После того как очередное слово найдено, просмотр продолжается с символа, следующего за найденным словом. На рис. 12.4 показано, как может выглядеть документ.

HTML-код документа представлен в листинге 12.6.

Листинг 12.6. Количество заданных слов в тексте

```
<HTML>
  <HEAD>
    <TITLE>Количество заданных слов в тексте</TITLE>
    <script language="JavaScript">
    <!-- //
      function numword(obj)
       { var h=obj.data.value
         var s=obj.textin.value
         s='  '+s+' '
         h=' '+h+' '
         var m=h.length
         var res=0
         var i=0
         while (i <= s.length-1)
          { ch=s.substr(i,m)
            if (ch==h) {res+=1; i = i+m-1}
            else
              i++
          }
         obj.result.value=res
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="#FFFFCC">
    <H4>Количество заданных слов в тексте</H4>
    <FORM name="form1">
      Введите текст:<br>
      <textarea name="textin" rows=4 cols=20></textarea><hr>
      Введите слово: <input type="text" name="data" size="8"><hr>
     <input type="button" value="Определить"
            onClick="numword(form1)"><hr>
```

```
    Количество слов в тексте: <input type="text" name="result"
                                    size=8><hr>
    <input type="reset" value="Отменить">
   </FORM>
  </BODY>
</HTML>
```

Зеркальная перестановка символов

Напишем функцию, которая формирует строку, являющуюся перевернутой по отношению к заданной. Если задана строка "топор", то по ней должна быть построена строка "ропот".

Просматривается исходная строка `s` слева направо, и одновременно формируется новая строка `s1`, элементы в которую добавляются с начала. Функция `rev` переворачивает исходную строку.

```
function rev(s)
{ var n=s.length-1
  var h
  var s1=''
  for (var i=0; i <=n; i++)
   { s1+=s.charAt(n-i) }
  return s1
 }
```

Палиндром

Напишем функцию, определяющую, является ли заданное предложение палиндромом.

Палиндромом считается строка, которая читается одинаково как слева на право, так и справа налево. Палиндромами являются слова: казак, кок, шалаш и др. Палиндромами могут быть фразы, обычно считается, что пробелы между словами, знаки препинания и различия между маленькими и большими буквами игнорируются. Например, палиндромами являются фразы:

- а роза упала на лапу Азора
- кит на море не романтик

На рис. 12.5 изображен результат выполнения сценария анализа текста.

Будем анализировать текст с обоих концов, пропускать пробелы. Если оба анализируемых символа не являются пробелами, то сравним эти символы. Если они различны, то анализ текста можно завершить, фраза не является

палиндромом. Если же очередные символы одинаковы, то анализ следует продолжить. Заметим, что значение `p1` в программе может только увеличиваться, значение `p2` лишь уменьшаться. Процесс проверки следует прекратить в случае, когда значение `p1` станет больше или равно значению `p2`. Это означает, что просмотрена вся строка, и символы на соответствующих местах одинаковы.

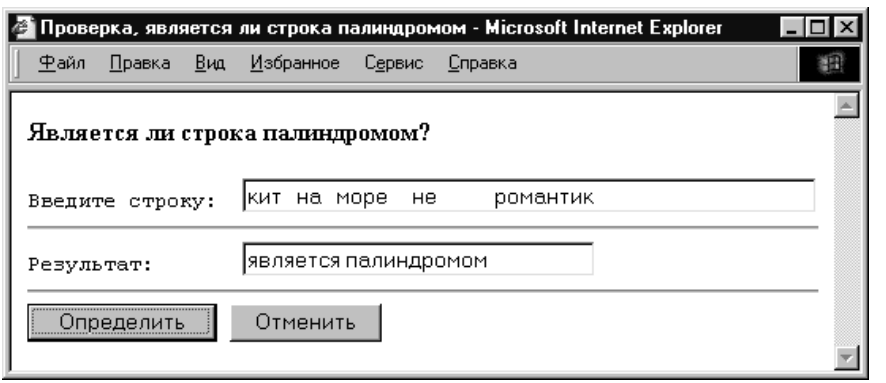


Рис. 12.5. Строки-палиндромы

HTML-код приведен в листинге 12.7.

Листинг 12.7. Строка-палиндром

```
<HTML>
  <HEAD>
    <TITLE>Проверка, является ли строка палиндромом</TITLE>
    <script language="JavaScript">
    <!-- //
    function pal(obj)
     { var s=obj.data.value
       var n=s.length-1
       var p1=0
       var p2=n
       var c1=s.charAt(0)
       var c2=s.charAt(n)
       var p=true
       var s1=""
       while ((p1<p2)&&(p==true))
        { if (c1==" ")  {p1+=1; c1= s.charAt(p1) }
```

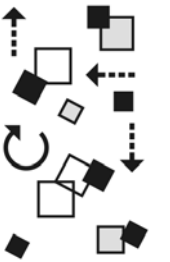
```
          else
            if (c2==" ")  {p2 -=1; c2= s.charAt(p2)}
            else
              if (c1==c2)
               {p1+=1; c1=s.charAt(p1); p2-=1; c2=s.charAt(p2)}
              else {p = false}
        }
       if (!p) s1="не является палиндромом"
       else s1="является палиндромом"
       obj.res.value=s1
      }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Является ли строка палиндромом?</H4>
    <FORM name="form1">
    <pre>
Введите строку:  <input type="text" name="data" size="50"><hr>
Результат:       <input type="text" name="res" size="30"><hr>
<input type="button" value="Определить" onClick="pal(form1)"><hr>
<input type="reset" value="Отменить">
</pre>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите программу, которая в последовательности чисел находит число правильных троек. Тройка чисел A, B, C называется правильной, если верно B – A = C – B.
2. Напишите программу, которая определяет, является ли последовательность символов идентификатором в заданном языке программирования.
3. В строке хранятся фамилия, имя и отчество, разделенные пробелами. Напишите программу, которая выводит:
   - фамилию;
   - инициалы;
   - имя.

4. Напишите программу, которая проверяет, является ли одна из строк префиксом или суффиксом другой заданной строки.
5. Слова в заданном тексте разделяются пробелами. Напишите программу, которая определяет:
   - количество слов в тексте;
   - *n*-е слово в тексте (если значение *n* больше, чем слов в тексте, то программа должна выдать соответствующее сообщение);
   - последнее слово в тексте;
   - самое короткое слово в тексте.
6. Напишите программу, в которой все слова A заменены словом B, где A и B — заданные слова, возможно, различной длины.
7. Напишите программу, которая "сжимает" заданный текст, т. е. заменяет все подряд идущие пробелы на один.
8. Напишите программу, которая все подряд идущие одинаковые символы заменяет на один.
9. Напишите программу, которая в заданном тексте находит все те слова, которые состоят только из заданных букв.
10. Напишите программу, которая определяет в заданном тексте все слова-палиндромы.
11. Напишите программу, которая определяет, состоят ли два заданных текста из одних и тех же слов.
12. Напишите программу, которая анализирует три текста:
    - определяет все слова, которые встречаются в каждом из текстов;
    - все слова, которые встречаются лишь в одном из текстов;
    - все слова, которые встречаются, по крайней мере, в двух текстах.
13. Напишите программу, которая определяет, что все слова в тексте различны.

Глава 13



Стандартные функции работы со строками

В языке JavaScript определены стандартные функции, при работе с которыми не требуется создавать никакого объекта. Функции осуществляют анализ своих аргументов.

Функция `Number(s)` преобразует строковый параметр `s` в число. Эту функцию уже неоднократно использовали в предыдущих сценариях. Функция `String(n)` преобразует число `n` в строку.

Автоморфные числа

Напишем сценарий, в результате работы которого определяется, является ли введенное число автоморфным числом. Напомним, что натуральное число называется *автоморфным*, если оно содержится в качестве младших цифр в своем квадрате. Например, число 25 является автоморфным, т. к. $25^2=625$.

Опишем функцию `avtomorf`, которая преобразует введенную строку в число и находит квадрат числа. Далее работа осуществляется со строками. Выделяются последние символы строки, представляющей квадрат числа, и сравниваются со строкой, соответствующей числу. Результатом этого анализа является решение задачи. Приведем сценарий в листинге 13.1.

Листинг 13.1. Автоморфные числа

```
<HTML>
  <HEAD>
    <TITLE>Автоморфные числа</TITLE>
    <script language="JavaScript">
    <!-- //
      function avtomorf(obj)
       { var a=Number(obj.num.value)
         var p=a*a
         obj.num1.value=p
         var sa= String(a)
```

```
        var sp= String(p)
        var nsa= sa.length
        var nsp= sp.length
        endsp=sp.substr(nsp-nsa,nsa)
        if (endsp==sa) obj.res.value="автоморфное"
        else obj.res.value="не является автоморфным"
      }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Проверка, является ли заданное число автоморфным</H4>
    <FORM name="form1">
<pre>
Число:            <input type="text" size=25 name="num"><hr>
Число в квадрате: <input type="text" size=25 name="num1"><hr>
Результат:        <input type="text" size=25 name="res"><hr>
</pre>
      <input type="button" value=Определить onClick="avtomorf(form1)">
      <input type="reset"  value=Очистить>
    </FORM>
  </BODY>
</HTML>
```

Автоморфные числа в заданном интервале

Напишем сценарий, определяющий все автоморфные числа в заданном интервале.

Функция `interav` обеспечивает перебор всех натуральных чисел в заданном диапазоне. Для каждого из чисел диапазона проверяется, является ли оно автоморфным, и если это так, то формируется строка результата. В строку результата заносится само число и его квадрат. Кроме того, информация о каждом числе располагается на отдельной строке. Строка результата помещается в текстовое поле. Если в качестве промежутка взять интервал [20; 9999], то результат работы сценария будет таким, как на рис. 13.1.

Текст программ, реализующих поиск автоморфных чисел в заданном интервале, и HTML-код представлены в листинге 13.2.

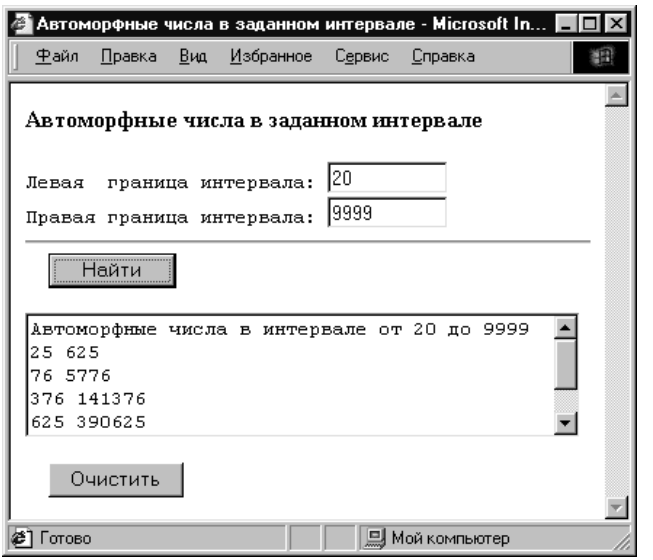


Рис. 13.1. Автоморфные числа в заданном интервале

Листинг 13.2. Автоморфные числа в заданном интервале

```
<HTML>
  <HEAD>
    <TITLE>Автоморфные числа в заданном интервале</TITLE>
    <script language="JavaScript">
    <!-- //
      function interav(obj)
       { var s=""
         var l=obj.left.value
         var r=obj.rig.value
         var s="Автоморфные числа в интервале от "+l+" до " +r+"\r\n"
         for (var i=l; i<=r; i++)
           if (avtomorf(i))
             s +=i+" "+i*i+"\r\n"
         obj.result.value=s
       }
      function avtomorf(a)
       { var p=a*a
         var sa= String(a)
         var sp= String(p)
```

```
          var nsa= sa.length
          var nsp= sp.length
          endsp=sp.substr(nsp-nsa,nsa)
          return (endsp==sa)
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Автоморфные числа в заданном интервале</H4>
    <FORM name="form1">
<pre>
Левая  граница интервала: <input type="text" size=10 name="left">
Правая граница интервала: <input type="text" size=10 name="rig"><hr>
  <input type="button" value="   Найти   "  onClick=" in-
terav(form1)"><br>
<textarea cols=45 rows=5 name=result></textarea><br>
  <input type="reset"  value="Очистить">
</pre>
    </FORM>
  </BODY>
</HTML>
```

Числа Армстронга в заданном интервале

Напишем сценарий, определяющий все числа Армстронга, расположенные в заданном интервале. Натуральное число называется *числом Армстронга*, если оно совпадает с суммой k-ых степеней составляющих его цифр, где k — количество цифр в числе. Например, таким является число 153, т. к. $153=1^3+5^3+3^3$.

Функция `arm` получает в качестве параметра строку, для которой требуется определить, представляет ли содержимое строки число Армстронга. Переменная `ls` определяет длину строки или количество цифр в числе. Далее в строке отщепляется по одному символу, символ преобразуется в число, определяется значение в заданной степени, и полученный результат добавляется к формируемой сумме. После того, как все символы строки будут просмотрены, проверяется условие для числа Армстронга. Функция `arm` выдает значение `true`, если число является числом Армстронга:

```
function arm(s)
 { var h; var k; var r
   var ls= s.length
```

```
   sum=0
   for (var i=0; i<=ls-1; i++)
    { h=s.charAt(i)
      k=Number(h)
      r=Math.pow(k,ls)
      sum+=r
    }
   return (s==String(sum))
}
```

Перебор осуществляется функцией `interav`. Просматриваются все числа из заданного диапазона, и для каждого из них проверяется, является ли данное значение числом Армстронга. Так как параметр функции `arm` — строка, то и при обращении к ней в функции `interav` числовой параметр `i` преобразуется в строку. Результат работы помещается в текстовое поле. Каждое число располагается в отдельной строке. Описанные сценарии приведены в листинге 13.3.

Листинг 13.3. Числа Армстронга в заданном интервале

```
<HTML>
  <HEAD>
    <TITLE>Числа Армстронга в заданном интервале</TITLE>
    <script language="JavaScript">
    <!-- //
      function interav(obj)
       { var s=""
         var l=obj.left.value
         var r=obj.rig.value
         var s="Числа Армстронга в интервале от "+l+" до " +r+"\r\n"
         for (var i=l; i<=r; i++)
          if (arm(String(i)))
             s+=i+"\r\n"
         obj.result.value=s
       }
      function arm(s)
       { var h; var k; var r
         var ls= s.length
         sum=0
         for (var i=0; i<=ls-1; i++)
          { h=s.charAt(i);  k= Number (h);  r=Math.pow(k,ls);  sum +=r }
         return (s==String(sum))
```

```
        }
    //-->
    </script>
  </HEAD>
  <BODY>
    <h4>Числа Армстронга в заданном интервале</h4>
    <FORM name="form1">
<pre>
Левая граница  интервала: <input type="text" size=10 name="left">
Правая граница интервала: <input type="text" size=10 name="rig"><hr>
<input type="button" value="  Найти  " onClick=" interav(form1)"><br>
<textarea cols=45 rows=8 name=result></textarea><br>
<input type="reset"  value="Очистить">
</pre>
    </FORM>
  </BODY>
</HTML>
```

Если поиск чисел производить в интервале [100; 9999], то результат работы сценария будет таким, как на рис. 13.2.

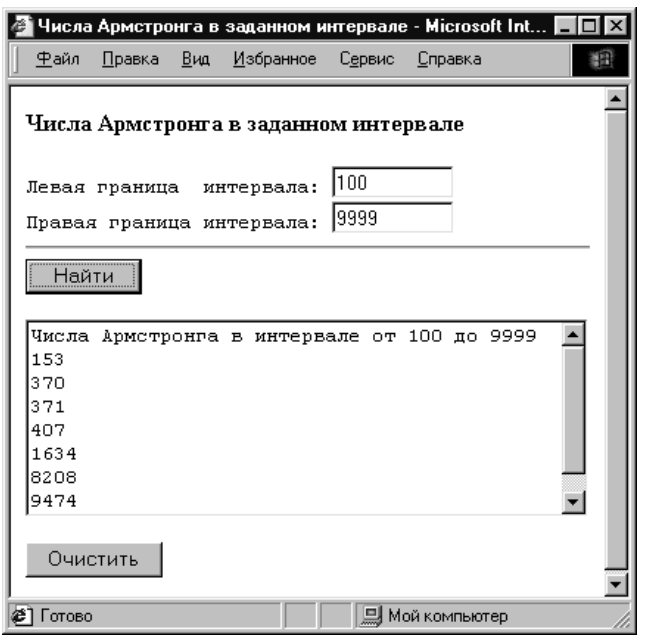


Рис. 13.2. Числа Армстронга в заданном интервале

Системы счисления

Функция `parseInt(s,n)` возвращает целое число по основанию, заданному в качестве второго параметра. Если первый символ в строке не является цифрой в указанной вторым параметром системе счисления, то функция `parseInt(s,n)` возвращает значение `"NaN"`.

Приведем сценарий, который содержит функцию-тест, демонстрирующую работу функции `parseInt(s,n)`. В результате работы теста исследуется введенная строка, и выводятся числа, соответствующие этой строке в различных системах счисления (листинг 13.4).

Листинг 13.4. Демонстрационные тесты для функции `parseInt(s,n)`

```
<HTML>
  <HEAD>
    <TITLE>Тест, демонстрирующий работу функции parseInt(s,n)</TITLE>
    <script language="JavaScript">
    <!-- //
      function test(obj)
       { var s=obj.numtest.value
         var t1=parseInt(s,16); obj.test1.value=t1
         var t2=parseInt(s,10); obj.test2.value=t2
         var t3=parseInt(s,8);  obj.test3.value=t3
         var t4=parseInt(s,2);  obj.test4.value=t4
         var t5=parseInt(s,7);  obj.test5.value=t5
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Тест, демонстрирующий работу функции parseInt(s,n)</P>
<pre>
<FORM name="form1">
Исходная строка:                  <input type="text" size=25
name="numtest"><hr>
Число в шестнадцатеричной системе <input type="text" size=25
name="test1"><hr>
Число в десятичной системе        <input type="text" size=25
name="test2"><hr>
Число в восьмеричной системе      <input type="text" size=25
name="test3"><hr>
```

```
Число в двоичной системе          <input type="text" size=25
name="test4"><hr>
Число в семеричной системе        <input type="text" size=25
name="test5"><hr>
<input type="button" value=Тест onClick="test(form1)"><hr>
<input type="reset"  value=Очистить>
</pre>
    </FORM>
  </BODY>
</HTML>
```

Идентификация кратности 9

Напишите программу, которая по введенной строке определяет, является ли содержимое строки числом, кратным 9.

Итак, строка может не представлять числа, или представлять число, не кратное 9, или, наконец, представлять число, кратное 9. Сначала убедимся в том, что строка содержит число, а затем проанализируем, является ли остаток от деления данного числа на 9 равным нулю. Для выполнения этих проверок воспользуемся стандартными функциями `parseFloat` и `isNaN`.

- Функция `parseFloat(s)` анализирует значение строкового параметра `s` и определяет, соответствует ли строка `s` представлению вещественного числа. Если в строке содержится символ, отличный от символов, разрешенных при формировании вещественного литерала, то оставшаяся часть строки игнорируется, в качестве результата функции возвращается числовое значение, которое обнаружено до неправильного символа. Если первый символ в строке не является цифрой, то функция `parseFloat(s)` возвращает значение `"NaN"`.
- Функция `isNaN(s)` проверяет, является ли параметр `s` числом. Если параметр `s` не является числом, то функция возвращает значение `true`, в противном случае — значение `false`.

Сценарий приведен в листинге 13.5.

Листинг 13.5. Представляет ли строка число, кратное 9

```
<HTML>
  <HEAD>
    <TITLE>Проверка, является ли содержимое строки числом,
           кратным 9</TITLE>
    <script language="JavaScript">
    <!-- //
```

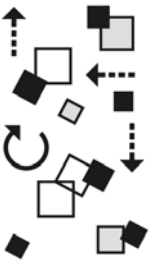
```
      function test(obj)
       { var n
         var s= obj.num.value
         var stres=""
         if (isNaN(s))
           stres="Введенная строка не является числом"
         else
          { n=Number(s)
            if ((n%9)==0)
              stres="Введенная строка - число, кратное 9"
            else
              stres="Введенная строка - число, не кратное 9"
          }
         obj.res.value=stres
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Проверка, является ли заданная строка числом, кратным 9</P>
<pre>
<FORM name="form1">
Исходная строка:  <input type="text" size=45 name="num"><hr>
Результат:        <input type="text" size=45 name="res"><hr>
<input type="button" value=Определить  onClick="test(form1)"><hr>
 <input type="reset" value=Очистить>
</pre>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Напишите программу, которая определяет, является ли введенная последовательность символов "изображением" целого двоичного числа без знака.
2. Напишите программу, которая для последовательности символов определяет, содержит ли она целое значение, кратное 5.

3. Напишите сценарий, в результате работы которого выясняется, является введенная последовательность символов натуральным числом, которое при делении на 7 дает в остатке 5.
4. Напишите сценарий, определяющий *k*-ю цифру последовательности, в которой выписаны подряд:
   - все натуральные числа (12345678910111213 14…);
   - квадраты натуральных чисел (149162536…);
   - все числа Фибоначчи (1123581321…);
   - все простые числа (2357111317…).

Глава 14



Массивы и методы работы с ними

Общие сведения

Массив представляет собой набор элементов, доступ к которым осуществляется по индексу. Массив создается оператором `new` и конструктором массива — системной функцией `Array`. Создать массив из названий дней недели можно, например, следующим образом:

```
var NDays = new Array("воскресенье","понедельник", "вторник",
"среда", "четверг", "пятница","суббота")
```

В качестве параметров конструктору передаются значения элементов массива. Можно создать массив, указав в нем лишь число элементов, например, так:

```
var Ndays1= new Array(7)
```

И, наконец, можно использовать конструктор без параметра:

```
var Ndays2= new Array()
```

В этом случае определяется лишь, что переменная `Ndays2` используется в качестве массива.

Все элементы массива перенумерованы, начиная с нуля. Для получения значения элемента массива необходимо задать имя массива и в квадратных скобках порядковый номер элемента. Для того чтобы получить доступ к первому элементу массива, следует воспользоваться конструкцией `Ndays[0]`.

Свойство `length` позволяет определить число элементов в массиве или, как говорят, длину массива. Доступ к последнему элементу массива можно осуществить, например, следующим образом:

```
Ndays[Ndays.length-1]
```

При описании переменной `Ndays1` задана только длина массива. Значения элементов массива можно указать с помощью оператора присваивания:

```
Ndays[0]="воскресенье"
Ndays[1]="понедельник"
Ndays[2]="вторник"
Ndays[2]="среда"
```

```
Ndays[2]="четверг"
Ndays[2]="пятница"
Ndays[6]="суббота"
```

Функция определения выходного/рабочего дня

Напишем функцию, которая по строке, являющейся названием дня недели, определяет, является ли этот день выходным или рабочим (листинг 14.1).

Листинг 14.1. Функция определения по номеру дня, выходной он или рабочий

```
function grd(s)
 { var res="рабочий"
   if ((s== NDays[0]) || (s== NDays[6]))
   res="выходной"
   return res
 }
```

Определение времени посещения Web-страницы

Необходимо написать сценарий, который формирует сведения о текущей дате и времени в следующем виде:

Вы посетили эту страницу
1 ноября 2000 года
в 5:26:03
Сегодня — среда

При решении задачи используется два массива: один для хранения названий месяцев, другой для хранения дней недели. В переменной `str` формируется строка, которая затем будет выведена в документе (листинг 14.2).

Листинг 14.2. Время посещения страницы с использованием массивов

```
<HTML>
  <HEAD>
    <script language="JavaScript">
      // Названия месяцев
      NMonths = new Array("января", "февраля", "марта", "апреля", "мая",
                          "июня", "июля", "августа", "сентября",
                          "октября", "ноября", "декабря")
```

```
      // Названия дней недели
      NDays = new Array("воскресенье", "понедельник", "вторник", "среда",
                        "четверг", "пятница", "суббота")
      function DateTime()
       { var now = new Date()
         var minute = now.getMinutes ()
         var second = now.getSeconds ()
         var  hour = now.getHours()
         var min = ( (minute < 10 ) ? ":0" : ":") + minute
         var sec = ( (second < 10 ) ? ":0" : ":") + second
         var  str = "Вы посетили эту страницу<br>"
         year = now.getYear() + " "
         month = now.getMonth()
         imonth = NMonths[ month]
         day=now.getDay()
         iday= NDays[day]
         str += now.getDate() + " " + imonth  + " " + year + " года<br>"
         str += "в " + hour + min + sec + "<br>"
         str += "Сегодня — " + iday
         document.write(str)
       }
    </script>
  </HEAD>
  <BODY>
    <H4 align=center>Пример использования функций
                     определения времени</H4>
    <CENTER>
      <script language = "JavaScript">
        DateTime()
      </script>
      </CENTER>
  </BODY>
</HTML>
```

Далее рассмотрим несколько классических задач, посвященных работе с массивами. Приведем функции работы с массивами, которые ценны сами по себе и могут применяться в различных программах.

Поиск максимального элемента массива

Напишем функцию, которая определяет максимальный элемент массива.

Сначала в качестве максимального элемента берется первый элемент массива. Затем просматриваются элементы массива, и каждый элемент сравнивается с максимальным значением уже рассмотренных элементов. Если текущий элемент больше максимального значения, то максимальным становится текущий элемент. Таким образом, после просмотра всего массива определяется максимальный элемент.

HTML-код приведен в листинге 14.3.

Листинг 14.3. Максимальный элемент массива

```
function maxelem (v)
 { var m= v[0]
   for (var i=1; i <= v.length-1; i++)
    { if (v[i] > m)
         m=v[i]
    }
   return m
 }
```

Определение количества максимальных элементов в массиве

Напишем функцию, определяющую количество максимальных элементов в массиве. Если задан массив, состоящий из элементов (4, 5, 5, 5, 3, 5, 2, 3, 1, 3, 4), то максимальным значением массива является 5, которое встречается 4 раза.

При просмотре массива кроме максимального значения запоминается количество максимальных значений для просмотренной части массива. Если очередной элемент массива больше, чем максимальное значение для просмотренных элементов, то максимальным становится очередной элемент, а количество максимальных устанавливается в 1. Если очередной элемент совпадает с максимальным, то число найденных максимальных элементов увеличивается на 1 (листинг 14.4).

Листинг 14.4. Количество максимальных элементов в массиве

```
function nummax (v)
{ var m= v[0]
  var k=1
  for (var i=1; i <= v.length-1; i++)
   { if (v[i] > m)
       {m=v[i]; k=1}
     else
       if (v[i] == m) k++
    }
  return k
}
```

Поиск заданного элемента в неупорядоченном массиве

Напишем функцию, определяющую, присутствует ли заданный элемент в массиве.

Если у нас нет дополнительной информации об организации элементов массива, то при поиске элемента следует воспользоваться алгоритмом, который получил название *линейный поиск*. Согласно этому алгоритму будем сравнивать поочередно все элементы массива `v` с образцом `t`. Если очередной элемент совпадает с `t`, то задача решена. Если искомого элемента в массиве нет, то убедимся в этом, лишь просмотрев все элементы массива. В качестве результата работы функции можно выдавать логическое значение `true`, если элемент найден, и `false` — в противном случае. Иногда удобно выводить номер найденного элемента, если же элемента в массиве нет, то отобразить некоторое заданное значение, например, `-1` (листинг 14.5).

Листинг 14.5. Поиск элемента в неупорядоченном массиве

```
function linsear(v,t)
 { var k=-1
   for (var i=0; i <= v.length-1; i++)
     if (v[i] == t )
       {k=i; break}
   return k
 }
```

Поиск заданного элемента в упорядоченном массиве

Напишем функцию, определяющую, присутствует ли заданный элемент в упорядоченном массиве.

Если элементы массива упорядочены, то организовать поиск элемента с заданным значением можно согласно алгоритму *бинарного поиска*. Пусть переменная `i` — индекс первого элемента, значение `j` — индекс последнего элемента массива, среди которых осуществляется поиск. Определяется индекс элемента `k`, находящегося посередине. Далее `k`-й элемент массива `v[k]` сравнивается с образцом `t`. Если окажется, что `t<=v[k]`, то поиск следует продолжать среди элементов с индексами `[i, k]`, если же `t>v[k]`, искать надо среди элементов `[k+1, j]`. Процесс поиска продолжается до тех пор, пока исследуемая часть массива не станет равной одному элементу, тогда результат будет зависеть от того, равен этот элемент образцу или нет. Функцию `binsear(v,t)` опишем так, как указано в листинге 14.6.

Листинг 14.6. Поиск элемента в упорядоченном массиве

```
function binsear (v,t)
   var i=0
   var j= v.length-1
   var k
   while (i < j)
    { k=Math.round((i+j)/2+0.5)-1
        if (t <= v[k])
          j=k
        else
          i=k+1
    }
    if (v[i] == t)  { return i}
    else return -1
}
```

Заметим, что значение переменной `i` может лишь увеличиваться, значение переменной `j` только уменьшаться, причем при каждом повторении тела цикла либо увеличивается значение переменной `i`, либо уменьшается значение переменной `j` и значение переменной `i` меньше значения `j`. Когда значения `i` и `j` совпадут, цикл завершит работу. Рассмотрим случай, когда исследуются только два элемента. Значение `k` на следующем шаге итерации станет равным значению `i`. Если значение `t` меньше или равно `v[i]`, то пе-

ременной j будет присвоено значение k, значения переменных i и j совпадут, цикл завершит работу. Если же значение t больше v[i], то переменной i присвоится значение k+1, и в этом случае значения i и j совпадут.

После выполнения цикла сравнивается значение i-го элемента со значением t, и результат выдается в качестве работы функции. Пусть исходный массив состоит из элементов, упорядоченных по возрастанию (2, 3, 5, 6, 6, 7, 10, 11, 20, 25), а значение t равно 7. Ожидаемый результат — индекс найденного элемента, равный 5.

Бинарный поиск с формированием таблицы результатов

Напишем функцию, которая реализует алгоритм бинарного поиска таким образом, чтобы во время работы программы формировалась таблица значений переменных i, j, k и некоторых выражений так, как показано на рис. 14.1.

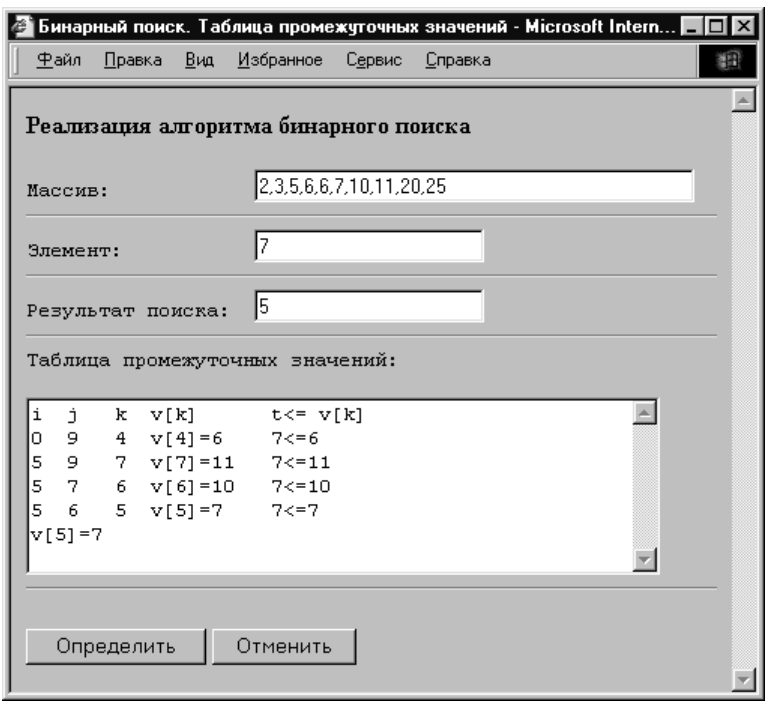


Рис. 14.1. Бинарный поиск с промежуточными значениями

Текст функции представлен в листинге 14.7.

Листинг 14.7. Поиск в упорядоченном массиве с таблицей промежуточных значений

```
<HTML>
  <HEAD>
    <TITLE>Бинарный поиск. Таблица промежуточных значений</TITLE>
    <script language="JavaScript">
      <!-- //
        var v=new Array(2,3,5,6,6,7,10,11,20,25)
        function testtab(obj,v,t)
         { var res="i  j   k  v[k]      t<= v[k]"+"\r\n"
           var i=0
           var j= v.length-1
           var k
           while ( i < j )
            { k=Math.round( (i+j)/2+0.5)-1
              res = res + i + "  "+j+"   "+k+"  "+"v[" + k + "]="  + v[k]
+ "    " + t + "<=" + v[k]+"\r\n"
              if (t <= v[k] )
                j=k
              else
                i=k+1
            }
           res += "v[" + i + "]=" +v[i]+"\r\n"
           obj.result1.value=res
           if (v[i] == t )
             { return i}
           else return -1
         }
        function test(obj)
         { obj.data1.value=v}
      //-->
    </script>
  </HEAD>
  <BODY bgcolor=silver>
    <H4>Реализация алгоритма бинарного поиска</H4>
    <FORM name="form1">
<pre>
Массив:            <INPUT type="text" size=40 name="data1"  ><hr>
```

```
Элемент:           <INPUT type="text" size=20 name="data2"  ><hr>
Результат поиска:  <INPUT type="text" size=20 name="result" ><hr>
Таблица промежуточных значений:  <BR>
<textarea cols=50 rows=7 name="result1" > </textarea><hr>
</PRE>
 <input type="button" value=Определить
     onClick="test(form1); form1.result.value=testtab
(form1,v,form1.data2.value)">
  <input type="reset" value=Отменить>
</FORM>
  </BODY>
</HTML>
```

Идентификация симметричности массива

Напишем функцию, определяющую, является ли массив *симметричным*, т. е. совпадает ли его первый элемент с последним, второй с предпоследним и т. д.

Решать задачу будем следующим образом. Сначала сравним первый и последний элементы массива. Если они не совпадают, то задача решена, и массив не является симметричным. Если значения первого и последнего элементов массива совпадают, то анализ массива требуется продолжить. Функция `simarr1(v)` реализует описанный алгоритм (листинг 14.8).

Листинг 14.8. Проверка, является ли массив симметричным

```
function simarr1 (v)
 { var p= true
   var n=  v.length
   for (var i=0; i <= (n-1)/2;  i++)
    if (v[i] != v[n-1-i])  {p = false; break}
   return p
 }
```

Приведем еще один вариант решения задачи. Будем использовать методы работы с объектом `Array`, определенные в JavaScript. Метод `reverse()` переставляет элементы массива в обратном порядке, первым элементом становится последний, вторым — предпоследний, а последний — первым. Метод `join()` преобразует элементы массива в строку. При решении задачи преобразуем исходный массив в строку `sv`, далее "перевернем" исходный массив и

опять преобразуем в строку. Если массив симметричен, то строки должны совпадать. Функция `simarr` в этом случае может быть описана так:

```
function simarr(v)
 { var sv=v.join()
   v.reverse()
   var sw=v.join()
   return (sv == sw)
 }
```

Объединение массивов с упорядочиванием результата

Пусть заданы два массива, состоящие из целых чисел, упорядоченных по возрастанию. Требуется написать функцию построения третьего массива, также упорядоченного по возрастанию и содержащего элементы как первого, так и второго массивов.

Опишем алгоритм решения задачи, который называется алгоритмом слияния двух упорядоченных массивов. Исходные массивы просматриваются "параллельно" до тех пор, пока в одном из них не будут исчерпаны все элементы. После этого непросмотренные элементы второго массива переписываются в массив результата без изменений. Пусть `i` — индекс очередного элемента массива `a`, `j` — индекс очередного элемента второго массива `b`. В массиве `c` содержатся уже упорядоченные элементы первого массива в количестве `(i-1)` и элементы второго массива в количестве `(j-1)`. Если `a[i]<=b[j]`, то в массив `c` записывается элемент `a[i]` и осуществляется переход к анализу следующего элемента массива `a`, т. е. значение переменной `i` увеличивается на `1`. Если же `a[i]>b[j]`, то в массив `c` помещается `j`-й элемент массива `b` и происходит переход к анализу очередного элемента массива `b`.

Если один из массивов просмотрен полностью, то в результирующий массив помещаются элементы другого массива. Опишем функцию `sl1(a,b)` (листинг 14.9, а).

Листинг 14.9, а. Объединение двух упорядоченных массивов. Вариант 1

```
function sl1(a,b)
 { var m=a.length-1
   var n=b.length-1
   var p=m+n
   var c=new Array (p)
```

```
  var i=0; var j=0; var k=0;
  while ((i <= m) && (j <= n))
    {if (a[i] <= b[j])
      {c[k] = a[i]; i++}
     else
      {c[k] = b[j]; j++}
     k++
    }
    if (i > m )
      { while (j<=n)
         {c[k]=b[j]; k++; j++}
      }
    else
      { while (i <= m)
         {c[k]=a[i]; k++; i++}
      }
    return c
}
```

Функцию `sl1` можно упростить, если в исходные массивы вслед за основными содержащимися в них элементами поместить элемент, заведомо бóльший любого элемента исходного массива. Тогда не возникает ситуации "дописывания" непросмотренных элементов какого-либо из массивов. В этом варианте решения в функции `sl2` будет использоваться лишь одни цикл (листинг 14.9, б).

Листинг 14.9, б. Объединение двух упорядоченных массивов. Вариант 2

```
function sl2(a,b)
 { var m=a.length
   var n=b.length
   a[m]=marker
   b[n]=marker
   var p=m+n
   var c=new Array (p)
   var i=0; var j=0; var k=0;
   for (k=0; k<p; k++)
     {if (a[i] <= b[j])
       {c[k] = a[i]; i++}
      else
```

```
      {c[k] = b[j]; j++}
    }
  return c
}
```

Результат работы функции представлен на рис. 14.2.

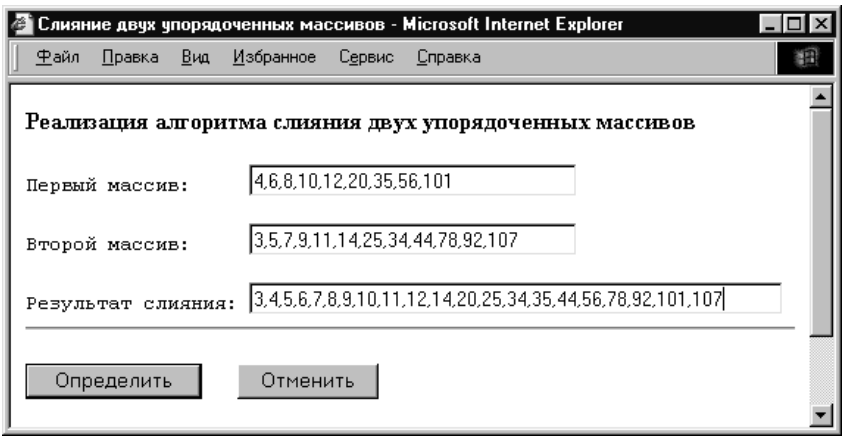


Рис. 14.2. Слияние упорядоченных массивов

При выполнении описанных функций `sl1` и `sl2` исходные массивы просматриваются только один раз.

Другой вариант решения задачи: соединить два массива в один, а затем отсортировать по возрастанию получившийся массив. Такое решение никак не использует упорядоченность исходных массивов и признано неэффективным из-за алгоритма сортировки. Приведем такое решение, потому что заманчиво использовать тот факт, что объект `Array` имеет метод `sort()`, позволяющий упорядочивать элементы массива в лексикографическом порядке, и метод `concat()`, объединяющий два массива в один. Используя оба метода, опишем функцию `sl3(a,b)`:

```
function sl3(a,b)
 { var c=new Array()
   c=a.concat(b).sort()
   return c
 }
```

При использовании метода `concat()` два массива объединяются в один:

```
a.concat(b)
```

а затем, чтобы упорядочить полученный массив, применяется метод `sort`:

```
c=a.concat(b).sort()
```

Для исходных данных предыдущего примера в результате работы функции `sl3` будет получен результат: 10, 101, 107, 11, 12, 14, 20, 25, 3, 34, 35, 4, 44, 5, 56, 6, 7, 78, 8, 9, 92.

Мы видим, что результат неверен, т. к. элементы массива сравниваются как строки. Если элементами исходных массивов действительно являются строки, то функция `sl3` работает правильно. Один из тестов приведен на рис. 14.3. Заметим, что если бы исходные массивы были не отсортированы, то результат был бы тот же.

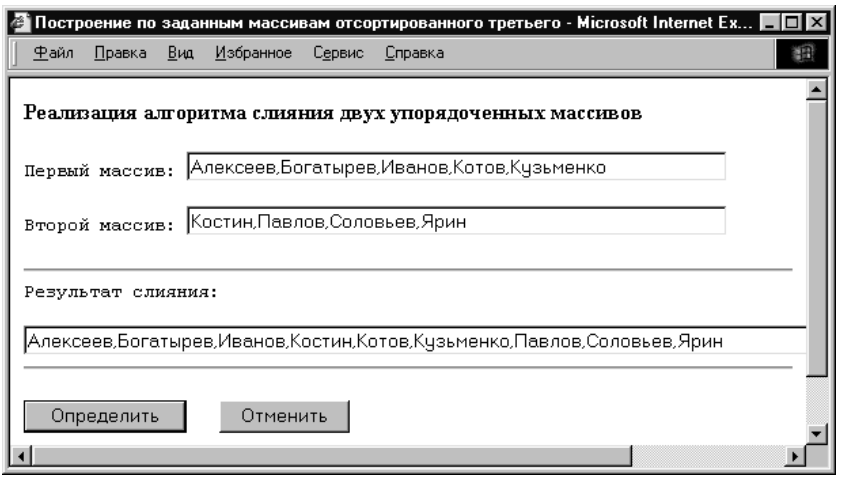


Рис. 14.3. Слияние массивов из строк

Перестановка элементов массива

Пусть в исходном массиве содержатся ненулевые вещественные числа разных знаков. Требуется написать функцию, которая переставляет элементы массива таким образом, что сначала располагаются все отрицательные элементы, а затем все положительные.

При первом варианте решения введем дополнительный массив и в него станем записывать элементы в нужном порядке. При этом будем использовать две переменные, в которых сохраним индексы последних записанных в новый массив значений: отрицательных и положительных. Исходный массив просматривается один раз, и необходимый массив будет сформирован (листинг 14.10).

Листинг 14.10, а. Переформирование массива. Вариант 1

```
function divar(a)
 { var el
```

```
  var i=-1
  var n=a.length
  var j=n
  var b=new Array()
  for (var k=0; k <= n-1; k++)
    { el=a[k]
      if (el < 0)
       { i++; b[i]=el }
      else {j--; b[j]=el}
    }
  a=b
  return a
}
```

Можно было бы всю работу по формированию массива произвести в исходном массиве. Для этого так же, как и в предыдущем случае, можно использовать две переменные `i` и `j`, "движущиеся" навстречу друг другу таким образом, что за переменной `i` остаются только отрицательные элементы, за переменной `j` — положительные. На каждом шаге работы цикла проверяются знаки `i`-го и `j`-го элементов. В зависимости от знаков элементов на `i`-м и `j`-м местах сдвигается либо один, либо другой индекс. Если же сдвинуть ни одну из переменных нельзя, то меняются местами крайние элементы рассматриваемой части массива и процесс продолжается (листинг 14.10, б).

Листинг 14.10, б. Переформирование массива. Вариант 2

```
function divar(a)
 { var n = a.length
   var i=0
   var j=n-1
   var el1=a[i]
   var el2=a[j]
   while (i < j)
    { if (el1 < 0)
       { i++; el1 = a[i] }
      else
        if (el2 >0)
          { j--; el2=a[j] }
        else
          { a[i]=el2; a[j]=el1; i++; el1=a[i]; j--; el2=a[j] }
```

```
      }
   return a
 }
```

Для решения этой задачи мы могли бы описать простую функцию, которая использует метод сортировки массива `Array`. В результате выполнения оператора `a.sort()` (листинг 14.10, в) элементы будут отсортированы в лексикографическом порядке, и хотя для чисел сортировка выполнена неверно, но задача разделения массива, заключающаяся в том, что сначала располагаются отрицательные, затем положительные элементы, выполнена.

Листинг 14.10, в. Переформирование массива. Вариант 3

```
function divar(a)
 { a.sort()
   return a
 }
```

Исполнение функции на одном из тестовых примеров приведено на рис. 14.4.

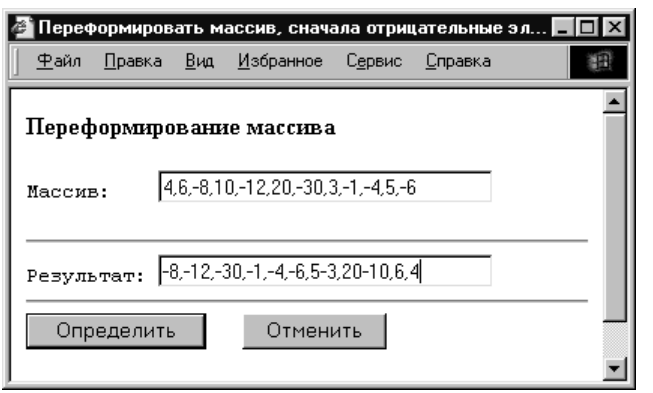


Рис. 14.4. Переформирование массива

Добавление элемента в массив без нарушения упорядоченности

Напишем функцию, добавляющую в заданный отсортированный массив элемент, не нарушая упорядоченности.

Будем просматривать элементы массива с конца, сдвигая каждый из них вниз до тех пор, пока не встретится элемент, значение которого меньше

значения вставляемого элемента. Когда необходимый элемент найден, на свободное место помещается добавляемый элемент (листинг 14.11).

Листинг 14.11. Добавление элемента в упорядоченный массив

```
function addar (v,t)
 { var n=v.length
   var j=n
   while (j > 0)
     { if (t < v[j-1])
        {v[j]=v[j-1]; j--}
       else
        {v[j]=t; break}
     }
   if (j==0) v[0]=t
   return v
 }
```

Если добавляемый элемент меньше первого элемента массива, то весь массив следует сдвинуть на один элемент вниз, и вне цикла на первое место поместить заданный элемент.

Удаление элемента массива

Напишем сценарий, удаляющий из упорядоченного массива элемент, значение которого совпадает с заданным. Если таких элементов несколько, то требуется удалить элемент с наименьшим индексом.

При решении задачи поступим следующим образом. Воспользовавшись функцией бинарного поиска, определим индекс элемента, совпадающего с удаляемым элементом. Если такого элемента нет, то задача решена, а в противном случае, сдвигаем "хвост" массива, начиная с `k`-го элемента на одну позицию вверх. HTML-код с описанными в нем функциями представлен в листинге 14.12.

Листинг 14.12. Удаление элемента из упорядоченного массива

```
<HTML>
  <HEAD>
    <TITLE>Удаление элемента из упорядоченного массива</TITLE>
    <script language="JavaScript">
```

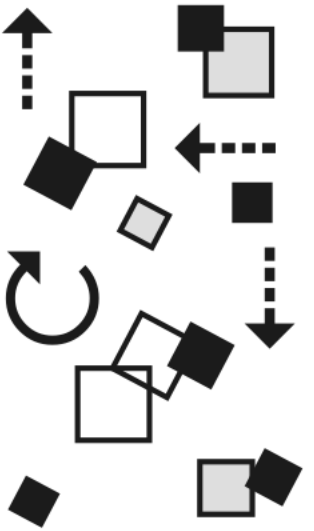
```
    <!-- //
      var v=new Array(4,5,15,17,31,52,72,79,81,83,89,91,93,94,98)
      function test(obj)
       { obj.data1.value=v }
      function binsear(v,t)
       { var i=0
         var j=v.length-1
         var k
         while (i < j)
          { k=Math.round((i+j)/2+0.5)-1
            if (t <= v[k])
              j=k
            else
              i=k+1
          }
         if (v[i] == t)
           { return i }
         else return -1
       }
      function delar (v,t)
       { var n=v.length
         var k=binsear(v,t)
         if (k != -1)
          { for (var i=k; i <= n-1; i++)
              v[i]= v[i+1]
            v.length=n-1
          }
         return v
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Удаление элемента из упорядоченного массива</H4>
    <FORM name="form1">
<pre>
Массив:      <input type="text" size=40 name="data1"  ><hr>
Элемент:     <input type="text" size=20 name="data2"  ><hr>
```

```
Результат поиска:  <input type="text" size=40 name="result" ><hr>
</pre>
 <input type="button" value=Определить
    onClick="test(form1); form1.result.value= delar
(v,Number(form1.data2.value))">
  <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Задан одномерный массив вещественных чисел. Напишите сценарий, который определяет число положительных элементов массива.
2. Задан одномерный массив вещественных чисел. Напишите сценарий, который определяет число отрицательных элементов.
3. Задан одномерный массив целых чисел. Напишите сценарий, который определяет число минимальных элементов.
4. Задан одномерный массив целых чисел. Напишите сценарий, который определяет число элементов, кратных 7.
5. Задан одномерный массив целых чисел. Напишите сценарий, который определяет номер последнего минимального значения.
6. Напишите сценарий, который определяет номер первого максимального значения в одномерном массиве целых чисел.
7. Напишите сценарий, при работе которого определяется число элементов одномерного массива, совпадающих с заданным.
8. Напишите сценарий, при работе которого определяется, есть ли в массиве элементы, значения которых совпадают.
9. Напишите сценарий, определяющий все ли элементы массива различны.
10. Напишите сценарий, определяющий максимальную по длине неубывающую последовательность.
11. Напишите сценарий, который объединяет два упорядоченных массива таким образом, что в результирующем массиве все элементы различны.
12. Напишите сценарий, который по двум массивам строит третий, являющийся пересечением заданных.
13. Напишите сценарий, который определяет, сколько различных чисел в заданном массиве.

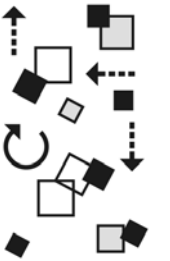
14. Напишите сценарий, который для заданного массива целых чисел определяет длину K самой длинной "пилообразной" последовательности идущих подряд чисел: $X_{p+1} < X_{p+2} > X_{p+3} < \ldots > X_{p+k}$.
15. Напишите сценарий, который для заданных двух массивов X из n элементов и Y из k элементов определяет, существуют ли в X подряд идущие элементы, для которых верно $X_{i+1} = Y_1$, $X_{i+2} = Y_2$, ..., $X_{i+k} = Y_k$.
16. Задана формула вида `a1?a2?...?an`, где `ai` — либо истина, либо ложь. Вместо знака вопроса (`?`) может стоять либо дизъюнкция, либо конъюнкция. Напишите сценарий, определяющий все комбинации знаков, при которых формула получает заданное значение.



ЧАСТЬ III

Решение вычислительных задач с использованием JavaScript

Глава 15



Процедурный тип данных и функция *eval*

Общие сведения

Предположим, что нам требуется написать программу, которая определяет значение функции в заданной точке. График функции представлен на рис. 15.1. Задачи подобного типа рассматривались ранее. Можно описать функцию `f1`, которая по заданному значению `x` будет выдавать соответствующее значение `y`, например, так:

```
function f1(x)
 { var y
   y=x*x*x-1
   return y }
```

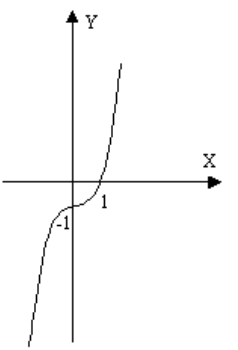


Рис. 15.1. График функции `f1`

Если использовать методы объекта `Math`, то функцию `f1` можно описать следующим образом:

```
function f1(x)
 { var y
   y=Math.pow(x,3)-1
   return y }
```

Когда аналогичную задачу требуется решить для функции, график которой представлен на рис. 15.2, можно описать функцию `f2`:

```
function f2(x)
 { var y
   y=(x-1)*(x-1)+1
   return y }
```

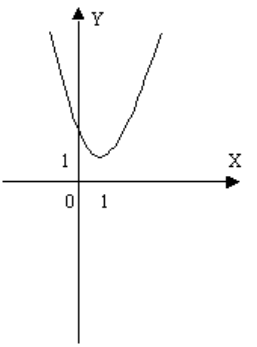


Рис. 15.2. График функции `f2`

Описание функции `f2` может быть и таким:

```
function f2(x)
 { var y
   y= Math.pow(x-1,2)+1
   return y }
```

Во многих языках программирования разрешается рассматривать функции `f1` и `f2` как процедурные константы, а в программе описать переменную процедурного типа, значение которой во время выполнения программы может меняться. Такие переменные связываются во время выполнения программы с разными процедурами. Процедурные константы и переменные можно использовать в качестве фактических параметров. В нашем случае ситуация была бы такой: в программе описана функция, которая вычисляет значение в точке, определяемой пользователем. Функция, задаваемая пользователем, передается в качестве параметра. Иногда в аналогичных случаях говорят, что процедура или функция рассматриваются как данные.

Идея единства данных и подпрограмм занимает одно из основных мест в объектно-ориентированном программировании. В языке JavaScript идея процедурных типов данных реализована с помощью функции `eval`. Функция `eval` получает в качестве параметра строку. Если строка представляет собой выражение языка JavaScript, то вычисляется ее значение и возвращается как результат функции `eval`.

Вычисление значения пользовательской функции

Напишем программу, которая определяет значение функции, задаваемой пользователем, в некоторой заданной точке.

Функция, точка и вычисленное значение указываются в текстовых полях формы, как показано на рис. 15.3.

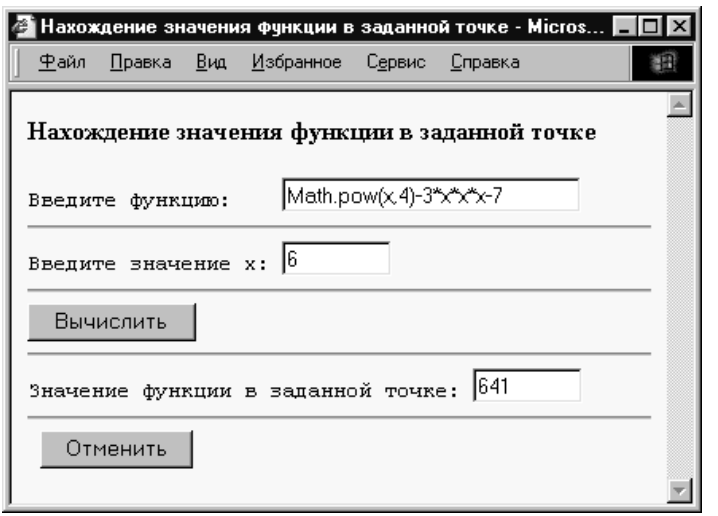


Рис. 15.3. Значение задаваемой функции в точке

Для того чтобы определить значение функции, график которой изображен на рис. 15.1, достаточно задать функцию формулой `x*x*x-1` или формулой `Math.pow(x,3)-1`. Если же нас интересует значение в некоторой точке функции, график которой изображен на рис. 15.2, то надо ввести формулу `(x-1)*(x-1)+1` или `Math.pow(x-1,2)+1`.

Функция может иметь такой вид, что ее нельзя описать с помощью лишь одного выражения. Например, для того, чтобы описать функцию, график которой представлен на рис. 15.4, следует воспользоваться условным оператором:

```
if (Math.abs(x)<1)
 {y=Math.sqrt(1-x*x)}
else
  if (x <-1) {y=x+1}
  else {y=x-1}
```

Для получения значения функции, график которой изображен на рис. 15.4, следует воспользоваться условным оператором:

```
if (Math.abs(x)<1)
 {y= x*x}
else
  if (x <-1) {y=x+2}
  else {y=1}
```

Метод `eval` может быть использован для выполнения операторов языка JavaScript, включенных в строку параметра. Таким образом, при задании функции можно вводить не только формулу, как мы поступали ранее, но и операторы. Для получения значений в заданных точках функций, графики которых изображены на рис. 15.4 и 15.5, можно ввести в поле формы соответствующие условные операторы. Таким образом, используя метод `eval`, мы получили возможность задавать в качестве исходных данных функцию, и тем самым решать целый класс однотипных задач.

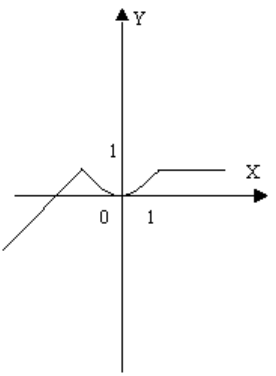


Рис. 15.4. График сложной функции (вариант 1)

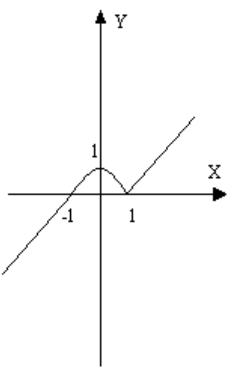


Рис. 15.5. График сложной функции (вариант 2)

Формирование таблицы значений пользовательской функции

Напишем программу, которая формирует таблицу значений задаваемой пользователем функции. Кроме функции указывается начальное и конечное значение аргумента, и шаг его изменения.

С помощью цикла вычисляется точка, значение функции в точке, и формируется строка, которая затем составит таблицу. После завершения работы цикла строка помещается в поле формы, предназначенное для отображения результата. Полностью HTML-код представлен в листинге 15.1.

Листинг 15.1. Построение таблицы значений функции

```
<HTML>
  <HEAD>
    <TITLE>Построение таблицы значений функции</TITLE>
    <script language="JavaScript">
    <!-- //
      function  valfunc (obj )
       { var fs=obj.func.value
         var m=Number(obj.left.value)
         var n=Number(obj.rig.value)
         var h=Number(obj.by.value)
         var y
         var x
         var s="x          f(x)"+"\r\n"
         for (var i=m; i<=n; i=i+h)
          {x=i; y=eval(fs); s+=x+"           "+y+"\r\n"}
         obj.res.value=s
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H4>Построение таблицы значений функции</H4>
    <FORM name="form1">
<pre>
Введите функцию:   <input type="text" size=20 name="func"><hr>
Введите левый конец интервала:   <input type="text" size=8
name="left"><hr>
Введите правый конец интервала:  <input type="text" size=8 name="rig"><hr>
Введите шаг изменения аргумента: <input type="text" size=8 name="by"><hr>
<input type="button" value=Вычислить onClick="valfunc(form1)"><hr>
Таблица значений функции:
 <textarea  cols=40 rows=10 name="res"></textarea><hr>
 <input type="reset" value=Отменить>
</pre>
</FORM></BODY></HTML>
```

На рис. 15.6 приведен результат выполнения сценария.

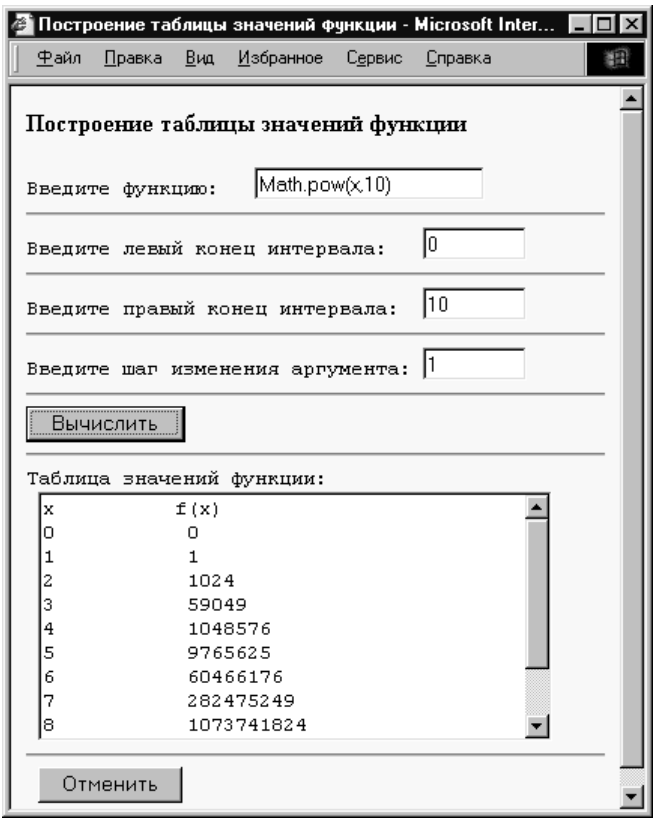


Рис. 15.6. Таблица значений функции, задаваемой пользователем в текстовом поле формы

Для того чтобы получить значение функции $y = 0{,}5 \times \cos(x)$ на интервале [0; 1] с шагом 0,05, следует заполнить соответствующие поля формы, причем, функцию надо задать как выражение JavaScript: `0.5*Math.cos(x)`.

Определение принадлежности точки некоторой области

На плоскости определяется некоторая область. Требуется написать программу, которая для произвольной точки, заданной своими координатами, определяет, принадлежит ли точка области.

Мы решали ранее подобные задачи, только описание каждой из областей производили с помощью соответствующей функции. Использование метода `eval` позволит нам написать программу, которая обрабатывает параметр, определяющий область плоскости. Область плоскости задается пользовате-

лем в качестве исходных данных. Рассмотрим варианты, изображенные на рис. 15.7—15.9.

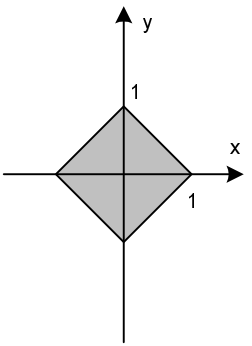


Рис. 15.7. Область плоскости P1

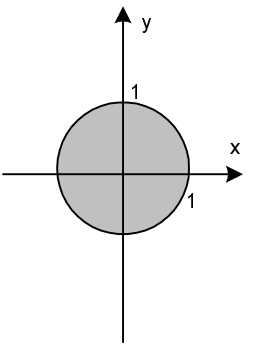


Рис. 15.8. Область плоскости P2

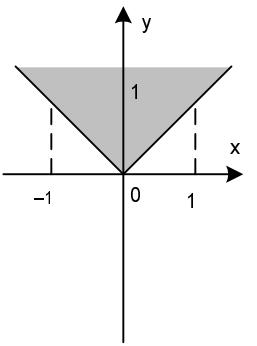


Рис. 15.9. Область плоскости P3

Для заштрихованных областей на приведенных рисунках формулы, описывающие область, могут быть следующими:

- `Math.abs(y)+ Math.abs(x)<=1`
- `x*x+y*y<=1`
- `y>= Math.abs(x)`

Полностью программа приведена в листинге 15.2.

Листинг 15.2. Принадлежит ли точка области, задаваемой пользователем

```
<HTML>
  <HEAD>
    <TITLE>Принадлежит ли точка определенной области</TITLE>
    <script language="JavaScript">
    <!-- //
    function  valfunc(obj)
     { var fs=obj.func.value
       var x=Number(obj.x.value)
       var y=Number(obj.y.value)
       var z=eval(fs)
       var s="принадлежит области"
       if (!z) s= "не " +s
         obj.res.value=s
     }
```

```
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H4>Принадлежность точки выделенной области плоскости</H4>
    <FORM name="form1">
<pre>
Введите формулу, описывающую область:   <input type="text" size=40
name="func"><hr>
Введите x:   <input type="text" size=8 name="x"><hr>
Введите y:   <input type="text" size=8 name="y"><hr>
<input type="button" value=Вычислить onClick="valfunc(form1)"><hr>
Точка с заданными координатами <input type="text" size=30 name="res"><hr>
 <input type="reset" value=Отменить>
</pre>
    </FORM>
  </BODY>
</HTML>
```

Вычисление корня уравнения методом итераций

Напишем программу, которая определяет корень уравнения вида $x = f(x)$ с начальным приближением t и точностью *eps* методом итераций.

Мы писали ранее программу, которая вычисляла корень конкретного уравнения. Напишем программу (листинг 15.3), в которой функция, представляющая правую часть уравнения, указывается пользователем при заполнении соответствующих полей формы. Используя программу, мы можем задавать различные функции, изменять точность и начальные приближения, анализировать результат.

Листинг 15.3. Нахождение корня уравнения вида $x = f(x)$ методом итераций

```
<HTML>
  <HEAD>
    <TITLE>Нахождение корня уравнения вида x = f(x)
          методом итераций</TITLE>
    <script language="JavaScript">
    <!--//
      function iter(obj)
```

```
       { var d=document
         var fs=obj.func.value
         var t=Number(obj.tv.value)
         var eps=Number(obj.epsv.value)
         var x=t;
         var y=eval(fs)
         while (Math.abs(y-x) >= eps)
          { x=y ; y= eval(fs) }
            obj.res.value= x
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H4>Нахождение корня уравнения вида x = f(x) методом итераций</H4>
    <FORM name="form1">
<pre>
Введите функцию f(x):          <input type="text" size=20 name="func"><hr>
Введите начальное приближение: <input type="text" size=8 name="tv"><hr>
Введите требуемую точность:    <input type="text" size=8 name="epsv"><hr>
<input type="button" value=Вычислить  onClick="iter(form1)"><hr>
Корень уравнения равен: <input type="text" size=20 name="res"><hr>
<input type="reset" value=Отменить>
</pre></FORM></BODY></HTML>
```

Вычисление корня уравнения методом деления отрезка пополам

Напишите программу, которая находит корень уравнения вида $f(x) = 0$ методом деления отрезка пополам с заданной точностью *eps*.

Нас интересует интервал $[a, b]$, на котором функция непрерывна и монотонна и на концах интервала принимает значения разных знаков. Тогда на этом интервале функция имеет ровно один корень и его можно приблизительно определить следующим образом. Разделить интервал $[a, b]$ на две части точкой $m = (a + b) / 2$ и вычислить значение $f(m)$. После этого из двух получавшихся интервалов $[a, m]$ и $[m, b]$ рассматривать тот, на концах которого функция принимает значения разных знаков, именно он содержит корень. Продолжить процесс поиска корня для вновь образованного интервала. Процесс будет закончен, когда длина очередного интервала станет мень-

ше значения *eps*. Следовательно, любая точка интервала будет представлять значение корня с точностью *eps*.

Напишем сценарий, который для уравнения вида $f(x) = 0$ находит корень с точностью *eps* методом деления отрезка пополам. При этом пользователь должен задать функцию, точность и границы интервала (листинг 15.4).

Листинг 15.4. Метод деления отрезка пополам

```
<HTML>
  <HEAD>
    <TITLE>Метод деления отрезка пополам</TITLE>
    <script language="JavaScript">
    <!--//
      function binroot (obj)
       { var fs=obj.func.value
         var a=Number(obj.left.value)
         var b=Number(obj.right.value)
         var eps=Number(obj.epsv.value)
         var x=a
         var fa = eval(fs)
         x=b
         var fb = eval(fs)
         var fc
         var c
         if (fa*fb < 0)
          { while (Math.abs(b-a) >= eps)
            { c= (a+b)/2;
              x=c; fc= eval(fs)
              if (fa*fc < 0)
                {b=c; fb= fc }
              else
                {a=c; fa=fc }
            }
           obj.res.value= (a+b)/2
           x=  (a+b)/2
           var tes = eval(fs)
           obj.test.value= tes
          }
         else document.write("на концах отрезка значения одного знака")
       }
```

```
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H4> Метод деления отрезка пополам</H4>
    <FORM name="form0">
<pre>
Введите функцию:              <input type="text" size=30 name="func"><hr>
Введите левый конец отрезка:  <input type="text" size=10 name="left"><hr>
Введите правый конец отрезка: <input type="text" size=10 name="right"><hr>
Задайте требуемую точность:   <input type="text" size=10 name="epsv"><hr>
<input type="button" value=Вычислить  onClick="binroot(form0)"><hr>
Корень уравнения равен: <input type="text" size=30 name="res"><hr>
Тест: <input type="text" size=30 name="test"><hr>
<input type="reset" value=Отменить>
</pre>
    </FORM>
  </BODY>
</HTML>
```

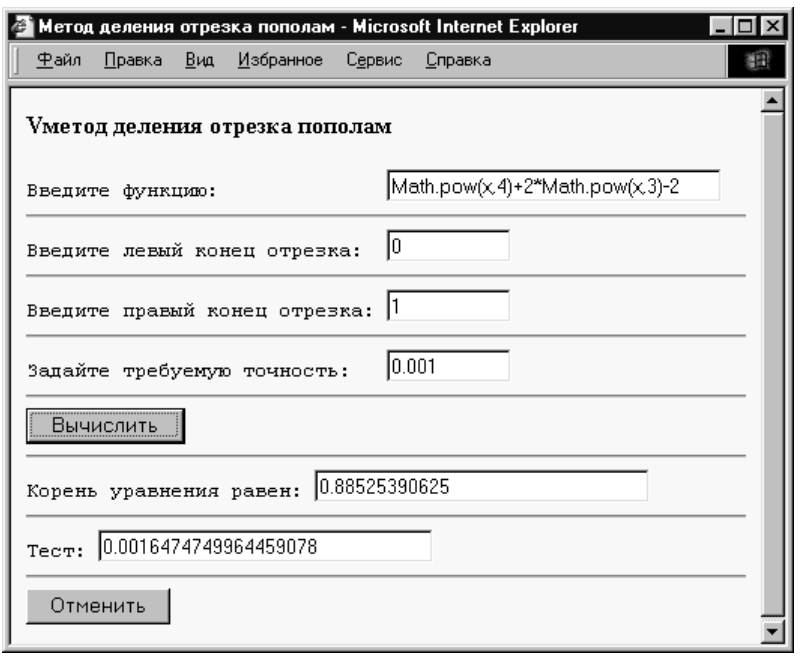


Рис. 15.10. Поиск корня уравнения методом половинного деления

Для того чтобы найти корень уравнения $y = x^4 - 3x + 1$ с точностью 0,001 можно взять интервал [0, 1], т. к. на концах этого интервала функция принимает значения разных знаков. Результат работы сценария приведен на рис. 15.10.

Вычисление интеграла

Напишите программу, которая приближенно вычисляет интеграл

$$\int_a^b f(x)dx.$$

Приближенное значение интеграла можно получить, если на промежутке [*a*, *b*] функцию *f* заменить кусочно-линейной функцией, т. е. превратить задачу о вычислении площади криволинейной трапеции в задачу вычисления площадей прямолинейных трапеций. Чем больше количество элементарных отрезков, тем более точные вычисления. Разобьем отрезок [*a*, *b*] на равные промежутки $a = x_0, x_1, ..., x_n = b$. Площадь трапеции, на отрезке $[x_{i-1}, x_i]$ определяется формулой

$$\frac{f(x_i) + f(x_{i-1})}{2} \times (x_i - x_{i-1}).$$

Если интервал [*a*, *b*] разбит на равные промежутки, то длина такого промежутка равна $p = (b - a)/n$, а точки определяются по формуле $x_i = a + i \times h$. Формула приближенного вычисления интеграла аналогично формуле трапеций может быть записана так:

$$\int_a^b f(x)dx = \frac{f(a) + f(b)}{2} \times h + \sum_{i=1}^{n-1} f(x_i) \times h.$$

Задачу будем решать следующим образом. Интеграл вычислять по формуле трапеций, последовательно уменьшая шаг в два раза, пока последние вычисленные приближения интеграла не станут меньше чем заданное значение *eps*. Кроме того, значение подынтегральной функции в каждой точке будем вычислять лишь один раз. Пусть рассматриваемым отрезком будет интервал [0, 1], а первоначальное разбиение $n = 10$. Значения подынтегральной функции должны быть подсчитаны в точках: $\frac{1}{10}, \frac{2}{10}, ..., \frac{9}{10}$. При следующем разбиении $n = 20$, значения функции должны считаться в точках: $\frac{1}{20}, \frac{2}{20}, \frac{3}{20}, ..., \frac{18}{20}, \frac{19}{20}$. Для некоторых значений в этом интервале значения функции уже считались на предыдущем шаге. Поэтому программу следует составить так, чтобы функция вычислялась лишь в точках $\frac{1}{20}, \frac{3}{20}, ..., \frac{19}{20}$.

В функции `integral` использованы следующие переменные. В переменной `fs` хранится подынтегральная функция, которую задает пользователь, заполняя поля предложенной формы. Переменные `a` и `b` служат для задания промежутка интегрирования. Переменная `n` задает начальное разбиение. Для хранения двух последовательных приближений предназначены переменные `i1` и `i2`. Напишем программу решения задачи (листинг 15.5).

Листинг 15.5. Вычисление значения интеграла по формуле трапеций

```
<HTML>
  <HEAD>
    <TITLE>Вычисление значения интеграла по формуле трапеций</TITLE>
    <script language="JavaScript">
    <!-- //
      function  integral(obj)
       { var d=document
         var fs=obj.func.value
         var a=Number(obj.left.value)
         var b=Number(obj.right.value)
         var n=Number(obj.nv.value)
         var eps=Number(obj.epsv.value)
         var i1;   var i2;   var h;   var s;   var i
         var fcur
         x=a
         var fa=eval(fs)
         x=b
         var fb=eval(fs)
         s=(fa+fb)/2; h=(b-a)/n
         for (i=1; i<= n-1; i =  i+1)
          {x=a+i*h ; fcur=eval(fs); s=s+fcur}
         i1=s*h
         n=n*2; h=h/2
         for (i=1; i<= n-1; i = i+2)
          {x=a+i*h; fcur=eval(fs); s=s+fcur}
         i2=s*h
         while (Math.abs(i1-i2) >= eps)
          {i1=i2;
           n=n*2; h=h/2
           for (i=1; i<= n-1; i = i+2)
```

```
            {x=a+i*h; fcur= eval(fs); s=s+fcur}
           i2=s*h
          }
         obj.res.value=i2
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor="F8F8FF">
    <H4>Численное интегрирование.
        Вычисление интеграла методом трапеций</H4>
    <FORM name="form0">
<pre>
Введите подынтегральную функцию:  <input type="text" size=20
name="func"><hr>
Введите левый конец промежутка интегрирования:  <input type="text" size=8
name="left"><hr>
Введите правый конец промежутка интегрирования: <input type="text" size=8
name="right"><hr>
Задайте начальное разбиение:                    <input type="text" size=12
name="nv"><hr>
Задайте требуемую точность:                     <input type="text" size=12
name="epsv"><hr>
<input type="button" value=Вычислить  onClick="integral(form0)"><hr>
Значение интеграла равно:                       <input type="text" size=20
name="res"><hr>
 <input type="reset" value=Отменить>
</pre>
</FORM></BODY></HTML>
```

Определение типа выравнивания изображения

Напишите программу, которая позволяет определить действие параметра `align` при вставке изображения в строку. Пользователь выбирает интересующий параметр, и в правой части появляется описание действий при выбранном параметре, как на рис. 15.11.

При выборе значения параметра выравнивания изображения возникает событие `focus`, реакцией на которое является вызов функции `set`. Второй параметр функции `set` применяется при формировании имени строковой переменной, значение которой помещается в соответствующее поле формы.

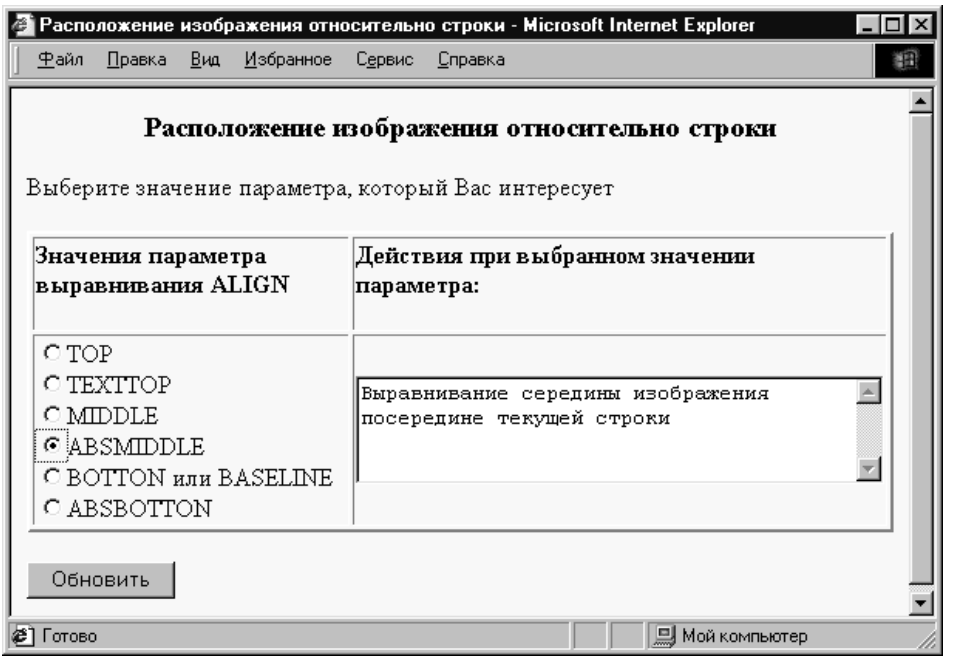


Рис. 15.11. Значения параметров горизонтального выравнивания изображения

```
function set(obj,i)
 {obj.res.value=eval("s"+i)}
```

Если написать конструкцию

```
obj.res.value="s"+i
```

то в зависимости от значения `i` будет выведена строка `s0`, `s1`, ..., `s5`. Полностью HTML-код приведен в листинге 15.6.

Листинг 15.6. Объяснение действия при выбранном значении параметра

```
<HTML>
  <HEAD>
    <TITLE>Расположение изображения относительно строки</TITLE>
    <script>
    <!--
      var s0="Верхняя граница изображения выравнивается"+
             "по самому высокому  элементу текущей строки"
      var s1="Верхняя граница изображения выравнивается "+
             "по самому высокому текстовому элементу текущей строки"
```

```
    var s2="Выравнивание середины изображения по базовой линии "+
           "текущей строки"
    var s3="Выравнивание середины изображения посередине "+
           "текущей строки"
    var s4="Выравнивание нижней границы изображения "+
           "по базовой линии текущей строки"
    var s5="Выравнивание нижней границы изображения "+
           "по нижней границе текущей строки"
    function set(obj,i)
     {obj.res.value=eval("s"+i)}
  //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <H3 align=center>Расположение изображения относительно строки</H3>
  Выберите значение параметра, который Вас интересует
  <FORM name="form1">
    <TABLE border=2>
      <TR><TD><h4>Значения параметра выравнивания ALIGN</h4>
          <TD><h4>Действия при выбранном значении параметра:</h4>
      </TR>
      <TR><TD>
          <input type="radio" name="m" value=0
                 onfocus="set(form1,0)">TOP <br>
          <input type="radio" name="m" value=1
                 onfocus="set(form1,1)">TEXTTOP<br>
          <input type="radio" name="m" value=2
                 onfocus="set(form1,2)">MIDDLE<br>
          <input type="radio" name="m" value=3
                 onfocus="set(form1,3)">ABSMIDDLE<br>
          <input type="radio" name="m" value=4
                 onfocus="set(form1,4)">BOTTON или BASELINE<br>
          <input type="radio" name="m" value=5
                 onfcus="set(form1,5)">ABSBOTTON<br></TD>
          <TD><textarea name="res" cols=40 rows=4>
              </textarea><br></TD></TR></TABLE><br>
          <input type="reset" value="Обновить">
  </FORM></BODY></HTML>
```

Упражнения

1. Напишите сценарий с функцией, которая определяет приближенное значение корня уравнения $f(x) = 0$ методом хорд с точностью *eps*. Найдите корни уравнения $e^x + x - 2 = 0$ на промежутке [0, 1] с точностью 10^{-4}.
2. Напишите сценарий с функцией, определяющей корень уравнения $f(x) = 0$ на промежутке $[a, b]$ с заданным начальным приближением x_0 методом Ньютона. Найдите с помощью описанной функции корни уравнения $\mathrm{tg}x - p \times x = 0$ на промежутке [0; 3/2] для $p = 1{,}4(0{,}2)2{,}1$, $x_0 = 1{,}5$.
3. Напишите сценарий с функцией, приближенно вычисляющей интеграл по формуле прямоугольников:

$$\int_a^b f(x)dx = h \times \left[f(x_1) + f(x_2) + ... + f(x_n) \right],$$

   где $h = (b - a)/n$, $x_i = a + i \times h - h/2$ при заданном значении n. Подынтегральную функцию, промежуток и разбиение пользователь должен задавать с помощью формы.
4. Напишите сценарий, который дает возможность пользователю выбрать график, для которого требуется определить значение функции при заданном значении аргумента. В сценарии следует предусмотреть проверку правильности написанного фрагмента для заданного графика. После выполнения всей работы (просмотра всех графиков) пользователю выставляется оценка.
5. Напишите сценарий, который дает возможность пользователю выбрать рисунок из набора заданных изображений. Пользователю для каждого из рисунков требуется написать фрагмент программы, определяющий, принадлежит ли точка с заданными координатами заштрихованной области. В сценарии предусмотреть проверку правильности написанного фрагмента для заданного графика. После выполнения работы (просмотра всех изображений) пользователю выставляется оценка.
6. Напишите сценарий, согласно которому пользователь выполняет задание, а затем ему выставляется оценка. Задание таково: напишите выражение, которое выдает значение функции при заданном значении аргумента. Пользователь выбирает график, вводит выражение, нажимает на кнопку **Готово**, выбирает следующий график и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы выставляется оценка.
7. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время.

После окончания работы выставляется оценка. При выполнении каждого задания требуется написать выражение, истинное тогда и только тогда, когда выполняется одно из следующих условий:

- хотя бы одна из переменных *a*, *b*, *c* имеет значение `false`;
- ровно одна из переменных *a*, *b*, *c* имеет значение `false`;
- ровно две переменные из переменных *a*, *b*, *c*, имеют значение `false`;
- все переменные *a*, *b*, *c* имеют одинаковое значение;
- ни одна из логических переменных *a*, *b*, *c* не истинна;

8. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы ставится оценка. При выполнении каждого задания требуется написать выражение, истинное тогда и только тогда, когда выполняется одно из следующих условий:
   - остаток от деления целого числа *n* на 7 равен либо 3, либо 7;
   - два числа нечетны и разность между ними равна 2;
   - сумма цифр заданного трехзначного числа *n* совпадает с самим числом;
   - заданное четырехзначное число *n* является палиндромом;
   - заданное трехзначное число *n* является числом Армстронга;
   - первая и последняя цифра четырехзначного числа *n* совпадают;
   - максимальная цифра четырехзначного числа *n* делится на минимальную цифру без остатка.
9. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы ставится оценка. При выполнении каждого задания требуется написать выражение, истинное тогда и только тогда, когда выполняется одно из следующих условий:
   - последовательность из пяти чисел *a*, *b*, *c*, *d*, *e* образует арифметическую прогрессию;
   - пять чисел образуют возрастающую последовательность;
   - пять чисел образуют строго возрастающую последовательность;
   - пять чисел образуют строго убывающую последовательность;
   - пять чисел образуют возрастающую последовательность;
   - в последовательности есть два одинаковых числа;

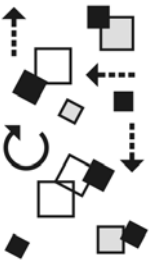
- в последовательности три одинаковых числа;
- в последовательности нет одинаковых чисел;
- в последовательности два равных числа расположены рядом.

10. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы ставится оценка. При выполнении каждого задания требуется написать выражение, истинное тогда и только тогда, когда выполняется одно из следующих условий:
    - в последовательности из четырех чисел *a*, *b*, *c*, *d* только одно максимальное число;
    - в последовательности из четырех чисел *a*, *b*, *c*, *d* чисел ровно два максимальных числа;
    - в последовательности из четырех чисел *a*, *b*, *c*, *d* максимальное число совпадает с квадратом минимального числа;
    - в последовательности из четырех чисел *a*, *b*, *c*, *d* первое минимальное число стоит на нечетном месте;
    - в последовательности из четырех чисел *a*, *b*, *c*, *d* максимальное число и минимальное число идут подряд;
    - значение *a* является наибольшим из четырех попарно различных значений *a*, *b*, *c*, *d*;
    - разность между максимальным и минимальным числом является числом, кратным 7.
11. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы ставится оценка. При выполнении каждого задания требуется написать выражение, истинное тогда и только тогда, когда выполняется одно из следующих условий:
    - последовательность из пяти символов *a*, *b*, *c*, *d*, *e* является идентификатором;
    - последовательность из пяти символов *a*, *b*, *c*, *d*, *e* является палиндромом;
    - в последовательности из пяти символов *a*, *b*, *c*, *d*, *e* есть одинаковые буквы;
    - в последовательности из пяти символов *a*, *b*, *c*, *d*, *e* все буквы различны;
    - буквы расположены в алфавитном порядке в последовательности из пяти букв;

- в последовательности символов *a*, *b*, *c*, *d*, *e* есть хотя бы одна цифра;
- в последовательности *a*, *b*, *c*, *d*, *e* есть две цифры, стоящие рядом;
- последовательность *a*, *b*, *c*, *d*, *e* представляет целое число, которое при делении на 7 дает в остатке 5;
- последовательность *a*, *b*, *c*, *d*, *e* представляет целое число, кратное 3;
- в последовательности *a*, *b*, *c*, *d*, *e* есть ровно три буквы;
- последовательность *a*, *b*, *c*, *d*, *e* представляет целое число, кратное 9.

12. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После выполнения работы ставится оценка. В заданиях, предлагаемых для тестирования, приняты следующие соглашения. Четыре числа трактуются как оценки за сессию. Требуется написать выражение, истинное тогда и только тогда, когда:
    - все экзамены сданы на 5;
    - экзамены сданы на оценки 4 и 5, но не на все пятерки;
    - имеется хотя бы одна тройка, но нет двоек;
    - имеется хотя бы одна двойка;
    - средний балл за все экзамены в пределах 3,5 и 4,5;
    - сумма баллов за сессию либо меньше 3,5, либо более 4,75;
    - среди оценок нет одинаковых;
    - ровно две четверки;
    - ровно три тройки.
13. Напишите сценарий, в результате выполнения которого выбирается задание из предложенного списка, затем пользователь вводит выражение, удовлетворяющее условию задания, и т. д., пока не будут выполнены все задания. На выполнение всего задания отводится определенное время. После окончания работы ставится оценка. В заданиях, предлагаемых для тестирования, предполагается выполнение следующих условий. Три числа *a*, *b*, *c* трактуются как длины сторон. Определите:
    - можно ли построить треугольник с заданными длинами сторон;
    - является ли треугольник равносторонним;
    - является ли треугольник равнобедренным; является ли треугольник разносторонним;
    - является ли треугольник прямоугольным; является ли треугольник остроугольным; является ли треугольник тупоугольным.

Глава 16



Рекурсивные методы

При создании программы для решения сложной задачи программист обычно разбивает ее на подзадачи, чтобы для каждой такой подзадачи написать подпрограмму, а затем объединить все написанные подпрограммы в программу. Если подзадача достаточно сложная, то для решения может оказаться полезным разбить ее еще на более мелкие подзадачи и программировать каждую из них отдельно и т. д. Часто бывает, что в процессе такого разбиения задача сводится к самой себе. Если при этом исходные данные становятся проще, то этот процесс можно продолжать до тех пор, пока исходные данные не окажутся настолько простыми, что решение задачи для них станет тривиальным. Приведем примеры, демонстрирующие такой принцип решения.

Перевод десятичного натурального числа в двоичное

Напишем сценарий, который для любого натурального числа строит его двоичное представление.

Опишем функцию `recbin`, которая получает в качестве параметра целое положительное число `n`, и формирует строку, являющуюся двоичным представлением `n`. Эту задачу можно свести к самой себе. Если в двоичном представлении `n` содержится `k>0` двоичных цифр, то первые `k-1` цифр представляют число, являющееся частным от деления `n` на `2`, а последнюю цифру легко узнать, проверив четность числа `n`. Если `к=1`, то `n=1`, то и двоичное представление `n` равно `1`, и этот случай рассмотрим отдельно. В теле функции `recbin` есть обращение к ней самой, но с другими, более простыми параметрами. При каждом вызове функции значение параметра уменьшается, поэтому наступит момент, когда параметр станет равным `1` и решение задачи будет тривиальным.

HTML-код приведен в листинге 16.1.

Листинг 16.1. Рекурсивные методы. Двоичное представление числа

```
<HTML>
  <HEAD>
    <TITLE>Рекурсивные методы. Двоичное представление числа</TITLE>
    <script language="JavaScript">
    <!--
      function recbin(n)
       { var s
         if (n==1) s="1"
         else
          { s=recbin((n-n%2)/2)
            s+=n%2
          }
         return s
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Рекурсивные методы. Двоичное представление числа</H4>
    <FORM name="form1">
<pre>
Введите натуральное число: <input type="text" size=12 name="num">
Двоичное представление:    <input type="text" size=12 name="res">
</pre>
 <input type="button" value=Выполнить
    onClick="this.form.res.value=recbin(Number(this.form.num.value))">
 <input type="reset" value=Отменить>
</FORM>
</BODY>
</HTML>
```

Вычисление факториала рекурсивным методом

Напишем программу, которая по любому натуральному n вычисляет $n!$ так, как показано на рис. 16.1.

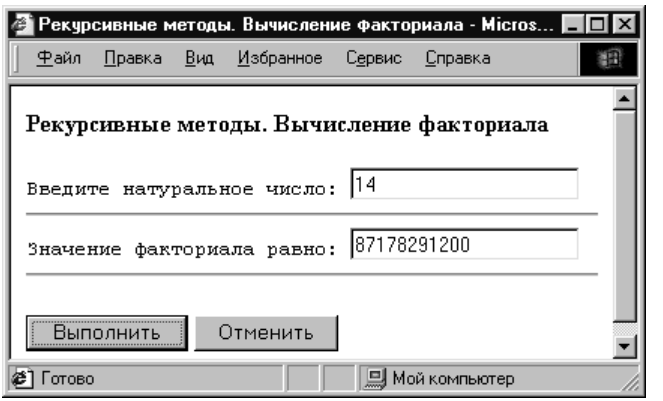


Рис. 16.1. Факториал натурального числа

Мы уже решали подобную задачу при демонстрации итеративных методов. Опишем рекурсивную функцию вычисления факториала. В функции один параметр — натуральное число n. Тривиальный случай — это вычисление значения 1!. Заметим, что верно следующее соотношение $n! = n \times (n - 1)!$, поэтому можно свести задачу вычисления $n!$ к решению той же задачи, но с другим, более "простым" параметром.

HTML-код представлен в листинге 16.2.

Листинг 16.2. Рекурсивные методы. Вычисление факториала

```
<HTML>
  <HEAD>
    <TITLE>Рекурсивные методы. Вычисление факториала</TITLE>
    <script language="JavaScript">
    <!-- //
      function fact(n)
       { var f=1
         if (n==1) f=1
         else f=n*fact(n-1)
         return f
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Рекурсивные методы. Вычисление факториала</H4>
    <FORM name="form1">
```

```
<pre>
Введите натуральное число: <input type="text" size=20 name="num"><hr>
Значение факториала равно: <input type="text" size=20 name="res"><hr>
</pre>
 <input type="button" value=Выполнить
       onClick="this.form.res.value=fact(this.form.num.value)">
 <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Пусть `n=3`. Как будет выполняться вызов `fact(n)`? На рис. 16.2 представлена схема выполнения вызова функции. Стрелки указывают порядок вычисления. Сначала происходит движение по стрелкам вниз, указывающее на то, что происходит временное прерывание выполнения текущего вызова, и переход к выполнению вызова более низкого уровня, пока не будет получен тривиальный случай. Затем происходит подъем вверх, что означает возобновление прерванных вызовов. Первым возобновится выполнение такого вызова, который был прерван последним.

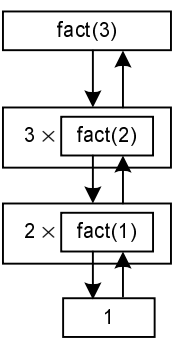


Рис. 16.2. Схема выполнения вызова рекурсивной функции

Определение чисел Фибоначчи рекурсивным методом

Напишем рекурсивную функцию, определяющую число Фибоначчи с номером *n*. По определению последовательности чисел Фибоначчи:

- $f_0 = f_1 = 1$;
- $f_i = f_i - 1 + f_i - 2$ при $i \geq 2$.

Можно написать функцию, основываясь на только что приведенном определении:

```
function fib(p,s,n)
 { var r
   if (n<=0)
     {r=s}
   else
     {r=fib(n-1) + fib(n-2)}
   return r
  }
```

В теле рекурсивной функции `fib` есть два обращения к ней самой с параметром `n-1` и `n-2`. При вычислении `fib(n-1)` будет вычисляться `fib(n-2)`, а затем значение это будет "забыто" и вычисляться заново при следующем рекурсивном вызове.

Для того чтобы избежать многих повторных вызовов используют прием, который получил название "накапливаемые" параметры. В рекурсивной функции `recfib` три параметра: два последних определенных числа Фибоначчи и номер того числа, которое нас интересует. В теле функции вызов `recfib(s,s+p,n-1)` означает переход к следующей паре полученных чисел Фибоначчи (параметры `s` и `s+p`), и тот факт, что номер искомого числа относительно найденных уменьшился на `1`.

```
function recfib(p,s,n)
 { var r
   if (n <=0)  {r=s}
   else
    { r=recfib(s,s+p,n-1) }
   return r
 }
```

Для того чтобы найти число Фибоначчи *n* с помощью функции `recfib`, надо выполнить вызов `recfib(1,1,n-1)`. Приведем полностью программу решения задачи (листинг 16.3).

Листинг 16.3. Рекурсивные методы. Числа Фибоначчи

```
<HTML>
  <HEAD>
    <TITLE>Рекурсивные методы. Числа Фибоначчи</TITLE>
    <script language="JavaScript">
```

```
    <!--
      function recfib(p,s,n)
       { var r
         if (n <=0)  {r=s}
         else
          { r=recfib(s,s+p,n-1) }
         return r
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Рекурсивные методы. Числа Фибоначчи</H4>
    <FORM name="form1">
<pre>
Введите номер числа: <input type="text" size=8 name="num">
Число в последовательности Фибоначчи:  <input type="text" size=8
name="res">
</pre>
<hr>
 <input type="button" value=Выполнить
    onClick="this.form.res.value=recfib(1,1,
             Number(this.form.num.value-1))">
 <input type="reset" value=Отменить>
    </FORM>
   </BODY>
</HTML>
```

Вычисление наибольшего общего делителя

Напишем программу, которая для двух заданных натуральных чисел определяет их наибольший общий делитель.

Рекурсивная функция `nod` имеет два параметра — числа `n` и `m`. Будем считать, что `n>m`. Если это не так, то параметры можно поменять местами. Если `m=0`, то задача решена, наибольший общий делитель совпадает со значением `n`. Сведение задачи к подзадаче основано на следующем соотношении:

```
nod(n,m)=nod(m,n%m)
```

Первый и второй параметры уменьшаются при каждом рекурсивном вызове, поэтому наступит момент, когда значение второго параметра станет равным `0`. Полностью текст программы приведен в листинге 16.4.

Листинг 16.4. Рекурсивные методы. Вычисление наибольшего общего делителя

```
<HTML>
  <HEAD>
    <TITLE>Рекурсивные методы.
           Вычисление наибольшего общего делителя</TITLE>
    <script language="JavaScript">
    <!-- //
      function  nod(n, m)
       { var r
         if (m==0)  r=n
         else r = nod(m, n%m)
         return r
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Рекурсивные методы. Вычисление наибольшего общего делителя</H4>
    <FORM name="form1">
<pre>
Число1:  <input type="text" size=10 name="num1"><hr>
Число2:  <input type="text" size=10 name="num2"><hr>
Наибольший общий делитель: <input type="text" size=10 name="res">
</pre><hr>
<input type="button" value=Выполнить
    onClick="this.form.res.value=nod(this.form.num1.value,
             this.form.num2.value)">
 <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Вычисление корня уравнения методом половинного деления с помощью рекурсии

Напишем рекурсивную процедуру поиска корня уравнения вида $f(x) = 0$ на промежутке $[a, b]$ методом деления отрезка пополам.

Предположим, что функция на отрезке $[a, b]$ непрерывна и монотонна и принимает на концах отрезка значения разных знаков.

Рекурсивная функция `fbin` зависит от трех параметров *a*, *b* и *eps*. Тривиальный случай состоит в том, что длина исследуемого отрезка меньше заданного значения *eps*. В этом случае любую точку отрезка можно принимать за приближенное значение корня. Если длина исследуемого участка более *eps*, то находится середина отрезка *c* и вычисляется значение функции $f(c)$. В зависимости от значения $f(c)$ продолжать поиск корня следует либо в промежутке $[a, c]$, либо в $[c, b]$. Если на концах промежутка $[a, c]$ значения функции противоположных знаков, то поиск корня осуществляется на этом промежутке с помощью рекурсивного вызова `fbin(a,c,eps)`, в противном случае нахождение корня обеспечит вызов `fbin(a,c,eps)`.

В листинге 16.5 описана функция `ftest`, которая осуществляет дополнительную проверку, вычисляя значения функции в найденной точке.

Листинг 16.5. Рекурсивные методы. Метод деления отрезка пополам

```
<HTML>
  <HEAD>
    <TITLE>Рекурсивные методы. Метод деления отрезка пополам</TITLE>
    <script language="JavaScript">
    <!--//
      function fbin (a,b,eps)
       { var fs=form1.func.value
         var x=a
         var fa=eval(fs)
         x=b
         var fb=eval(fs)
         var fc
         var c
         var r
         if (Math.abs(b-a) < eps)
```

```
          { r= (a+b)/2 }
        else
          { c=(a+b)/2;
            x=c; fc=eval(fs)
            if (fa*fc < 0)
             { r=fbin(a,c,eps) }
            else
             { r=fbin(c,b,eps) }
          }
        return r
       }
      function ftest(obj)
       { var fs=form1.func.value
         var x=obj.res.value
         var tes=eval(fs)
         obj.test.value=tes
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Нахождение корня уравнения вида <i>f</i>(<i>x</i>)=0
        методом деления отрезка пополам</H4>
    <FORM name="form1">
<pre>
Введите функцию:               <input type="text" size=30
name="func"><hr>
Введите левую границу отрезка:  <input type="text" size=10 name="l"><hr>
Введите правую границу отрезка: <input type="text" size=10 name="r"><hr>
Задайте требуемую точность:    <input type="text" size=10 name="eps"><hr>
<input type="button" value=Вычислить
onClick="form1.res.value=fbin(1*form1.l.value,1*form1.r.value,
         1*form1.eps.value)">
<hr>
Корень уравнения равен: <input type="text" size=30 name="res"><hr>
<input type="button" value="      Тест      " onClick="ftest(form1)"><hr>
Тест: <input type="text" size=30 name="test"><hr>
<input type="reset" value=Отменить>
```

```
</pre>
    </FORM>
  </BODY>
</HTML>
```

Для уравнения $x^4 + 2x^3 - 2 = 0$ все условия, при которых можно применять метод деления отрезка пополам, выполнены для интервала [0, 1]. Результат работы программы приведен на рис. 16.3.

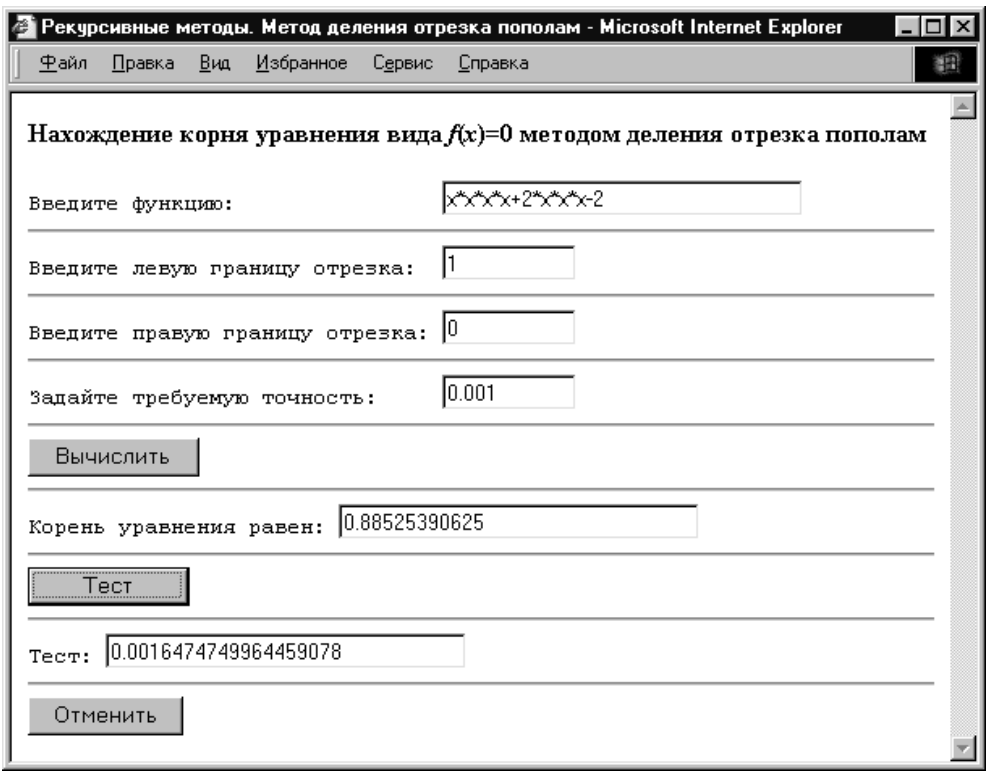


Рис. 16.3. Рекурсивная функция нахождения корня уравнения

Рассмотренные примеры показывают, что в ряде случаев можно получить короткую и выразительную программу, если строить рекурсивный алгоритм для решения задачи. Для продемонстрированных примеров, однако, существуют не очень сложные алгоритмы решения без применения рекурсии. Рассмотрим задачу, для которой легко предложить рекурсивный алгоритм решения, в то время как нерекурсивный алгоритм был бы сложным и искусственным.

Задача о Ханойских башнях

Рассмотрим решение задачи, которая получила название "Задача о Ханойских башнях". Существует древнеиндийская легенда, согласно которой в городе Бенаресе под куполом главного храма, в том месте, где находится центр Земли, на бронзовой площадке стоят три алмазных стержня. В день сотворения мира на один из этих стержней было надето 64 кольца. Бог поручил жрецам перенести кольца с одного стержня на другой, используя третий в качестве вспомогательного. Жрецы обязаны соблюдать условия:

1. Переносить за один раз только одно кольцо;
2. Кольцо можно переносить с одного стержня (x) на другой (y), только если оно имеет меньший диаметр, чем верхнее кольцо, находящееся на стрежне y, или стержень y свободен.

Согласно легенде, когда, соблюдая все условия, жрецы перенесут все 64 кольца, наступит конец света.

На рис. 16.4 изображена ситуация, когда требуется перенести 7 колец.

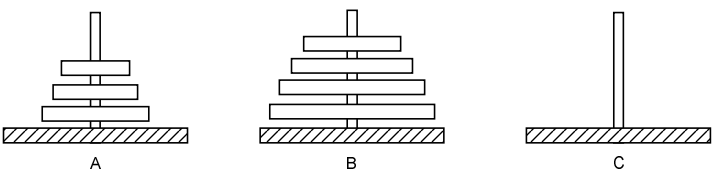


Рис. 16.4. Ситуация при переносе семи колец

Кольцо со стержня A можно перенести на стержень B или C, кольцо со стержня B можно перенести на стержень C, однако, нельзя перенести его на стержень A.

Задача состоит в том, чтобы определить последовательность минимальной длины переноса колец. Решением задачи будем считать последовательность допустимых переносов, каждый из которых имеет вид $x \rightarrow y$, где x и y одно из обозначений: A, B или C. Если кольцо всего одно, то задача решается за один перенос A $\rightarrow$ B. Для перемещения двух колец требуется выполнить три действия: A $\rightarrow$ C, A $\rightarrow$ B, C $\rightarrow$ B. Решение задачи для трех колец содержит семь действий, для четырех — 15.

Напишем рекурсивную функцию, которая находит решение для произвольного числа колец. Функция имеет четыре параметра, первый параметр — число переносимых колец, второй параметр — стрежень, на который первоначально нанизаны кольца. Третий параметр функции — стержень, на который требуется перенести кольца, и, наконец, четвертый параметр — стержень, который разрешено использовать в качестве вспомогательного. Последовательность переноса n колец изображена на рис. 16.5.

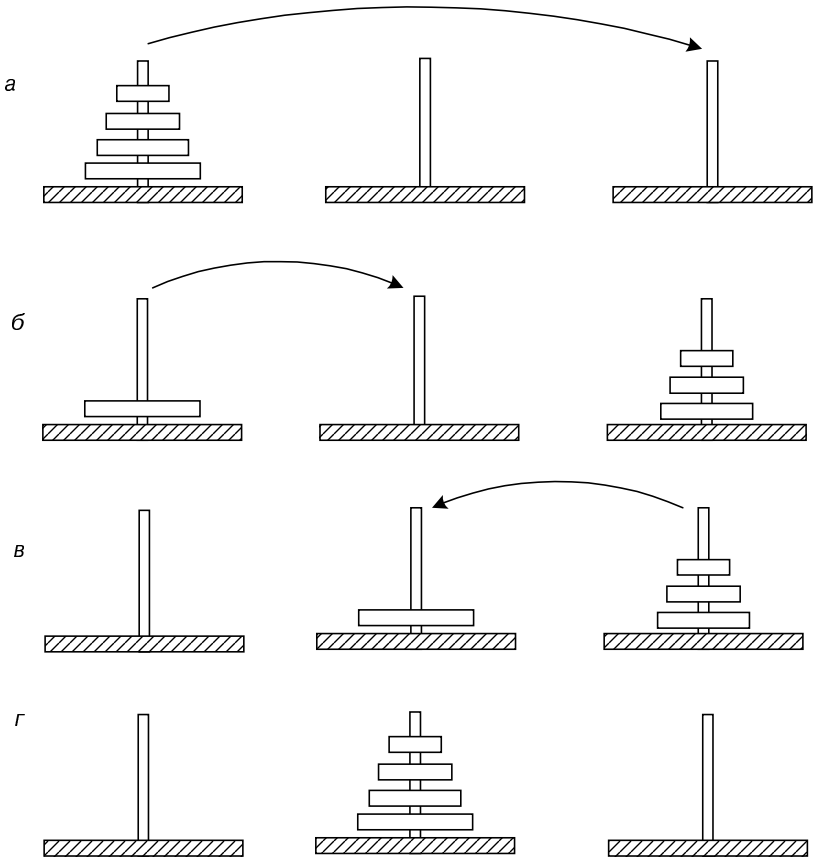


Рис. 16.5. Схема решения задачи о Ханойских башнях

Чтобы перенести n колец со стержня x на стержень y, используя стрежень z в качестве вспомогательного (рис. 16.5, *г*), можно поступить следующим образом:

- перенести $n-1$ кольцо со стержня x на z, используя стержень y в качестве вспомогательного стержня (рис. 16.5, *а*);
- перенести последнее кольцо со стержня x на стержень y (рис. 16.5, *б*);
- перенести $n-1$ кольцо со стержня z на y, используя стержень x в качестве вспомогательного стержня (рис. 16.5, *в*).

При переносе $n-1$ кольца можно воспользоваться тем же алгоритмом, т. к. на нижнее кольцо с самым большим диаметром можно просто не обращать внимания. Перенос одного кольца в программе выражается в том, что выводится соответствующий ход.

HTML-код, содержащий описанный сценарий, приведен в листинге 16.6.

Листинг 16.6. Задача о Ханойских башнях

```
<HTML>
  <HEAD>
    <TITLE>Задача о Ханойских башнях</TITLE>
    <script language="JavaScript">
    <!--
    function  hb(n,x,y,z)
     { if (n>0)
       { hb (n-1,x,z,y);
         document.writeln ("<P>", x,"  ->  ",y);
         hb (n-1,z,y,x);
       }
     }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Решение задачи о Ханойских башнях</H4>
    <FORM name="form1">
<pre>
Введите число колец:              <input type="text" size=5
name="st1"><hr>
Введите начальный стержень:       <input type="text" size=5
name="st2"><hr>
Введите конечный стержень:        <input type="text" size=5
name="st3"><hr>
Введите вспомогательный стержень: <input type="text" size=5
name="st4"><hr>
</pre>
      <input type="button" value=Выполнить
             onClick="hb(form1.st1.value,form1.st2.value,
                         form1.st3.value,form1.st4.value)">
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Оценим, сколько переносов будет сделано и сколько шагов выведено при работе программы. Обозначим T_n — число ходов, необходимых для переноса n колец. Очевидно, что $T_0 = 0$, $T_n = 2 \times T_{n-1} + 1$ при $n>0$, отсюда получаем:

- $T_n = 2^n - 1$
- $T_{64} = 2^{64} - 1 = 2 \times 10^{19}$

Если считать, что на перенос одного кольца потребуется одна секунда, то на перенос всех колец потребуется более 5 млрд веков.

Упражнения

1. Напишите рекурсивную функцию `summ(s,i,j)`, проверяющую, является ли симметричной часть строки `s`, начинающаяся `i`-м и заканчивающаяся `j`-м ее элементами.
2. Напишите рекурсивную функцию, "переворачивающую" заданное натуральное число.
3. Напишите рекурсивную функцию, которая по последовательности литер, представляющих двоичное число, определяет соответствующее десятичное число.
4. Напишите рекурсивную функцию, вычисляющую $n!!$.
5. Напишите рекурсивную функцию, определяющую количество единиц в двоичном представлении натурального числа.
6. Напишите рекурсивную функцию, которая по заданным натуральным числам m и n выводит все различные представления числа n в виде суммы m натуральных слагаемых. Представления, различающиеся лишь порядком слагаемых, считаются одинаковыми.
7. Напишите рекурсивную функцию, которая формирует в обратном порядке текст, задаваемый пользователем в текстовом поле формы.
8. Напишите рекурсивную функцию для вычисления биномиального коэффициента C_n^m.
9. Напишите рекурсивную функцию, проверяющую, является ли последовательность символов идентификатором. Напомним, что идентификатором будем считать последовательность букв и/или цифр, начинающуюся с буквы.
10. В последовательности символов могут встречаться только цифры и знаки + (плюс), при этом последовательность представляет собой формулу сложения однозначных чисел. Напишите рекурсивную функцию, определяющую значение формулы.
11. В последовательности символов могут встречаться только цифры и знаки + (плюс) или - (минус), при этом последовательность представляет собой формулу арифметической суммы однозначных чисел. Напишите рекурсивную функцию, определяющую значение формулы.

Глава 17

Метод исчерпывающего перебора

Рассмотрим следующую задачу. Пусть задаются значения переменных целого типа *a*, *b*, *c*, *d*. В формуле вида *a* ? *b* ? *c* требуется вместо знака вопроса поставить операцию (сложение, вычитание, умножение, остаток от деления) так, чтобы значение всей формулы равнялось значению переменной *d*. Данную задачу можно решать методом, который получил название *метода исчерпывающего перебора*. Будем перебирать возможные комбинации знаков (шестнадцать вариантов), вычислять значение формулы при каждой комбинации и отбирать формулы, удовлетворяющие условию.

Нахождение формул вида *a* ? *b* ? *c* = *d*

Напишем программу, которая для формулы вида *a* ? *b* ? *c* расставляет знаки операций таким образом, что значение формулы равно заданному значению *d*.

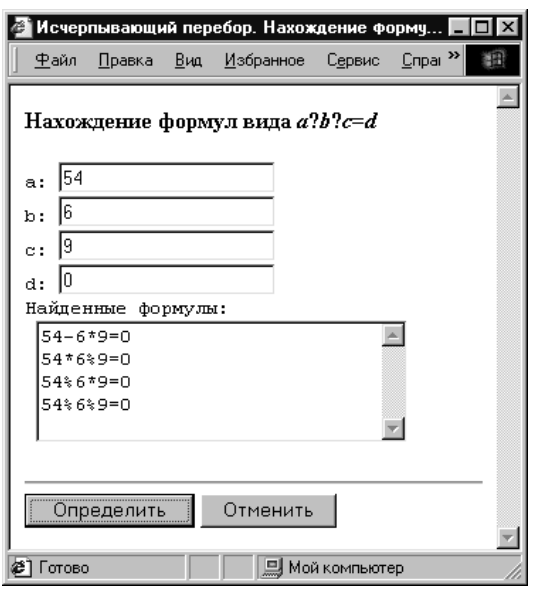


Рис. 17.1. Поиск комбинаций знаков

Для формирования формулы будем использовать переменную `s`. Сначала с помощью переменной `s1` запоминается подформула вида *a* ? *b*, затем подбираются знаки для второй операции. Если значение формулы совпадает с заданным значением, то формула запоминается в переменной `sres`. После анализа всех вариантов в текстовое поле формы выводятся либо найденные формулы, либо сообщение о том, что требуемые формулы не обнаружены. После того как формула сформирована, вычисляется ее значение с помощью стандартного метода `eval`. Результат работы функции на одном из тестовых примеров приведен на рис. 17.1.

Сценарий с описанной функцией имеет вид, представленный в листинге 17.1.

Листинг 17.1. Исчерпывающий перебор. Нахождение формул вида *a* ? *b* ? *c* = *d*

```
<HTML>
  <HEAD>
    <TITLE>Исчерпывающий перебор. Нахождение формул вида a?b?c=d</TITLE>
    <script LANGUAG="JavaScript">
    <!-- //
      function formula(obj)
       { var a=obj.data1.value
         var b=obj.data2.value
         var c=obj.data3.value
         var d=obj.data4.value
         var sres=""
         var s
         var s1
         var s2
         var r
         var p=false
         for (var  i=1; i<=4; i++)
          { s1= a
            switch (i)
             { case 1: s1+="+"; break
               case 2: s1+="-"; break
               case 3: s1+="*"; break
               case 4: s1+="%"; break
             }
            s1+=b
            for (var j=1; j<=4; j++)
```

```
            { s2=""
              switch (j)
               { case 1: s2+="+"; break
                 case 2: s2+="-"; break
                 case 3: s2+="*"; break
                 case 4: s2+="%"; break
               }
              s2+=c
              s=s1+s2
              r=eval(s)
              if (r==d)
               {sres +=s+ "="+d+"\r\n"; p=true}
            }
          }
         if (p)
          obj.result.value=sres
         else
          obj.result.value="Формулы, удовлетворяющие условию, не найдены"
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Нахождение формул вида <I>a</I>?<I>b</I>?<I>c</I>=<I>d</I></H4>
    <FORM name="form1">
<pre>
a: <input type="text" size=20 name="data1" >
b: <input type="text" size=20 name="data2" >
c: <input type="text" size=20 name="data3" >
d: <input type="text" size=20 name="data4" >
Найденные формулы:
 <textarea cols=30 rows=5 name="result" ></textarea>
</pre>
      <HR>
      <input type="button" value=Определить onClick="formula(form1)">
      <input type="reset" value=Отменить>
    </FORM>
  </BODY>
</HTML>
```

Нахождение формул вида *a* ? (*b* ? (*c* ? (*d* ? (*e* ? *f*)))) с заданным значением

На одной из Олимпиад по программированию была предложена следующая задача. Рассматривается формула вида 1 ? (2 ? (3 ? (4 ? (5 ? 6)))). Вместо знака вопроса поставьте одну из арифметических операций так, чтобы результат вычислений был равен 25. Сколько формул удовлетворяет этому условию? Сформулируем задачу в общем виде.

Необходимо написать функцию, которая для формулы вида *a* ? (*b* ? (*c* ? (*d* ? (*e* ? *f*)))) находит все комбинации знаков, при которых формула имеет заданное значение.

При решении задачи формулы будем хранить в одномерном массиве. Некоторые значения массива нам известны (заданные операнды и скобки), а некоторым элементам следует присвоить значения операций. При задании массива `s` все операции совпадали с операцией `+` (сложение). В формуле, представленной массивом, операции являются элементами с индексами 1, 4, 7, 10, 13. Для задания операций используются циклы. Когда очередная комбинация знаков готова, массив преобразуется в строку. При этом применяется метод `eval` для вычисления значения формулы, представленной строкой. Формулы, удовлетворяющие условию, запоминаются в строке `sres`. Описанная функция представлена в листинге 17.2.

Листинг 17.2. Нахождение формул вида *a* ? (*b* ? (*c* ? (*d* ? (*e* ? *f*)))) с заданным значением

```
<HTML>
  <HEAD>
    <TITLE>Нахождение формул вида a?(b?(c?(d?(e?f)))) </TITLE>
    <script language="JavaScript">
    <!-- //
      function formula (obj)
       { var a=Number(obj.data1.value)
         var b=Number(obj.data2.value)
         var c=Number(obj.data3.value)
         var d=Number(obj.data4.value)
         var e=Number(obj.data5.value)
         var f=Number(obj.data6.value)
         var g=Number(obj.data7.value)
         var s=new Array
(a,"+","(",b,"+","(",c,"+","(",d,"+","(",e,"+",f,")",")",")",")",")")
```

```
         var op=new Array ("+","-","*","%")
         var str=""
         var sres=""
         var str
         var p=false
         for (var  i=0; i<=3; i++)
          { s[1]=op[i];
            for (var j=0; j<=3; j++)
            { s[4]=op[j];
              for (var k=0; k<=3; k++)
               { s[7]=op[k];
                 for (var m=0; m<=3; m++)
                  { s[10]=op[m];
                    for (var n=0; n<=3; n++)
                     { s[13]=op[n]
                       str=""
                       for (var t=0; t <= s.length-1; t++)
                         { str+=s[t] };
                       r=eval(str)
                       if (r==g)
                        {sres +=str+ "="+g+"\r\n"; p=true}
                     }
                  }
               }
            }
          }
         if (p)
           obj.result.value=sres
         else
           obj.result.value="Формул, удовлетворяющих условию, не найдено.
"
       }
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=silver>
    <H4>Нахождение формул вида a?(b?(c?(d?(e?f))))</H4>
    <FORM name="form1">
<pre>
```

```
a: <INPUT type="text" size=20 name="data1" >
b: <INPUT type="text" size=20 name="data2" >
c: <INPUT type="text" size=20 name="data3" >
d: <INPUT type="text" size=20 name="data4" >
e: <INPUT type="text" size=20 name="data5" >
f: <INPUT type="text" size=20 name="data6" >
g: <INPUT type="text" size=20 name="data7" >
Найденные формулы:
 <textarea cols=30 rows=5 name="result" > </textarea>
</pre>
<HR>
 <INPUT type="button" value=Определить  onClick="formula(form1)">
  <INPUT type="reset" value=Отменить>
</FORM>
  </BODY>
</HTML>
```

Результат работы функции на одном из тестовых примеров приведен на рис. 17.2.

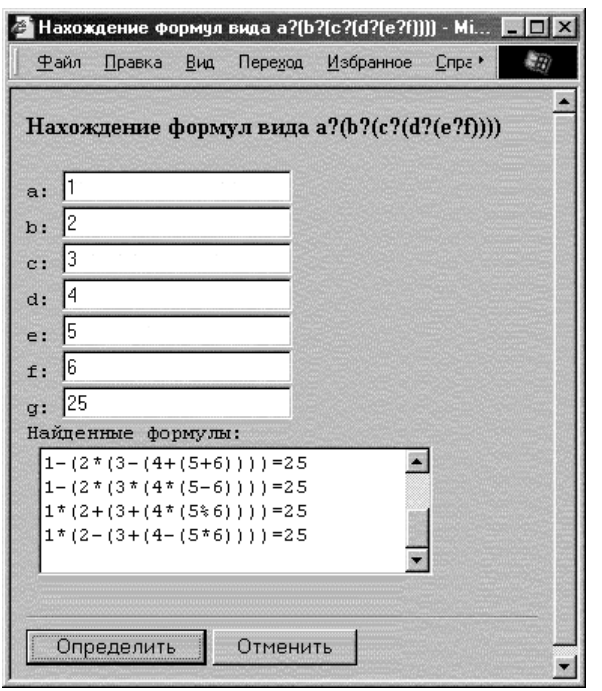


Рис. 17.2. Полный перебор

При решении задачи методом исчерпывающего перебора следует обратить внимание на два момента:

1. Требуется определить порядок, в котором следует рассматривать варианты.
2. Необходимо убедиться в том, что все варианты рассмотрены.

Более подробно на этих вопросах остановимся при решении следующей задачи.

Нахождение формул вида $a_1 ? a_2 ? a_3 ? ... ? a_n = b$

Пусть имеется n ($n \le 10$) целых значений a_1, a_2, …, a_n и целое число B. Требуется написать программу, определяющую все формулы, для которых верно

```
a1⊕a2⊕...⊕an = B
```

где ⊕ — обозначение операции + (сложение) или - (вычитание). Например, если значение n равно 4, все числа a_1 равны 1, значение B равно 2, то в результате выполнения программы должны быть выведены следующие формулы:

```
1-1+1+1 = 2
1+1-1+1 = 2
1+1+1-1 = 2
```

Если же значение B равно 4, то в качестве ответа должна быть формула

```
1+1+1+1 = 4
```

В случае, когда B равно 5, следует выдать информацию, что требуемые формулы не найдены.

Если формула содержит n значений, то в искомой формуле должно быть использовано $n-1$ знаков операций. Так как по условию задачи разрешено использовать только знаки + (плюс) или - (минус), то возможно 2^n-1 различных комбинаций знаков. Например, если исследуется формула, содержащая четыре переменных вида `a1⊕a2⊕a3⊕a4`, то необходимо рассмотреть и вычислить восемь формул, представленных в табл. 17.1.

Таблица 17.1. *Формулы и соответствующие им комбинации знаков*

№ варианта	Формула	Комбинация знаков
1	$a_1 - a_2 - a_3 - a_4$	– – –
2	$a_1 - a_2 - a_3 + a_4$	– – +
3	$a_1 - a_2 + a_3 - a_4$	– + –
4	$a_1 - a_2 + a_3 + a_4$	– + +

Таблица 17.1 *(окончание)*

№ варианта	Формула	Комбинация знаков
5	$a_1 + a_2 - a_3 - a_4$	+ – –
6	$a_1 + a_2 - a_3 + a_4$	+ – +
7	$a_1 + a_2 + a_3 - a_4$	+ + –
8	$a_1 + a_2 + a_3 + a_4$	+ + +

Если знаку - (минус) сопоставить ноль, а знаку + (плюс) — единицу, то комбинацию знаков можно трактовать как двоичную запись числа α, где $0 \le \alpha \le 2^{n-1} - 1$. Разбор всех возможных комбинаций знаков соответствует анализу двоичных представлений всех чисел α, где $0 \le \alpha \le 2^{n-1} - 1$. Если n равно 4, то выписанные ранее комбинации знаков отвечают двоичным представлениям чисел от 0 до 7. Комбинации знаков представлены в табл. 17.2.

Таблица 17.2. *Соответствие между комбинацией знаков и двоичным числом*

№ варианта	Число в двоичной системе счисления	Комбинация знаков
1	000	– – –
2	001	– – +
3	010	– + –
4	011	– + +
5	100	+ – –
6	101	+ – +
7	110	+ + –
8	111	+ + +

Теперь надо уточнить порядок рассмотрения вариантов. Самый простой подход состоит в том, чтобы переход от одной комбинации к другой означал переход от двоичной записи числа α к двоичной записи числа $\alpha + 1$, т. е. прибавление к текущему значению двоичного числа единицы. Если в качестве начальной комбинации знаков взять комбинацию из всех минусов (комбинация соответствует числу ноль), то после выполнения $2^{n-1} - 1$ числа сложений будет получена комбинация знаков, состоящая только из знаков плюс, и соответствующая двоичной записи числа $2^{n-1} - 1$. Все возможные комбинации знаков будут исчерпаны. Таким образом, процедура моделирует увеличение двоичного числа на единицу и состоит в просмотре массива `z` с конца и поиске первого знака минус. При этом все встретившиеся знаки

плюс заменяются знаком минус. Первый с конца знак минус также заменяется знаком плюс и просмотр массива знаков завершается. При решении задачи нам потребуется формировать массив из $n-1$ знаков. Переход к следующей комбинации знаков осуществляется при вызове функции `NewSign`:

```
function NewSign()
 { var i=n-2
   while (i>=0)
    { if (z[i]=="+")
       { z[i] = "-"; i-- }
      else
       { z[i] = "+"; break }
    }
 }
```

При выполнении функции `formula` перебираются все комбинации знаков. Если значение формулы совпадает с требуемым значением, то такая формула запоминается. Кроме того, подсчитывается количество формул, удовлетворяющих условию. Массив, содержащий значения операндов формулы, хранится в отдельном файле. Пример работы сценария для некоторого тестового набора приведен на рис. 17.3.

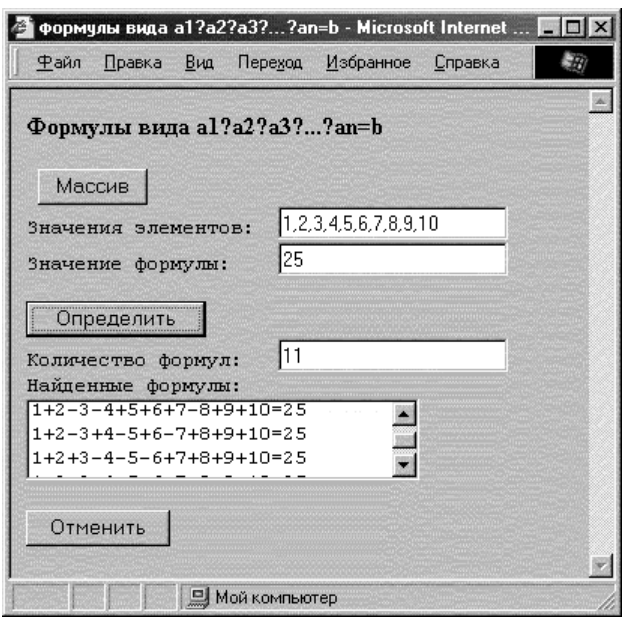


Рис. 17.3. Формула из n значений без скобок

Полностью сценарий с описанными функциями представлен в листинге 17.3.

Листинг 17.3. Нахождение формул вида $a_1 ? a_2 ? a_3 ? \ldots ? a_n = b$

```
<HTML>
  <HEAD>
    <TITLE>формулы вида a1?a2?a3?...?an=b</TITLE>
    <script src="arrform.js">
    </script>
    <script language="JavaScript">
    <!-- //
      var sres=""
      var k               // Число комбинаций знаков
      var b               // Значение формулы
      var n= v.length     // Реальное число переменных
      var varsh=0         // Количество найденных решений
      // Функция Vform вычисляет значение формулы
      // при заданной комбинации знаков
      function Vform()
       { var  s = Number (v[0])
         for (var i = 0; i <= n-2; i++)
          switch (z[i])
           { case "+": s += Number( v[i+1]);  break
             case "-": s -=  Number( v[i+1]); break
           }
         return s
       }
      // Функция WriteForm выводит формулу
      function WriteForm()
       { var str= v[0]
         for ( var i = 0 ; i <= n-2; i++)
          switch ( z[i] )
           { case "+": str +="+"+ v[i+1];  break
             case "-": str +="-"+  v[i+1]; break
            }
         str += "="+b+"\r\n"
         return str
       }
```

```
// Функция NewSign формирует следующую комбинацию знаков
function NewSign()
 { var i=n-2
   while (i>=0)
    { if (z[i]=="+")
       { z[i] = "-";  i-- }
      else
       { z[i] = "+"; break }
    }
 }
// Функция InitSign формирует начальную комбинацию знаков
function InitSign()
 { for (var i =0; i<= n-2; i++)
    { z[i] = "-" }
 }
// Функция formula просматривает все варианты и отбирает
// удовлетворяющие условию
function formula (obj)
 { b= obj.valform.value
   if (b=="")
     alert("Вы не ввели значение формулы")
   else
    { k = Math.pow(2,n-1)
      sres=""
      varsh=0
      InitSign()
      for (var t = 1; t<= k; t++)
       { if (Vform() ==Number(b))
          { sres += WriteForm(); varsh++ }
         NewSign()
       }
      if (varsh==0)
        sres= "таких формул нет"
      obj.result.value=sres
      obj.data1.value=varsh
    }
 }
function writearr(obj)
 {obj.vararray.value=v}
```

```
    //-->
    </script>
  </HEAD>
  <BODY bgcolor=silver>
    <H4>Формулы вида  a1?a2?a3?...?an=b</H4>
    <FORM name="form1">
<pre>
 <input type="button" value=Массив  onClick="writearr(form1)">
Значения элементов:  <input type="text" size=20 name="vararray">
Значение формулы:    <input type="text" size=20 name="valform"><BR>
<input type="button" value=Определить  onClick="formula(form1)">
Количество формул:   <input type="text" size=20 name="data1">
Найденные формулы:
<textarea cols=30 rows=3 name="result" > </textarea>
</pre>
 <input type="reset" value=Отменить onClick= 'sres=""; varsh=0 '>
    </FORM>
  </BODY>
</HTML>
```

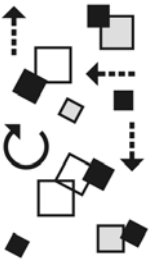
Упражнения

1. Напишите функцию, которая для формулы вида ((*a* ? *b*) ? *c*) ? ((*d* ? *e*) ? *f*), где вместо знака вопроса может стоять любая из операций сложения, вычитания, умножения, находит все комбинации знаков, при которых формула имеет заданное значение.
2. Напишите сценарий, который размещает цифры 1, 2, 3, 4, 5, 6, 7, 8, 9 так, чтобы имело место: `-----/-----=n`, где `n` равно одному из чисел 2, 3, 4, 5, 6, 7, 8, 9.
3. Пусть задана формула вида $A_1 \oplus A_2 \oplus ... \oplus A_n$, где A_i может принимать целое значение, знак $\oplus$ соответствует операции сложения, вычитания или умножения. Напишите сценарий, который определяет, существует ли набор знаков операций, при которых формула принимает заданное значение.
4. Пусть задана формула вида $A_1 \oplus A_2 \oplus ... \oplus A_n$, где A_i может принимать целое значение, знак $\oplus$ соответствует операции сложения, вычитания или умножения. Напишите сценарий, который определяет все наборы знаков операций, при которых формула принимает заданное значение.
5. Пусть задана формула вида $A_1 \oplus A_2 \oplus ... \oplus A_n$, где A_i принимает значения либо истина, либо ложь, знак $\oplus$ соответствует операции дизъюнкции или

конъюнкции. Напишите сценарий, который определяет, существует ли набор знаков операций, при которых формула принимает заданное значение.

6. Пусть задана формула вида $A_1 \oplus A_2 \oplus \ldots \oplus A_n$, где A_i может принимать значения либо истина, либо ложь, знак $\oplus$ соответствует операции дизъюнкции или конъюнкции. Напишите сценарий, который по указанному набору операций присваивает переменным A_i такие значения, чтобы значение всей формулы совпадало с заданным.
7. Дано уравнение вида $F(x) = 0$, где $F(x)$ строится из целых чисел, арифметических операций (+, –, *, /), и переменной x, которая может входить не более одного раза. Напишите программу, которая решает заданное уравнение и печатает его корни в виде несократимой дроби.

Глава 18



Метод рекурсивного спуска

Про формулы, в которых знак операции стоит между операндами, говорят, что они записаны в инфиксной нотации. Именно такие формулы мы и рассматривали и обрабатывали ранее. Рассмотрим рекурсивные алгоритмы обработки формул, представленных как в инфиксной, так и в других формах.

Представление формулы в прямой польской записи

Во многих случаях можно использовать формулы в бесскобочной записи. В префиксной форме каждый знак операции ставится перед своими операндами. Такую запись иногда называют прямой польской записью. Формула $a+b*c$ в префиксной нотации запишется так $+a*bc$, а формула $(a+b)*c-(d+e)/k$ следующим образом: $-*+abc/+dek$. Итак, формула в инфиксной форме $A \oplus B$, где $\oplus$ — бинарная операция, A и B — операнды (которые в свою очередь могут быть формулами). В префиксной форме формула выглядит так $\oplus AB$. В прямой польской записи скобки не требуются, т. к. для каждой операции известны операнды.

Напишем программу, вычисляющую значение формулы, содержащей целые константы, операции `+`, `-`, `*`, `%`, и представленной в прямой польской записи.

Будем считать, что в формуле встречаются целые константы из диапазона [0; 9]. Формула

((3+7)*(5−2)+7)*2+6.

в префиксной форме запишется так:

+*+*+37−52726.

Результат выполнения сценария приведен на рис. 18.1.

Введенная пользователем формула в префиксной нотации хранится в строке `st`. Формула просматривается слева направо. Функция `cursym` обеспечивает выбор очередного символа для анализа. Допустимыми символами являются знаки операций и цифры, выступающие в роли операндов. Является ли оче-

редной символ знаком операции, проверяется функцией `sig`. Сообщение об ошибке формирует функция `er`.

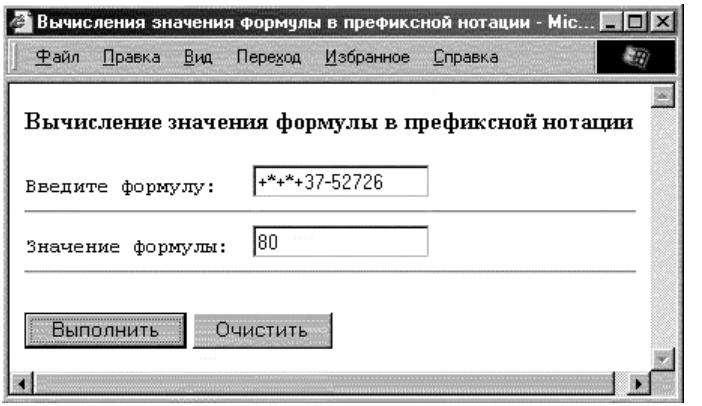


Рис. 18.1. Значение формулы в префиксной нотации

Значение формулы считается с помощью рекурсивной функции `vform`. Поясним ее работу. Формула в префиксной нотации имеет вид `ΘAB`, следовательно, первый символ обязан быть знаком операции. Запоминаем его в переменной `h`. Подформула `A` является формулой в префиксной нотации, поэтому ее значение можно вычислить с помощью той же самой функции `vform`. Следует выбрать первый символ формулы, вычислить ее значение и присвоить переменной `l`. Если формула не содержит ошибок, то следует вычислить значение подформулы `B`. Выбирается первый символ подформулы `B`, вычисляется значение и присваивается переменной `r`. После этого все готово для того, чтобы завершить вычисление значения всей формулы `ΘAB`. В зависимости от знака операции выполняются соответствующие действия. Приведем программу, содержащую сценарий решения задачи (листинг 18.1).

Листинг 18.1. Вычисления значения формулы в префиксной нотации

```
<HTML>
  <HEAD>
    <TITLE>Вычисления значения формулы в префиксной нотации</TITLE>
    <script language="JavaScript">
    <!-- //
      // Глобальные переменные
      var  st
      var i
      var c
      var r
```

```
// Функция выбора для анализа очередного символа
function cursym()
 { i=i+1
   var h
   h =  st.charAt(i);
   return h
 }
//Функция, определяющая, является ли символ знаком операции
function sig(z)
 { var p = (z=="+") || (z=="-") || (z=="*") || (z=="/"); return p}
// Функция обработки ошибки
function  er (s )
 { var y = "ошибка в формуле: "+s+"<br>"
   document.write (y)
 }
// Значение формулы в префиксной нотации
function vform()
 { var g;   var h;       var l;      var r;
   if (c >= "0" && c <= "9")
    { g= c*1; return g }
   else
    { if ( sig (c) == true)
      { h=c
        c = cursym();  l = vform()
        c = cursym();  r = vform()
        switch (h)
         { case "+": g=l +r; break;
           case "-": g= l-r; break;
           case "*": g= l*r; break;
           case "/" :  if (r != 0 )  { g= l/r; break }
                       else er("Деление на ноль")
         }
        return g
       }
      else er("неверный символ")
    }
 }
 // Функция, обеспечивающая начальные установки
 // и вызов функции vform
```

```
      function  main (obj)
       { st = obj.f1.value
         i=-1
         c = cursym ()
         r = vform()
         obj.fres.value = r
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Вычисление значения формулы в префиксной нотации</H4>
    <FORM name="form1">
<pre>
Введите формулу:   <input type="text" size=15 name="f1"><HR>
Значение формулы:  <input type="text" size=15 name="fres"><HR>
</pre>
      <input type="button" value=Выполнить onClick="main(form1)">
      <input type="reset" value=Очистить >
    </FORM>
  </BODY>
</HTML>
```

Представление формулы в обратной польской записи

При записи формулы в постфиксной нотации (*обратной польской записи*) знак операции ставится непосредственно после операндов. Например, формула $a+b*c$ в постфиксной нотации: $abc*+$, формула $(a+b)*c-(d+e)/k$ в обратной польской записи выглядит так: ab+c*de+k/−. Для формулы в инфиксной нотации вида $A \oplus B$ формула в постфиксной нотации имеет вид $AB \oplus$, где A и B — формулы в постфиксной нотации.

Напишем программу, которая по произвольной формуле в инфиксной нотации строит формулу в постфиксной нотации.

Формула состоит из переменных (представленных однобуквенными идентификаторами), знаков операций и круглых скобок. Допустимыми знаками операции являются знаки `+`, `-`, `*` и `/`. При просмотре формулы слева направо число открывающих скобок должно быть более или равно числу закрывающих. Во всей формуле число открывающих и закрывающих скобок должно быть одинаково. Формулу при решении задачи просмотрим слева

направо один раз. Приведем метод решения, который получил название *метода рекурсивного спуска.*

Для решения задачи удобно ввести понятия, характеризующие отдельные части формулы и формулу целиком. Будем называть *множителем* либо переменную, либо формулу, заключенную в скобки. Будем называть *термом* — либо один множитель, либо последовательность множителей, соединенных знаками `*` или `/`. И, наконец, назовем *формулой* — один терм, либо последовательность термов, соединенных знаками `+` или `-`.

Сначала опишем функцию (назовем ее `postform`), которая по формуле формирует обратную польскую запись. Формула состоит либо из одного терма, либо из последовательностей термов, соединенных знаком `+` или `-`. Если формула имеет вид $T_1 + T_2 - T_3$, то записанная в постфиксной нотации, она будет выглядеть так: $T'_1 T'_2 + T'_3$, где T'_i — постфиксная нотация формулы T_i ($i = 1, 2, 3$).

При написании функции `postform` будем использовать функцию `postterm`, которая "умеет" строить постфиксную нотацию для подформулы, определенной как терм. Пусть функция `wrsym(h)` помещает символ в строку результат. Как и ранее будем считать, что процедура `cursym` выбирает для анализа очередной символ.

Процедуру `postform`, которая строит по формуле в инфиксной нотации формулу в постфиксной нотации, можно записать следующим образом:

```
function  postform()
 { postterm()
   while ((c == "+") || (c == "-"))
    { var h=c
      c=cursym()
      postterm()
      wrsym(h)
    }
 }
```

Теперь следует описать функцию `postterm`. Вспомним, что терм состоит либо из одного множителя, либо из последовательности множителей, соединенных знаками `*`, `/`.

Если терм имеет вид $M_1 \times M_2 \,/\, M_3 \times M_4$, то записанный в постфиксной нотации он будет выглядеть так: $M'_1 M'_2 \times M'_3 \,/\, M'_4 \times$, где M'_i — постфиксная нотация формулы M_i ($i = 1, 2, 3, 4$). При написании функции `postterm` можно использовать функцию `postmn`, которая "умеет" строить постфиксную нотацию для подформулы, определенной как множитель.

```
function  postterm()
 { postmn()
   while ((c == "*") || (c == "/"))
```

```
    { var h=c
      c=cursym()
      postmn()
      wrsym(h)
    }
 }
```

Если множитель состоит лишь из переменной, то она уже находится в постфиксной нотации. Если множитель представляет собой формулу в скобках, т. е. имеет вид (*F*), то постфиксная нотация такой формулы совпадает с постфиксной нотацией формулы *F*. Для того чтобы построить постфиксную нотацию формулы *F*, следует обратиться к описанной ранее функции `postform`:

```
function  postmn()
 { if((c >= "a") && (c <= "z"))
    { wrsym(c)
      c=cursym()
    }
   else
    { if (c == "(" )
       { c = cursym()
         postform()
         if (c != ")") er=true
         else
          { c = cursym() }
         }
       else er=true
    }
 }
```

Проследите, как будет осуществляться вызов описанных функций при преобразовании формулы, представленной на рис. 18.2.

Заметим, что функция `postform` содержит вызов функции `postterm`, которая обращается к функции `postmn`, последняя, в свою очередь, содержит вызов функции `postform`. Такую рекурсию иногда называют *косвенной*.

После того как пользователь ввел формулу и нажал на кнопку **Выполнить**, происходит вызов функции `main`, в которой выполняются необходимые начальные действия, такие как выбор первого символа для анализа, присвоение значения `false` переменной `er`, чтобы отследить ситуацию, если в формуле содержится ошибка. В функции `main` осуществляется вызов функции `postform()`, и анализ ситуации после завершения работы этой функции.

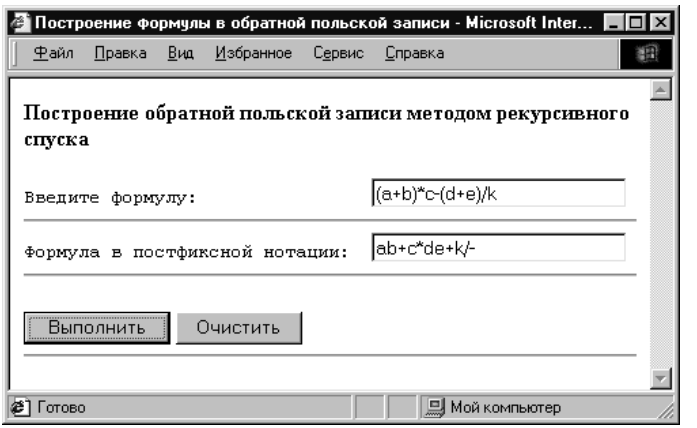


Рис. 18.2. Построение постфиксной нотации

HTML-код приведен в листинге 18.2.

Листинг 18.2. Построение формулы в обратной польской записи

```
<HTML>
  <HEAD>
    <TITLE>Построение формулы в обратной польской записи</TITLE>
    <script language="JavaScript">
    <!--
      // Глобальные переменные
      var s    // Формула в обратной польской записи
      var  st  // Исходная формула в инфиксной нотации
      var i    // Индекс очередного анализируемого символа
      var er   // Ошибка
      var  c   // Очередной символ формулы
      // Функция, осуществляющая начальные установки и
      // вызов основной функции, осуществляющей построение
      // обратной польской записи для формулы
      function main(obj)
       { s=""
         st=obj.f1.value
         i=-1
         er=false
         c=cursym()
         postform()
         if (er)
```

```
     alert ("Ошибка при задании формулы","<br>")
   else
    { form1.fres.value=s }
 }
// Выбор очередного символа
function cursym()
 { i=i+1
   return  st.charAt(i);
 }
// Формирование строки, представляющей формулу
// в обратной польской записи
function wrsym(h)
 { s+=h }
// Обратная польская запись для формулы
function postform()
 { postterm()
   while ((c == "+") || (c == "-"))
    { var h=c
      c=cursym()
      postterm()
      wrsym(h)
    }
  }
// Обратная польская запись для терма
function postterm()
 { postmn()
   while ((c == "*") || (c == "/"))
    { var h=c
      c=cursym()
      postmn()
      wrsym(h)
    }
 }
// Обратная польская запись для множителя
function postmn()
 { if ((c >= "a") && ( c <= "z"))
    { wrsym (c)
      c = cursym ();
    }
```

```
        else
         { if (c == "(" )
            { c = cursym()
              postform()
              if (c != ")")  er= true
              else
               { c = cursym() }
             }
           else  er = true
         }
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Построение обратной польской записи методом
        рекурсивного спуска</H4>
    <FORM name="form1">
<pre>
Введите формулу:                <input type="text" size=25 name="f1"><HR>
Формула в постфиксной нотации:  <input type="text" size=25
name="fres"><HR>
</pre>
      <input type="button" value=Выполнить onClick=" main (form1)">
      <input type="reset" value=Очистить><HR>
    </FORM>
  </BODY>
</HTML>
```

Если формула содержит ошибку (либо два операнда располагаются друг за другом, либо подряд следуют два знака, либо встретился недопустимый символ и др.), то значение переменной `er` станет равным `true`. После завершения работы функции `postform()` проверяется значение переменной `er`. Результат работы сценария в случае, когда формула содержит ошибку, представлен на рис. 18.3.

Может возникнуть ситуация, когда проанализирована не вся формула, поэтому после работы функции `postform()` надо убедиться, что формула просмотрена целиком. Изменить сценарий, чтобы учесть такой случай, читателю предлагается в качестве упражнения.

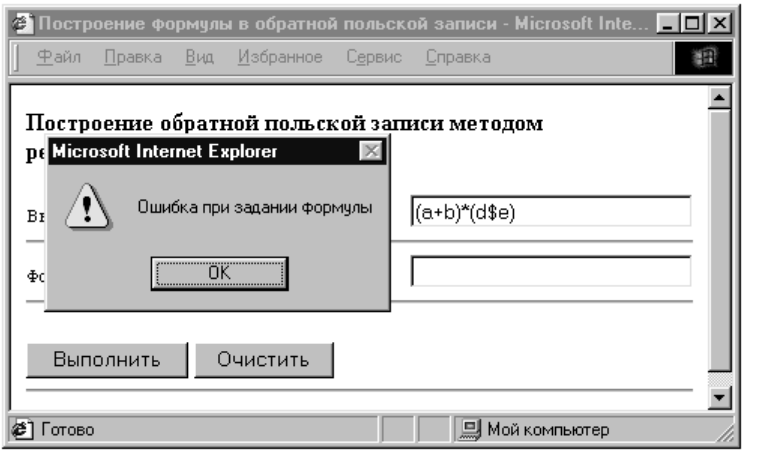


Рис. 18.3. Обработка ошибки

Вычисление значения формулы в прямой польской записи

Напишем сценарий, вычисляющий значение формулы. Формула состоит из целочисленных констант (от 0 до 9), знаков операций и круглых скобок. Допустимыми операциями являются сложение, вычитание, умножение и деление. Формулу при решении задачи требуется просмотреть слева направо один раз.

Для решения задачи удобно использовать понятия формулы, терма и множителя, введенные при решении предыдущей задачи.

Как и ранее, будем считать, что функция `cursym()` выдает для анализа текущий символ. Опишем функцию `valform()`, которая анализирует формулу, и, если формула построена правильно, вычисляет ее значение. По данному определению формула всегда начинается с терма. Будем считать, что функция `valterm()` вычисляет значение для части формулы, которую мы определили как терм. После вызова функции `valterm()` будет вычислено значение первого терма. Если далее в формуле следует знак `+` или знак `-`, то за знаком в формуле должен опять следовать терм. Вычислим значение и этого терма, воспользовавшись процедурой `valterm()`. Далее можно определить значение подформулы с двумя термами. Процесс продолжается до тех пор, пока после анализа очередного терма текущий символ не станет отличным от знака `+` или знака `-`.

Полностью функцию `valform` можно описать так:

```
function valform()
 { var t1 = valterm()
```

```
   var t2
   while ((c == "+") || (c == "-"))
    { var h = c
      c =cursym()
      t2 = valterm()
      if (h=="+") t1 = t1 + t2
      else t1= t1 – t2
    }
   return Number(t1)
 }
```

Напомним, что при вычислении значения формулы мы воспользовались функцией вычисления значения терма. Опишем ее. Мы предполагали, что терм состоит либо из одного множителя, либо из множителей, соединенных знаками умножения и деления. Поступим так, как уже поступали ранее. Будем считать, что функция `valmn()` вычисляет значение формулы, определенной как множитель. Для того чтобы вычислить значение терма, следует сначала определить значение множителя. Анализируя знак операции, следующий за первым термом, мы либо прекращаем анализ формулы, либо вычисляем значение очередного множителя и т. д. Функция `valterm()` очень похожа на функцию `valform()`. Следует обратить внимание на значение делителя при выполнении операции деления.

```
function valterm()
 { var t1 = valmn()
   var t2
   while (c == "*")
    { var h = c
      c = cursym()
      t2 = valmn()
      t1 = t1 * t2
    }
   return Number(t1)
 }
```

Теперь необходимо описать функцию `valmn()`. Множителем может быть константа (ее значение нам известно) или формула в скобках (то есть множитель может иметь вид (F)). Если же очередной символ не является ни числом, ни открывающей скобкой, то формула содержит ошибку.

Рассмотрим случай, когда множителем является формула в скобках. Значение формулы (F) совпадает со значением формулы F, а значение формулы мы уже умеем вычислять с помощью функции `valform()`, которой мы и воспользуемся. После вычисления значения формулы F очередным симво-

лом должна быть закрывающая скобка. Если это не так, то нарушен баланс скобок и формула содержит ошибку.

Будем, как и ранее, использовать логическую переменную `er`, значение которой изменится на `true` в случае определения ошибки в формуле. Приведем описание функции `valmn()`:

```
function valmn()
 { if ((c >= "1") && ( c <= "9"))
    { var h=Number(c)
      c = cursym();
      return h
    }
   else
    { if (c == "(")
       { c = cursym()
         var t = valform()
         if (c != ")")       er= true
         else
          { c = cursym ();  return Number(t) }
       }
      else  er = true
    }
 }
```

Заметим, что функция `valform` содержит вызов функции `valterm`, которая содержит вызов функции `valmn`, последняя, в свою очередь, обращается к функции `valform`. Здесь, как и в предыдущем случае, имеет место косвенная рекурсия.

Напомним, что рассмотренная нами ранее функция `eval(s)` рассматривает строку `s` как выражение, и вычисляет ее значение.

В программе для контроля работы описанных функций используется функция `valtest`, которая обращается к функции `eval` и записывает в соответствующее поле формы вычисленное с ее помощью значение (рис. 18.4).

Функция `eval` обладает бóльшими возможностями, чем приведенная программа. В этом смысле программа носит демонстрационный характер. Но если в формуле присутствуют знаки операций, не определенные в языке JavaScript, то воспользоваться методом `eval` не удается, требуется писать свою программу определения значения формулы. Предложенный метод рекурсивного спуска оказывается полезным при решении многих задач.

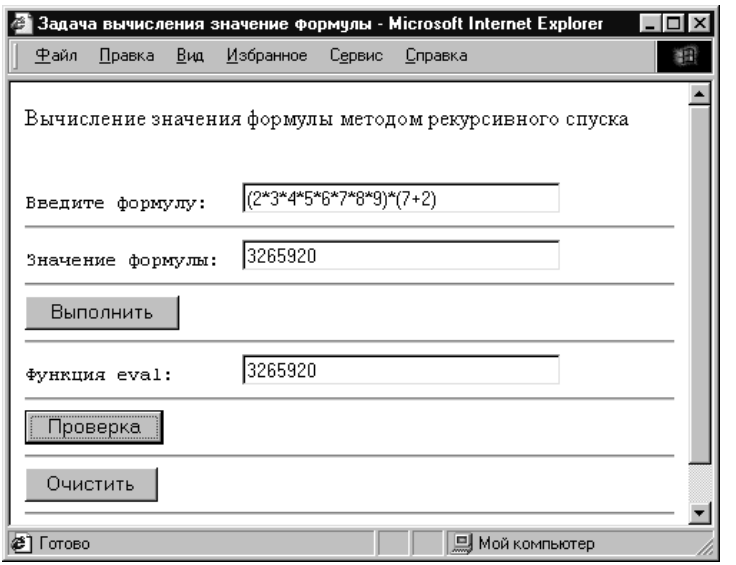


Рис. 18.4. Значение формулы в инфиксной нотации

Приведем полностью программу, решающую задачу (листинг 18.3).

Листинг 18.3. Вычисление значения формулы в инфиксной нотации

```
<HTML>
  <HEAD>
    <TITLE>Задача вычисления значения формулы</TITLE>
    <script language="JavaScript">
    <!-- //
      var  st     // Исходная формула
      var  i      // Индекс текущего символа
      var  er     // Фиксация ошибки
      var  c      // Очередной символ формулы
      // Выбор очередного символа
      function  cursym()
       { i=i+1; return st.charAt(i)}
      // Значение формулы
      function  valform()
       { var t1 = valterm()
         var t2
         while ((c == "+") || (c == "-"))
          { var h = c
```

```
      c =cursym()
      t2 = valterm()
      if (h=="+") t1 = t1 + t2
      else t1= t1 – t2
    }
   return Number(t1)
 }
// Значение терма
function  valterm()
 { var t1 = valmn()
   var t2
   while (c == "*")
    { var h =c
      c = cursym()
      t2 = valmn()
      t1 = t1 * t2
    }
   return Number(t1)
 }
// Значение множителя
function  valmn()
 { if ((c >= "1") && (c <= "9"))
    { var h= Number(c)
      c = cursym();
      return Number(h)
    }
   else
    { if (c == "(")
      { c = cursym()
        var t = valform()
        if (c != ")")           er= true
        else {  c = cursym ();  return Number(t)}
      }
      else  er = true
    }
 }
// Установка начальных значений и вызов функции vform
function  main(obj)
```

```
       { st = obj.f1.value
         i=-1
         er = false
         c = cursym()
         r = valform()
         if (er || (i != st.length))
           alert ("Ошибка при задании формулы","<br>")
         else  obj.fres.value = r
       }
      // Проверка работы описанных функций с помощью функции eval
      function  testval (obj)
       { st = obj.f1.value
         r= eval(st)
         obj.test.value = r
         if (er)
          alert ("Ошибка при задании формулы","<br>")
         else
          { obj.test.value = r }
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <P>Вычисление значения формулы методом рекурсивного спуска</P>
<pre>
<FORM name="form1">
Введите формулу:   <input type="text" size=30 name="f1"><HR>
Значение формулы:  <input type="text" size=30 name="fres"><HR>
<input type="button" value=Выполнить onClick="main(form1)"><HR>
Функция eval:      <input type="text" size=30 name="test"><HR>
<input type="button" value=Проверка onClick="testval(form1)"><HR>
<input type="reset" value=Очистить ><HR>
</FORM>
</pre>
  </BODY>
</HTML>
```

Расчет сопротивления параллельно-последовательной схемы

Применим метод рекурсивного спуска для решения следующей задачи. К наиболее простым и часто встречающимся типам соединения проводников относятся *последовательное* и *параллельное соединения*. При последовательном соединении электрическая цепь не имеет разветвлений, все проводники включают в цепь поочередно друг за другом. На рис. 18.5 показано последовательное соединение двух проводников, имеющих сопротивление R_1 и R_2. Полное сопротивление R при последовательном соединении определяется следующим образом:

$$R = R_1 + R_2.$$

Аналогичную формулу можно вывести для любого числа последовательно соединенных проводников.

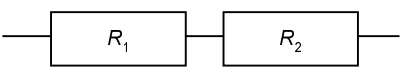


Рис. 18.5. Последовательное соединение проводников

На рис. 18.6 показано параллельное соединение двух проводников с сопротивлениями R_1 и R_2. Величина, обратная полному сопротивлению участка, равна сумме величин, обратных сопротивлению отдельных проводников:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2}.$$

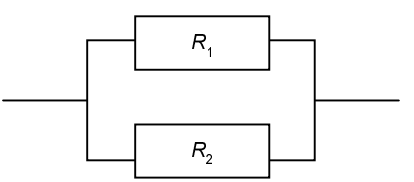


Рис. 18.6. Параллельное соединение проводников

Отсюда следует, что

$$R = \frac{R_1 \times R_2}{R_1 + R_2}.$$

Аналогичную формулу можно вывести для любого числа параллельно соединенных проводников.

Предположим, что задана электрическая схема из резисторов (сопротивлений). Схема является параллельно-последовательной (далее, ПП-

схема), т. е. содержит только последовательные и параллельные соединения регистров.

Формально понятие ПП-схемы можно представить следующим образом:

- Один резистор является ПП-схемой. Сопротивление схемы равно сопротивлению единственного входящего в нее резистора.
- Последовательно соединенные ПП-схемы являются ПП-схемой. Сопротивление схемы равно сумме сопротивлений последовательно соединенных схем.
- Параллельное соединение ПП-схем является ПП-схемой. Сопротивление схемы вычисляется по формуле

$$R = \frac{1}{\frac{1}{R_1} + ... + \frac{1}{R_n}},$$

где R_1, R_2, ..., R_n — сопротивления параллельно соединенных схем. На рис. 18.7 иллюстрируется понятие ПП-схемы.

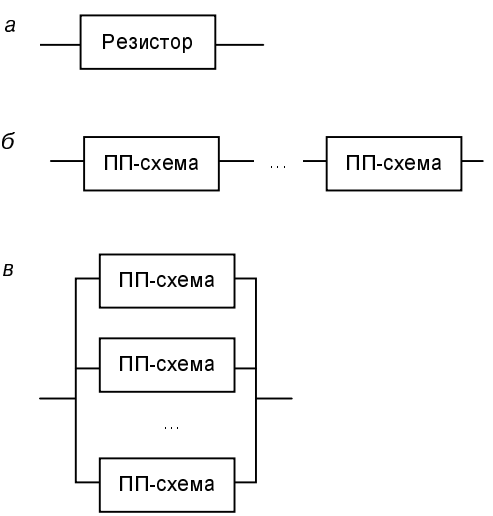


Рис. 18.7. Вычисление сопротивления ПП-схемы

Напишем программу, которая определяет полное сопротивление параллельно-последовательной схемы. В последовательных и параллельных соединениях участвуют, по крайней мере, две схемы. Величины сопротивлений задаются целыми числами из диапазона [1; 9].

Изображение электрической схемы будем записывать по определенным правилам в виде строки символов. Если схема состоит из одного проводника,

то задается число — величина сопротивления. Последовательное соединение схем будем задавать строкой вида `(A1+A2+...+An)`, где `Ai` — либо один проводник, либо электрическая схема. Параллельное соединение будем задавать строкой вида `(A1*A2*...*An)`, где `Ai` — либо один проводник, либо электрическая схема.

Такую задачу мы можем решить методом рекурсивного спуска. Предположим, что мы написали функцию (назовем ее `VF`), которая обрабатывает последовательное соединение и вычисляет его сопротивление. Функция `VF` использует функцию `VT`, которая обрабатывает параллельное соединение и вычисляет для него значение сопротивления. Функция `VT` вызывает функцию `VM`, которая обрабатывает или схему из одного элемента, или произвольную схему. Так как в последовательных и параллельных соединениях участвуют, по крайней мере, два элемента, то описание схемы должно начинаться со скобки.

HTML-код приведен в листинге 18.4.

Листинг 18.4. Задача вычисления сопротивления ПП-схемы

```
<HTML>
  <HEAD>
    <TITLE>Задача вычисления сопротивления ПП-схемы</TITLE>
    <script language="JavaScript">
    <!-- //
      var c        //Текущий символ анализируемой схемы
      var r
      var s        // Заданная схема
      var   i
      // Функция cursym выдает очередной символ схемы
      function  cursym()
       { i=i+1; return s.charAt(i)}
      //  Функция  errform фиксирует ошибку
      function errform (s)
       { document.writeln ('Ошибка при задании схемы: ',s) }
      // Функция vF обрабатывает последовательно соединенную схему
      // и выдает для нее значение сопротивления
      function vF()
       { var x
         var y
         y = vT ()    // Значение первого элемента схемы
         while (c  =='+')
```

```
    { c = cursym()
      x = vT()
      y = y+x
    }
   return y
 }
// Функция vT обрабатывает параллельно соединенную схему
// и выдает для нее значение сопротивления
function vT()
 { var x; var y; var z;
   z = vM()
   y = 1/z    // значение первого элемента схемы
   while (c == '*')
    { c = cursym();  x = vM();      y = y+1/x }
   return (1/y)
 }
// Функция vM обрабатывает схему из одного элемента
// или произвольную ПП-схему
function vM()
 { var x; var y; var r
   var cp
   if ((c >='1')&&(c<='9'))
    { r= 1*c
      c = cursym()
    }
   else
    { if (c =='(')
      { c = cursym()
        x  = vM()
        if (!((c == '+')  || (c == '*')))
          { errform('Нет знака')}
        else
          { cp = c
            c = cursym()
            y = vF()
            if (cp == '+')
              { r = x+y}
            else
              { r= y/(y/x + 1)}
```

```
                if ( c != ')')
                  errform('Нет закрывающей скобки')
                else
                  c = cursym()
               }
             }
           else  errform('Неверное начало схемы')
          }
        return r
       }
      // Инициализация глобальных переменных и вызов основной функции
      function main (obj)
       { s= obj.f.value
         i = -1
         c = cursym()
         r = vF()
         obj.fres.value = r
       }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4>Вычисление сопротивления ПП-схемы
        методом рекурсивного спуска</H4>
<pre>
<FORM name="form1">
Введите формулу:   <input type="text" size=30 name="f"><HR>
Значение формулы:  <input type="text" size=30 name="fres"><HR>
<input type="button" value="Выполнить" onClick="main(form1)"><HR>
<input type="reset" value="  Очистить "><HR>
</FORM>
</pre>
  </BODY>
</HTML>
```

На рис. 18.8 приведен результат работы программы. Изобразите схему, которая представлена строкой, и проверьте правильность работы программы.

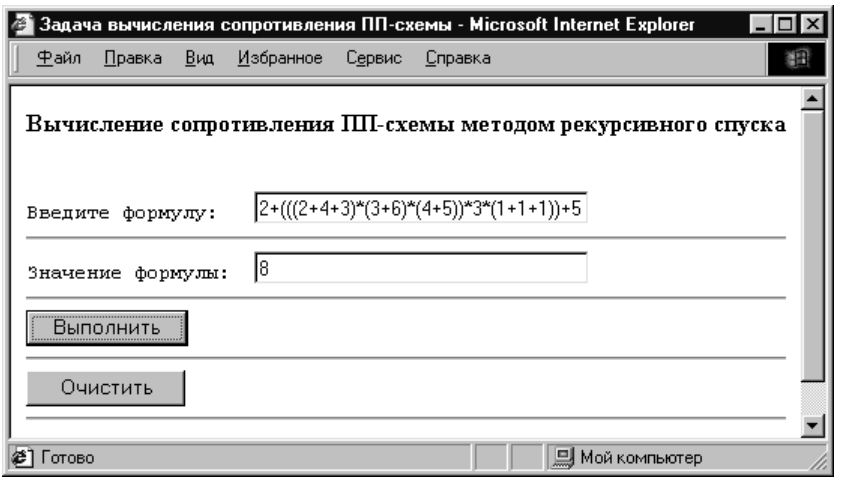
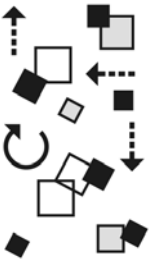


Рис. 18.8. Пример работы сценария вычисления сопротивления заданной ПП-схемы

Упражнения

1. Напишите функцию, которая по формуле, содержащей однобуквенные идентификаторы, круглые скобки, знаки операций сложения, вычитания, умножения, выясняет, правильно ли построена формула.
2. Напишите функцию, которая по формуле в префиксной нотации строит формулу в инфиксной нотации.
3. Напишите функцию, которая по формуле в постфиксной нотации строит формулу в инфиксной нотации.
4. Пусть формула строится из логических переменных, круглых скобок, логического отрицания, логического и, логического или. Напишите сценарий, который преобразует данную формулу в префиксную нотацию.
5. Пусть формула строится из логических переменных, круглых скобок, логического отрицания, логического и, логического или. Напишите сценарий, который преобразует данную формулу в постфиксную нотацию

Глава 19



Решение локальных задач

Расписание занятий

Предположим, что занятия в некотором учебном заведении начинаются по мере комплектования групп. В анкете заполняется дата начала и конца работы группы. Для проведения занятий по определенной дисциплине выбирается день недели. Требуется написать сценарий, который формирует расписание занятий.

После того как пользователь заполнил поля формы (рис. 19.1), указав дату начала и окончания занятий группы и день недели для проведения занятий по некоторой дисциплине, определяется дата первого занятия.

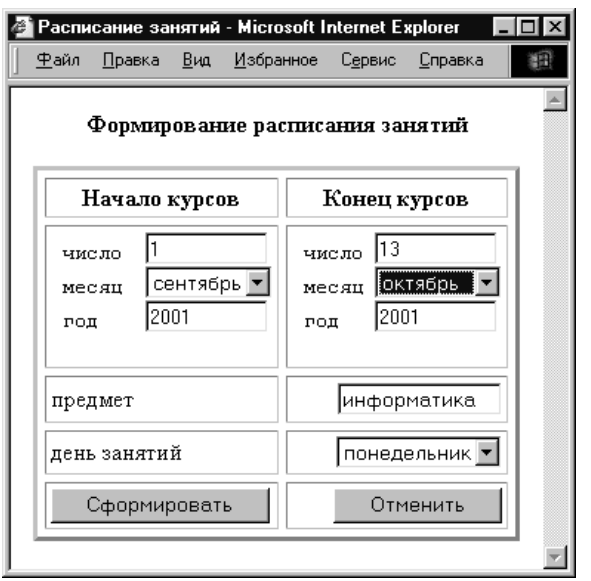


Рис. 19.1. Расписание занятий

Предположим, группа начинает работу 15 февраля 2001 года (четверг), а занятия по заданной дисциплине должны проводиться по средам. Дата пер-

вого занятия — 21 февраля. Далее формируется дата следующего занятия и проверяется, попадает ли сформированная дата в интервал времени, определенный для занятий, т. е. предшествует ли текущая дата дате окончания курсов. Далее формируется очередная строка таблицы расписания. Расписание представляется в виде таблицы. Сначала в строковой переменной `s1` формируется заголовок таблицы. Дата очередного занятия представляет строку таблицы. В таблице будет столько строк, сколько занятий требуется провести с данной группой. Дата очередного занятия формируется в строке `scur`. После анализа дат построение таблицы завершается. В сценарии используются массивы `ident` для хранения названий месяцев и `iday` для хранения названий дней. В строке `s` формируется таблица с расписанием. HTML-код, содержащий сценарий построения расписания, приведен в листинге 19.1.

Листинг 19.1. Формирование расписания занятий

```
<HTML>
  <HEAD>
    <TITLE>Расписание занятий</TITLE>
    <script language="JavaScript">
    <!-- //
      // Формирование по номеру названия месяца
      var ident=new Array ("январь  ", "февраль ", "март    ",
                           "апрель", "май", "июнь",  "июль", "август",
                           "сентябрь",  "октябрь", "ноябрь", "декабрь")
      var iday =new Array ("воскресенье", "понедельник", "вторник",
                          "среда", "четверг",  "пятница", "суббота")
      // Формирование расписания занятий
      function  rasp(obj)
       { // Дата начала занятий
         var d= new Date (obj.begy.value, obj.begm.value, obj.begd.value)
         // Дата окончания занятий
         var w= new Date (obj.endy.value, obj.endm.value, obj.endd.value)
         // дата текущего занятия
         var t=  new Date (obj.begy.value, obj.begm.value,obj.begd.value)
         var s=""
         var n=obj.dt.value          // Выбранный для занятия день недели
         var k=d.getDate()+Number(n)- Number(d.getDay())
         if (n < d.getDay())
         k += 7
         // Определения даты первого занятия
```

```
        t.setDate(k)
        // Формирование заголовка таблицы
        var s1 = "<h4 align=center>Расписание занятий</h4>"
        s1+="предмет <i><b>"+obj.predmet.value+" </b></i><br>"
        s1+="<i>день занятий <b>" +iday[Number(n)]+"</b></i>"
        var sHEAD="<TABLE align=center border=3   bgcolor='#FFFFCC'
cellpadding=3 cellspacing=5>"+ "<TR><th>число </th><th>месяц
</th><th>год  </th></TR>"
        document.write(sHEAD)
        var scur
        var s=s1
        // Поиск дат занятий
        while (t.getYear () <= w.getYear ())
         { // Формирование текущей строки таблицы расписания
           scur="<TR align=center><TD>"+ t.getDate ()+"</TD><TD>"+
                ident[ t.getMonth ()]+
                "</TD><TD>"+t.getYear ()+"</TD></TR>"
           if (t.getYear () < w.getYear ())
            { s+= scur }
           else
            { if (t.getMonth() < w.getMonth())
               { s +=scur }
              else
               { if (t.getMonth() == w.getMonth())
                 { if (t.getDate()<w.getDate())
                    { s+=scur }
                   else
                    { if (t.getDate()==w.getDate())
                       { s+= scur; break }
                    }
                 }
               }
            }
           k=t.getDate()+7
           t.setDate(k)
         }
       if (s==s1)
         alert ("Проверьте даты начала и конца занятий")
       else
```

```
         document.write(s+"</TABLE>")
      }
    //-->
    </script>
  </HEAD>
  <BODY>
    <H4 align=center>Формирование расписания занятий</H4>
    <FORM name="form1">
      <TABLE border=3 align=center  cellpadding=3 cellspacing=5>
        <TR><th>Начало курсов</th><th>Конец курсов</th></TR>
        <TR><TD>
<pre>
 число  <input type="text" name="begd" size=10>
 месяц  <select name="begm" size=1>
                <option value=0>январь
                <option value=1>февраль
                <option value=2>март
                <option value=3>апрель
                <option value=4>май
                <option value=5>июнь
                <option value=6>июль
                <option value=7>август
                <option value=8>сентябрь
                <option value=9>октябрь
                <option value=10>ноябрь
                <option value=11>декабрь
              </select>
 год    <input type="text" name="begy" size=10 value=2001>
</pre></TD>
<TD>
<pre>
 число <input type="text" name="endd" size=10>
 месяц <select name="endm" size=1>
                <option value=0>январь
                <option value=1>февраль
                <option value=2>март
                <option value=3>апрель
                <option value=4>май
```

```
                <option value=5>июнь
                <option value=6>июль
                <option value=7>август
                <option value=8>сентябрь
                <option value=9>октябрь
                <option value=10>ноябрь
                <option value=11>декабрь
              </select>
 год   <input type="text" name="endy" size=10 value=2001>
</pre></TD></TR>
      <TR><TD>предмет</TD>
          <TD align=right>
            <input type="text" name="predmet" size=14
                   value=информатика></TD></TR>
      <TR><TD>день занятий</TD>
          <TD align=right>
            <select name= dt size=1>
              <option value=0>воскресенье
              <option value=1 selected>понедельник
              <option value=2>вторник
              <option value=3>среда
              <option value=4>четверг
              <option value=5>пятница
              <option value=6>суббота
            </select>
          </TD></TR>
      <TR><TD align=left>
            <input type="button" value="Сформировать"
                   onClick="rasp (form1)"><br>
          </TD>
          <TD align=right>
            <input type="reset" value="  Отменить  ">
</TD></TR></TABLE></FORM></BODY></HTML>
```

Для исходных данных, представленных в анкете (рис. 19.1), в результате выполнения сценария будет сформирована таблица (рис. 19.2).

Рис. 19.2. Пример сформированного расписания занятий

Ведомость проведения занятий

Напишем сценарий, в результате работы которого формируется ведомость проведения занятий. Пользователь в анкете задает дату начала и окончания семестра, выбирает день проведения занятий, может указать дисциплину, количество студентов в группе и номер группы. Ведомость представляет собой таблицу, в заголовке которой предусмотрены поля: номер записи, фамилия студента, даты проведения занятий, отметка о зачете.

В этой задаче в отличие от предыдущей динамически формируются сначала столбцы таблицы в зависимости от продолжительности семестра. Далее будут сформированы строки таблицы. В каждой строке должна располагаться информация об одном студенте.

Вычисление даты очередного занятия происходит так, как и в предыдущей задаче.

Анкета для заполнения представлена на рис. 19.3.

Приведем только сценарий решения задачи, необходимые пояснения содержаться в тексте (листинг 19.2).

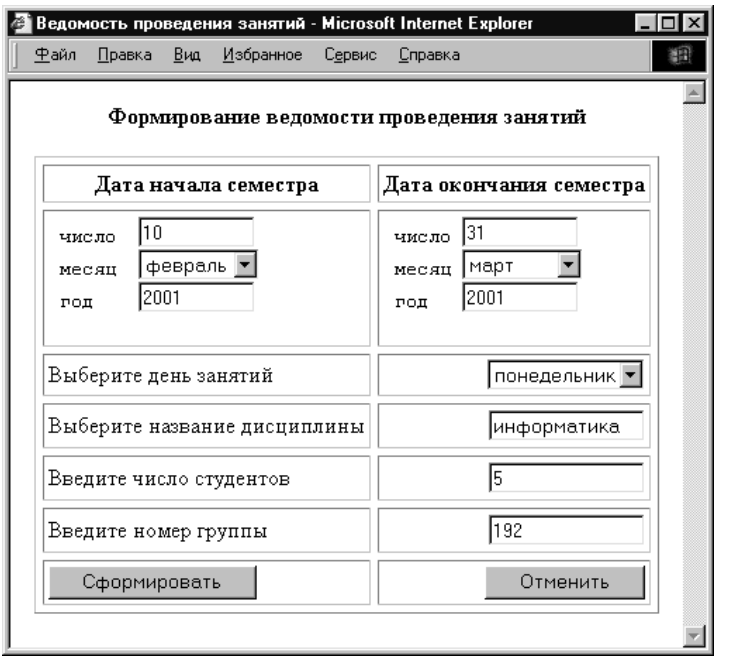


Рис. 19.3. Формирование ведомости

Листинг 19.2. Формирование ведомости проведения занятий

```
<script language="JavaScript">
<!-- //
  var iday =new Array("воскресенье", "понедельник", "вторник", "среда",
                      "четверг", "пятница", "суббота")
  // Формирование ведомости проведения занятий
  function rasp(obj)
   { // Дата начала занятий
     var d= new Date(obj.begy.value, obj.begm.value, obj.begd.value)
     // Дата окончания занятий
     var w= new Date(obj.endy.value, obj.endm.value, obj.endd.value)
      // Дата текущего занятия
     var t= new Date(obj.begy.value, obj.begm.value, obj.begd.value)
     var s=""
     var n=Number(obj.dt.value)   // Выбранный для занятия день недели
     var nn=Number(obj.numst.value)
     var k=d.getDate()+Number(n)- Number(d.getDay())
     if (n < d.getDay())
```

```
    k += 7
    // Определения даты первого занятия
    t.setDate(k)
    // Формирование заголовка таблицы
    var s1 = "<h4 align=center>Ведомость проведения занятий</h4>"
    s1+="<p>дисциплина<i><b> "+obj.predmet.value+"</b></i></p>"
    s1+="<p><i>номер группы </i><b> "+obj.numgr.value+"</b></p>"
    s1+="<i>день занятий "+"<b>"+iday[Number(n)]+"</b></i>"
    var sHEAD="<TABLE align=center border=3 bgcolor='#FFFFCC'"+
              " cellpadding=3 cellspacing=0>"+
              "<TR><th>номер</th><th>фамилия</th>"
    document.write(sHEAD)
    var scur
    var s=s1
    var num=0
    // Поиск дат занятий
    while (t.getYear() <= w.getYear())
     { // Формирование текущей даты проведения занятий
       var mon= t.getMonth ()+1
       var temp=((mon<10) ? ".0" :".")+mon
       scur="<th>"+ t.getDate ()+temp+"</th>"
       if (t.getYear() < w.getYear())
        { s+= scur; num+=1}
       else
        { if (t.getMonth() < w.getMonth())
            {s +=scur; num+=1}
          else
            {if (t.getMonth() == w.getMonth())
               { if (t.getDate()<w.getDate())
                   {s +=scur; num+=1}
                 else
                   {if (t.getDate()==w.getDate())
                     {s+= scur; num+=1; break }
                    }
               }
            }
     }
    k=t.getDate()+7
    t.setDate(k)
}
```

```
  if (s==s1)
    alert ("Проверьте даты начала и конца занятий")
  else
  { s+="<th>отметка о зачете</th></TR>"
    var sn=""
    for (var i=1; i<=nn; i++)
     {sn="<TR><TD>"+i+"</TD>"+"<TD>   </TD>"
       for (var j=1; j<=num+1; j++)
         {sn+="<TD>  </TD>"}
       sn+="</TR>"
       s+=sn
     }
    s+="</TABLE>"
    document.write(s)
    document.write("<p align=center>Число занятий в семест-
pe<b>"+num+"</b></p>")
  }
}
//-->
</script>
```

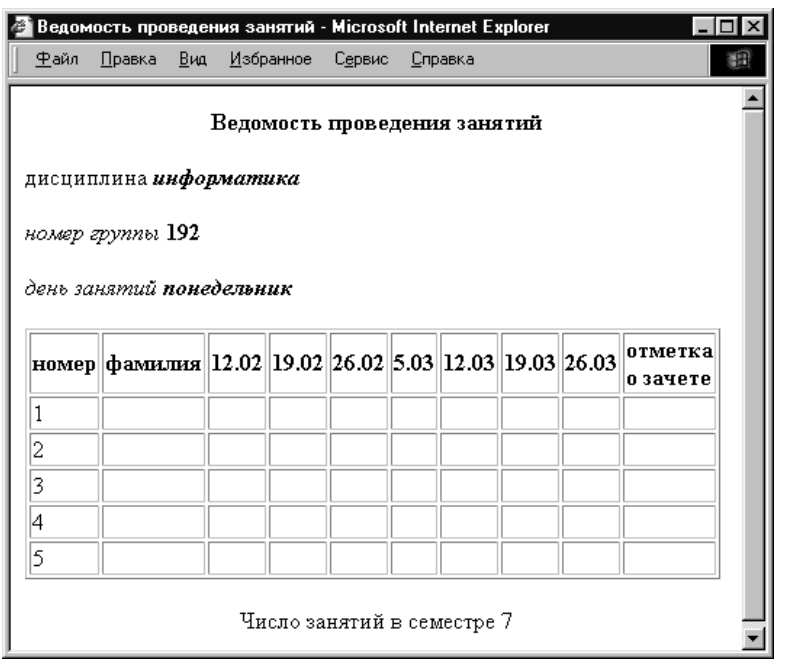


Рис. 19.4. Пример сформированной ведомости

По тем данным, которые представлены в анкете на рис. 19.3, будет сформирована ведомость, вид которой представлен на рис. 19.4. После формирования таблицы будет указано число занятий в заданном семестре.

Если известны личности студентов (например, хранятся в каком-либо массиве), то нетрудно изменить сценарий таким образом, чтобы фамилии располагались в соответствующей строке ведомости.

Ближайший праздник

Напишем сценарий, в результате работы которого рассчитывается количество дней, оставшихся до ближайшего праздника. Если день праздничный, то формируется соответствующее поздравление.

На рис. 19.5 приведен пример работы сценария.

Рис. 19.5. Формирование календаря событий

Кроме сообщения о праздничных датах выводится календарь, в календаре день, в который запускается сценарий, выделен цветом.

HTML-код приведен в листинге 19.3.

Листинг 19.3. Количество дней до ближайшего праздника. Формирование календаря

```
<HTML>
  <HEAD><TITLE>Календать событий. Raskin A.</TITLE></HEAD>
  <BODY bgcolor="#F8F8FF" text="#000000">
    <CENTER>
    <IMG align=center SRC="buket.gif" width=164 height=164>
    <Script Language="JavaScript">
    <!--
      // Переменные для хранения цвета различных элементов таблицы
      var cToday="bgcolor='#BBEEFF'"
      var cEmpty="bgcolor='#00CCFF'"
      var cDay="bgcolor=  '#00CCFF'"
      var cCap="bgcolor=  '#FFA0A0'"
      var cTABLE="bgcolor='#D0D0D0'"
      // Празничный день
      var HolidayDay
      var HolidayMonth
      var HolidayName
      var Left
      // Приветствие
      var Congrate
      // Текущая дата
      var today=new Date()
      var Mon=today.getMonth()+1
      var Day=today.getDate()
      var currDay=today.getDate()
      var currMonth=today.getMonth()
      var currYear=today.getYear()
      var i
      function array(m0,m1,m2,m3,m4,m5,m6,m7,m8,m9,m10,m11)
       {this[0]=m0;  this[1]=m1;  this[2]=m2;  this[3]=m3;
        this[4]=m4;  this[5]=m5;  this[6]=m6;  this[7]=m7;
        this[8]=m8;  this[9]=m9;  this[10]=m10;  this[11]=m11;
       }
      var monames=new array("Январь","Февраль","Март","Апрель","Май",
```

```
                    "Июнь","Июль","Август","Сентябрь",
                    "Октябрь","Ноябрь","Декабрь")
// Количество дней в каждом месяце
var days=new array(31,28,31,30,31,30,31,31,30,31,30,31);
function rDay(ptoday)
 { var vDay=ptoday.getDay();
   vDay=vDay-1;
   if (vDay==-1) vDay=6
   return vDay
 }
//Формирование приветствия
function GetHol()
 { var HolMonth=new Array(1,1 ,2 ,3 , 4, 5, 5, 6,6, 9,11,12);
   var HolDays= new Array(7,13,23,8 , 1, 1, 9, 1,12, 1, 7,31);
   var HolNames= new Array("Рождества","Старого Нового Года",
                           "23 Февраля",
                           "8 Марта","1 Апреля — Дня Смеха",
                           "1 Мая — Праздника Весны и Труда",
                           "9 Мая — Дня Победы",
                           "1 Июня — Дня Защиты Детей",
                           "12 Июня — Дня Независимости",
                           "1 Сентября — Дня Знаний",
                           "7 Ноября — Дня Примирения",
                           "Нового Года");
   var HolNow= new Array(" с Рождеством! ",
            " с праздником <br>Старого Нового Года! ",
            " защитников Отечества <br>с праздником 23 Февраля!",
            " женщин <br>с праздником 8 Марта!",
            " с 1 Апреля <br> — Днем Смеха!",
            " с 1 Мая <br> — Праздником Весны и Труда!",
            " с 9 Мая <br> — Днем Победы!",
            " с 1 Июня <br> — Днем Защиты Детей!",
            " с 12 Июня <br> — Днем Независимости России! ",
            " учащихся с 1 Сентября <br> — Днем Знаний! ",
            " с 7 Ноября <br> — Днем Примирения! ",
            " с Новым Годом! ");
  for(i=0; i<=HolMonth.length; i++)
   { if (Mon<=HolMonth[i])
      { if (Mon==HolMonth[i])
```

```
              { if (Day<=HolDays[i])
                 { HolidayDay=HolDays[i]
                   HolidayMonth=HolMonth[i]
                   HolidayName=HolNames[i]
                   Congrate=HolNow[i]
                   Left=HolidayDay-Day
                   break;
                 }
              }
             else
              { HolidayDay=HolDays[i]
                HolidayMonth=HolMonth[i]
                HolidayName=HolNames[i]
                Congrate=HolNow[i]
                var HolDate=new Date(today.getYear(),
                                     HolidayMonth-1,HolidayDay)
                Left=Math.round((HolDate.getTime()-
today.getTime())/(24000*3600))
                break;
              }
           }
         }
        if (Left==0)
         document.write("<h3><I>Поздравляем всех "+Congrate+"</I></h3>")
        else
         document.write("<h3><I>До "+HolidayName+"<br> осталось дней: "+
                        Left+" !</I></h3>")
        }
      // Построение календаря
      function showCalendar(Month,Year)
       { document.write("<B><I>"+monames[Month]+" "+Year+"</B><I>")
         firstDay=new Date(Year,Month,1)
         startDay=rDay(firstDay)
         if(((Year%4==0)&&(Year%100!=0))||(Year%400==0))
           days[1]=29;
         else
           days[1]=28;
         document.write("<TABLE "+cTABLE+
                        " CallSpacing=1 CellPadding=1 Border=1>");
```

```
        document.write("<TR"+cCap+
                 "><th>Пн</th><th>Вт</th><th>Ср</th>"+
                 "<th>Чт</th><th>Пт</th><th>Сб</th><th>Вс</th></TR>")
        document.write("<TR align=Right>");
        var column=0
        for (i=0; i<startDay; i++)
         { document.write("<TD ",cEmpty,"> </TD>");
           column++;
         }
        for (i=1; i<=days[Month]; i++)
         { if((i==currDay)&&(Month==currMonth)&&(Year==currYear))
            { document.write("<TD ",cToday,">",i,"</TD>")}
           else
            { document.write("<TD ",cDay,">","<b>"+i+"</B>","</TD>")
            }
        column++;
        if (column==7)
         { document.write("</TR><TR align=Right>");
           column=0;
         }
       }
       document.write("</TR></TABLE>");
       }
     //Today=new Date()
     GetHol();
     showCalendar(today.getMonth(),today.getYear());
   //-->
   </Script>
   В календарь занесены основные праздники и поздравления к ним
  </BODY>
</HTML>
```

Латинский квадрат

Напишем сценарий построения латинского квадрата. В квадрате размером $N \times N$ в каждой из клеток требуется поставить одно из чисел 1, 2, …, N так, чтобы сумма чисел, стоящих в каждом вертикальном, горизонтальном ряду и по диагонали, равнялась одному и тому же числу $1 + 2 + 3 + ... + N$.

Разобьем окно документа на две горизонтальные области с помощью фреймов. В верхней части поместим условие задачи и форму для ввода размера таблицы. При нажатии кнопки **OK** в нижнем фрейме будет помещен документ, содержащий таблицу, представляющую собой латинский квадрат (рис. 19.6).

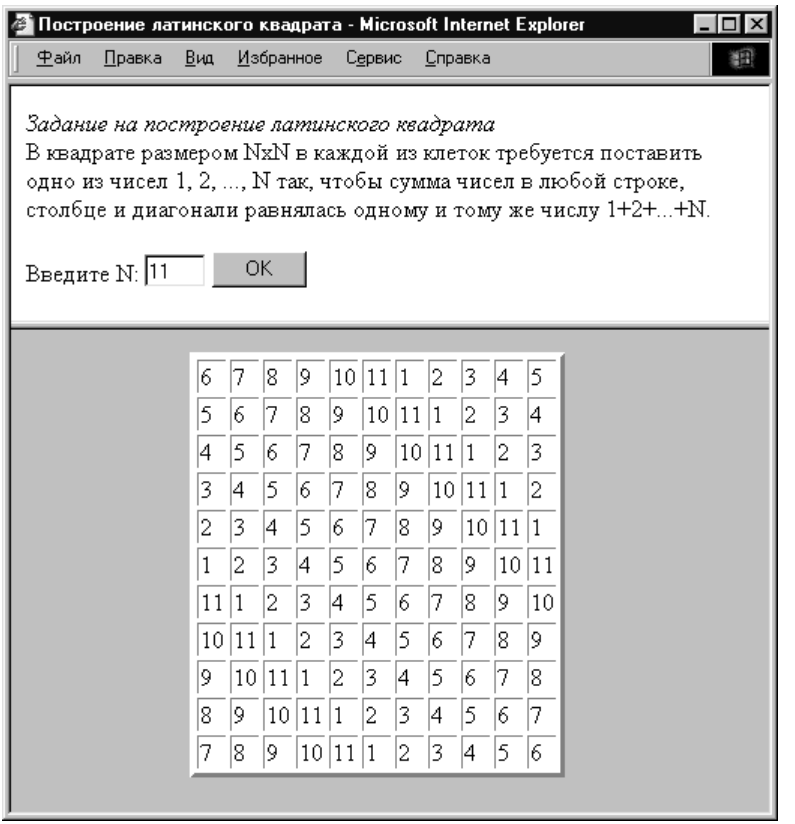


Рис. 19.6. Латинский квадрат для $N = 11$

Для разбиения окна на две области требуется создать документ, описывающий фреймовую структуру. В данном случае документ имеет простую структуру и может быть задан, например, как в листинге 19.4.

Листинг 19.4. Фреймовая структура для задачи "Построение латинского квадрата"

```
<HTML>
  <HEAD>
    <TITLE>Построение латинского квадрата</TITLE>
```

```
  </HEAD>
  <frameset ROWS="35%, *">
    <frame name="lattab" SRC="formtab.html">
    <frame name="bottom">
  </frameset>
</HTML>
```

Таблица формируется по строкам. Элемент, который заносится в таблицу с *i*-ым номером строки и *j*-ым номером столбца, определяется по формуле `Math.round(Number((j+k-i+3*n-1)%n+1))`, причем значение *k* зависит от того, четное или нечетное заданное значение *n*. При переходе к формированию следующей строки происходит как бы циклический сдвиг предыдущей.

Переменная `out` предназначена для работы с документом, который загружается в нижний фрейм, это обеспечивается выполнением присваивания `out = top.frames['bottom'].document`. Применением к `out` метода `close()` закрывается поток вывода в документ. Метод `open()` открывает поток вывода в документ. Далее в сценарии формируются элементы таблицы с помощью методов `write()` и `writeln()`, например, так

```
out.writeln('<BODY bgcolor=silver><CENTER>')
```

Полностью HTML- код документа приведен в листинге 19.5.

Листинг 19.5. Построение латинского квадрата

```
<HTML>
  <HEAD>
    <TITLE>Латинский квадрат</TITLE>
    <script language="Javascript">
    <!--//
    function solvetab()
     { var n = Number(document.forms['form1'].num.value);
       var k
       if (n%2==0)
         k= Number(n/2)
       else
         k= Number((n+1)/2);
       var out = top.frames['bottom'].document;
       out.close()
       out.open()
       out.writeln('<BODY bgcolor=silver><CENTER>');
       out.writeln('<TABLE bgcolor=white COLS='+n+' ROWS='+n+
```

```
                    ' border=3>');
        for (i=0; i<n; i++)
         { out.writeln('<TR>');
           for (j=0; j<n; j++)
            { out.writeln('<TD> '+Math.round(Number((j+k-i+3*n-1)%n+1))+
                          ' ')}
              out.writeln('</TR>');
            }
           out.write('</TABLE>');
           out.writeln('</CENTER><BODY>');
         }
    -->
    </script>
  </HEAD>
  <BODY>
    <i>Задание на построение латинского квадрата</i><br>
    В квадрате размером NxN в каждой из клеток требуется поставить
    одно из  чисел  1, 2, ..., N так, чтобы сумма чисел в любой строке,
    столбце и диагонали равнялась одному и тому же числу 1+2+...+N.<br>
    <FORM name="form1">
      Введите N: <input name="num" TYPE="text" size=4>
      <input TYPE="button" value="    OK    " onclick="solvetab()">
    </FORM>
  </BODY>
</HTML>
```

Упражнения

1. Задаются две литеры — латинская буква (от *a* до *h*) и цифра (от 1 до 8). Рассматривая их как координаты поля шахматной доски, на которой находится выбранная пользователем фигура, напишите сценарий построения шахматной доски с выбранной фигурой. Требуется указать все поля, которые "бьет" данная фигура (рис. 19.7).
2. Напишите сценарий, который позволяет либо задать начальную расстановку фигур на шахматной доске (рис. 19.8), либо расставить фигуры по указаниям пользователя.

   Пользователь должен иметь возможность переставлять фигуры на доске, указывая координаты начала и конца хода. На рис. 19.9 изображена комбинация после выполнения нескольких ходов.

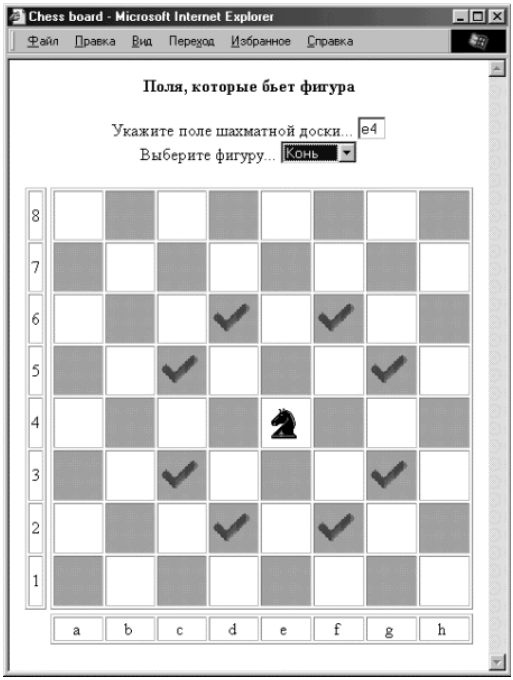


Рис. 19.7. Указание полей, которые "бьет" конь

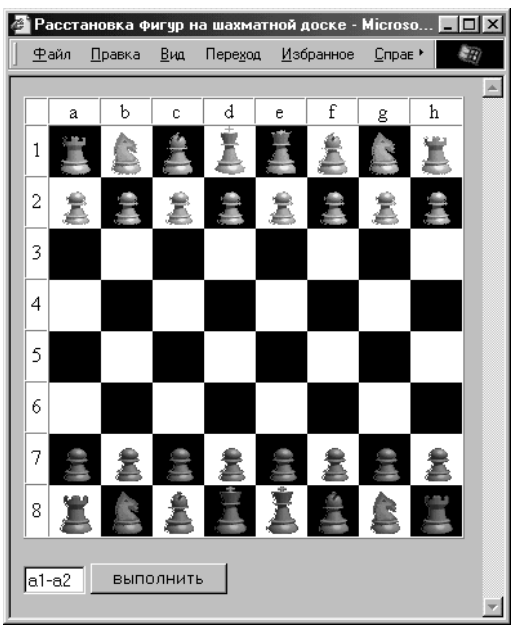


Рис. 19.8. Расстановка фигур при игре в шахматы

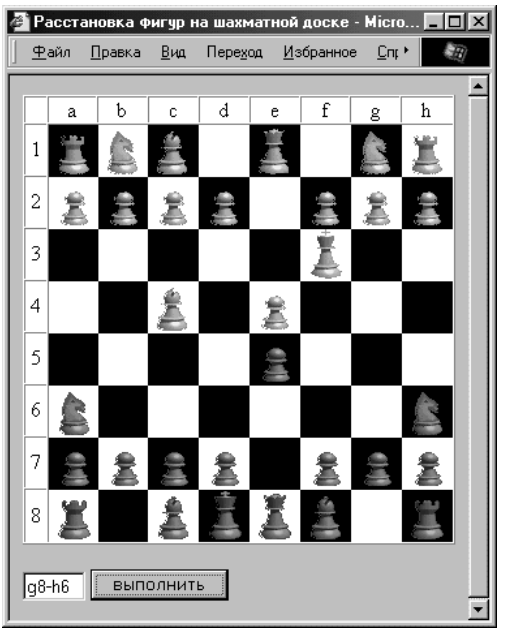


Рис. 19.9. Комбинация фигур после выполнения нескольких ходов

3. Пользователь на шахматной доске расставляет два ферзя. Укажите поля, на которые может пойти белый ферзь, чтобы не попасть под удар черного. В сценарии предусмотрите случай, когда пользователь может произвольно выбрать две фигуры и расставить их на шахматной доске.
4. У белых на доске только король, у черных — король, конь, слон. Напишите сценарий, который осуществляет расстановку фигур на доске и при анализе ситуации характеризует положение белых с помощью слов "мат", "шах", "пат", "обычная позиция".
5. Напишите сценарий, который расставляет на шахматной доске 8 ладей так, чтобы ни одна из них не угрожала другой.
6. Напишите сценарий, который расставляет на шахматной доске 8 ферзей так, чтобы ни один из них не угрожал другому. Один из вариантов расстановки приведен на рис. 19.10.
7. Напишите сценарий, который по заданному расположению одного ферзя расставит еще 7 так, чтобы ни один из них не угрожал другому. Если пользователь первого ферзя поставил на поле A3, то следующие фигуры будут расставлены так, как показано на рис. 19.11.
8. Напишите сценарий, который выводит изображение шахматной доски с указанием порядка ходов, которые необходимо сделать коню для того, чтобы обойти все клетки доски, побывав в каждой из них только один

раз. Пользователь может задать начальную позицию коня. Если начальное положение коня — A6, то последовательность обхода конем доски изображена на рис. 19.12.

Рис. 19.10. Расстановка восьми ферзей

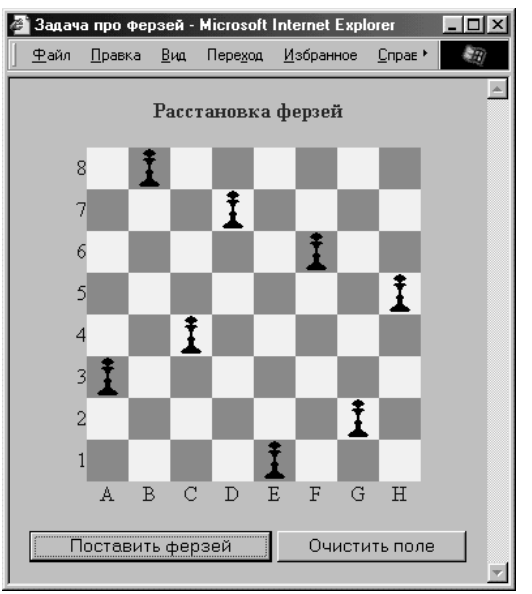


Рис. 19.11. Расстановка семи ферзей при заданном положении восьмого

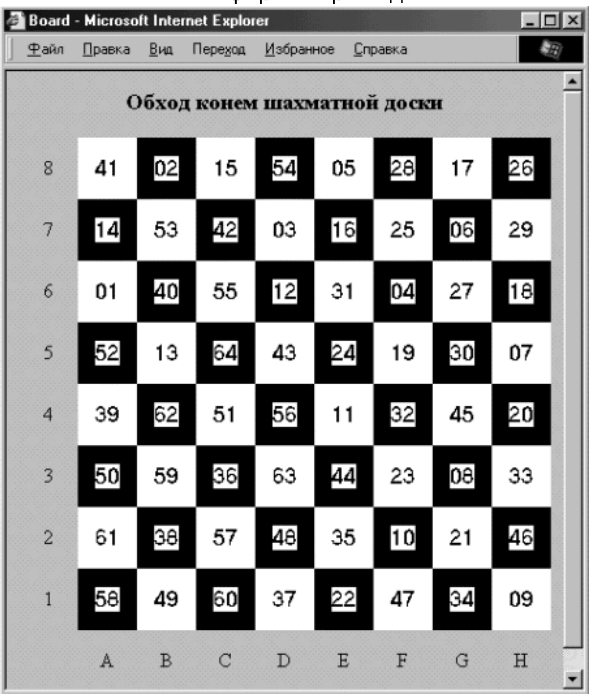
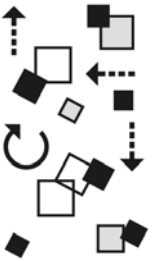


Рис. 19.12. Пример обхода конем шахматной доски

9. Напишите сценарий, который расставляет на шахматной доске 12 коней, таким образом, что каждое поле шахматной доски находится под ударом одного из коней.
10. Напишите сценарий, определяющий такую расстановку 5 ферзей, при которой каждое поле будет находиться под ударом одного из них.
11. Напишите сценарий, определяющий такую расстановку 8 слонов, при которой каждое поле будет находиться под ударом одного из них.
12. Напишите сценарий, при работе которого устанавливается начальная комбинация при игре в шашки, либо подготавливается собственная. При этом пользователю предоставляется возможность выбрать шашки и установить их на соответствующее поле доски. Доска должна иметь стандартную нумерацию и раскраску.
13. Лабиринт задается прямоугольной матрицей с целыми значениями. Разрешается сделать ход только вниз или вправо на клетку с большим значением. Определите, существует ли путь в лабиринте от клетки левого верхнего конца к правому нижнему. Напишите сценарий, который рисует лабиринт и найденный путь.

14. Лабиринт задается прямоугольной матрицей с целыми значениями. Разрешается сделать ход только вниз или вправо на клетку с большим значением. Определите, существует ли путь в лабиринте от заданного пользователем входа до заданного выхода. Напишите сценарий, который рисует лабиринт и найденный путь.
15. Лабиринт задается прямоугольной матрицей с целыми значениями. Разрешается сделать ход по любому из 4-х направлений на клетку с большим значением. Напишите сценарий, определяющий, существует ли путь в лабиринте. Требуется нарисовать лабиринт и путь, найденный в нем.
16. *Магическим квадратом* порядка n называется квадратная таблица размера $n \times n$, составленная из чисел 1, 2, ..., n^2 так, что суммы по каждому столбцу, каждой строке и каждой из двух диагоналей равны между собой. Задается порядок n. Постройте магический квадрат порядка n.
17. *Задача о рюкзаке*. Имеется m различных предметов, известен вес каждого и стоимость. Определите, какие предметы надо положить в рюкзак, чтобы общий вес не превышал заданного значения, а общая стоимость была максимальной.

Глава 20



Формулы исчисления высказываний

Общие сведения

В предыдущих главах для самостоятельной работы предлагались упражнения, которые располагались в конце каждой главы. При выполнении упражнения следовало написать сценарий решения некоторой задачи. В настоящей главе изложение материала будет изменено. Упражнение может быть предложено сразу после рассмотрения теоретического материала или примера, иллюстрирующего теорию. Кроме того, упражнения не обязательно предполагают создание сценария. Отдельно формулируется задание на создание сценария.

Материал этой главы посвящен введению в один из разделов искусственного интеллекта — автоматическое доказательство теорем.

В обычной жизни человек принимает решения в зависимости от конкретной ситуации. Предположим, что нам известны некоторые факты (или ранее уже доказанные утверждения) F_1, F_2, …, F_n, и нас интересует, следует ли некоторое утверждение G из утверждений F_1, F_2, …, F_n. Утверждение, что G логически следует из утверждений F_1, F_2, …, F_n называют *теоремой*. *Доказательство теоремы* — рассуждения, позволяющие установить, что теорема верна.

Математическая логика — наука о правильных математических рассуждениях, о математическом мышлении. Впервые правила рассуждений систематизировал Аристотель. Однако как наука математическая логика сложилась лишь в середине XIX века, когда Джордж Буль ввел логические связки и исчисление высказываний. Математическая логика изучает:

- структуру математических высказываний;
- исходные постулаты математики (аксиомы и правила вывода);
- математические доказательства (выводы);
- истинные математические утверждения (теоремы, леммы).

и многое другое. Одной из задач математической логики является формализация понятия доказательства.

Для описания утверждений можно использовать *формулы исчисления высказываний*. Каждое высказывание либо истинно, либо ложно. Можно строить

составные высказывания, используя *логические связки*: отрицание (~), конъюнкцию (&), дизъюнкцию ($\vee$), импликацию ($\Rightarrow$) и эквивалентность ($\Leftrightarrow$). Логические константы "истина" и "ложь" будем обозначать буквами И и Л соответственно. Логические переменные (в математической логике их называют пропозициональные переменные) будем обозначать большими буквами латинского алфавита.

Унарная связка ~ меняет значение высказывания на противоположное. В табл. 20.1 представлены результаты логических связок таблицы истинности.

Таблица 20.1. *Таблица истинности бинарных связок*

A	B	A & B	A $\vee$ B	A $\Rightarrow$ B	A $\Leftrightarrow$ B
И	И	И	И	И	И
И	Л	Л	И	Л	Л
Л	И	Л	И	И	Л
Л	Л	Л	Л	И	И

В исчислении высказываний логические константы и логические переменные называются *атомами* и считаются простейшими логическими формулами. Кроме атомов логической формулой считается отрицание логической формулы, а также $(F_1 \oplus F_2)$, где F_1, F_2 — логические формулы, а знак $\oplus$ обозначает одну из бинарных логических связок. Других формул в исчислении высказываний нет.

Мы видим, что при построении сложных логических формул появляется большое количество круглых скобок. Для сокращения их числа можно использовать старшинство логических связок. Самой старшей является унарная связка "отрицание", которая выполняется в первую очередь и применяется к непосредственно следующей за ней формуле. Бинарные логические связки, приведенные выше в таблице истинности, перечислены по убыванию их старшинства. Связки одного старшинства применяются в порядке их следования слева направо. Если учитывать старшинство операций, то часть скобок можно опустить. Например, формула

$$(((P \ \& \sim Q) \Rightarrow R) \Rightarrow (P \ \& \ R))$$

может быть записана вообще без скобок:

$$P \ \& \sim Q \Rightarrow R \Rightarrow P \ \& \ R.$$

Формулу исчисления высказываний будем называть *постоянной*, если она не содержит переменных.

Для того чтобы вычислить значение постоянной формулы

$$\text{И} \ \& \sim \text{Л} \Rightarrow \text{И} \Rightarrow \text{И} \ \& \ \text{И}.$$

расставим в ней скобки и получим следующую формулу:

$$(((И \ \& \sim Л) \Rightarrow И) \Rightarrow (И \ \& \ И)).$$

Далее, вычисляя сначала выражения в скобках по таблице истинности, получаем значение всей формулы — истина.

Упражнения

1. Вычислите значение следующих формул исчисления высказываний:
   - $((И \Rightarrow Л) \Rightarrow И) \ \& \ (И \Rightarrow (И \Rightarrow Л)) \Rightarrow Л$;
   - $(И \Leftrightarrow (И \Leftrightarrow Л)) \ \& \sim Л$;
   - $(И \Rightarrow (Л \Rightarrow И)) \Leftrightarrow \sim (Л \ \& \ И)$.
2. Предположим, что мы разговорились с тремя жителями *A*, *B* и *C*, о каждом из которых известно, что он либо рыцарь, либо лжец. Рыцари всегда говорят правду, лжецы всегда лгут. Двое из них (*A* и *B*) высказали следующие утверждения.
   - *A*: Мы все лжецы.
   - *B*: Один из нас рыцарь.

   Кто из трех жителей *A*, *B* и *C* рыцарь и кто лжец?

   Запишите утверждения жителей острова с помощью формул исчисления высказываний.

Вычисление значения постоянной логической формулы

Создадим сценарий, который облегчает ввод постоянных пропозициональных формул. При нажатии кнопки, соответствующей логическим константам или логическим операциям, определенная операция появляется в поле ввода так, как показано на рис. 20.1. После построения формулы и нажатия кнопки **Вычислить** определяется значение формулы.

Напомним, что при построении логических формул в языке JavaScript разрешено использовать три логические операции: отрицание (`!`), логическое И (`&&`) и логическое ИЛИ (`||`). При нажатии кнопки соответствующая операция появляется в текстовом поле, причем знак операции должен быть тем, который принят в языке JavaScript. Подобная формула может быть вычислена, если подать ее в качестве параметра методу `eval`. Если же формула содержит знаки импликации ($\Rightarrow$) и логической эквивалентности ($\Leftrightarrow$), то требуется провести анализ формулы и вычислить ее значение с помощью определяемых программистом функций. HTML-код документа, содержащего сценарий решения задачи, представлен в листинге 20.1.

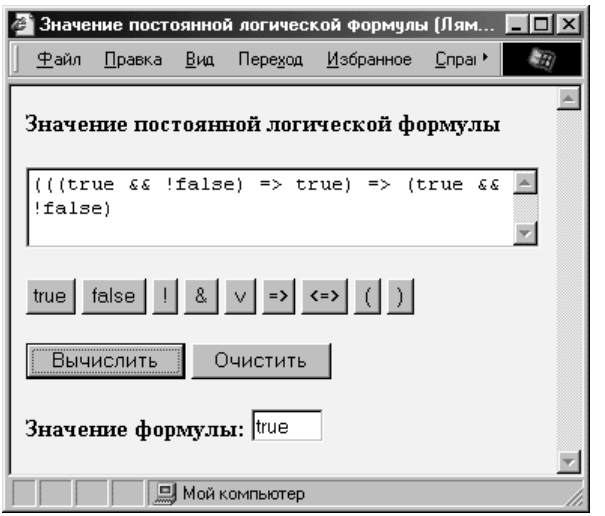


Рис. 20.1. Построитель постоянных логических формул

Листинг 20.1. Значение постоянной логической формулы

```
<HTML>
  <HEAD>
    <TITLE>Значение постоянной логической формулы. (Лямов К.)</TITLE>
    <script>
    <!-- //
      var pos = 0;
      var lex;
      var falseSym = "false";
      var trueSym = "true";
      var notSym = "!";
      var andSym = "&&";
      var orSym = "||";
      var impSym = "=>";
      var equSym = "<=>";
      // Вычисление значения формулы
      function evaluateFormula()
       { pos = 0;
         document.myForm.result.value = valFormula()
       }
      function add(str)
```

```
    { document.myForm.logFormula.value += str}
  // Выбор для анализа следующего символа
  function nextSym()
    { var t = ' ';
      while (t == ' ' && pos <
document.myForm.logFormula.value.length)
       {t = document.myForm.logFormula.value.charAt(pos++)}
      return t;
    }
  // Анализ очередной лексемы
  function nextLex()
    { var tmp = nextSym();
      switch (tmp)
       { case '&': pos++;    return andSym;
         case '|': pos++;    return orSym;
         case '<': pos+=2;   return equSym;
         case '=': pos++;    return impSym;
         case 't': pos += 3; return trueSym;
         case 'f': pos += 4; return falseSym;
         default:  return tmp;
       }
    }
  // Вычисление значения логической эквивалентности
  function valFormula()
    { var left, right;
      left = valImp();
      while (lex == equSym)
       { right = valImp();
         left = ((left) == (right));
       }
      return left;
    }
  // Вычисление импликации
  function valImp()
    { var left, right;
      left = valSimple();
      while (lex == impSym)
       { right = valSimple();
         left = (!(left) || (right)) }
```

```
        return left;
      }
     function valSimple()
      { var s = "";
        while (true) {
          lex = nextLex();
          //alert("Lex: "+lex);
          switch (lex)
           { case trueSym:
             case falseSym:
             case andSym:
             case orSym:
             case notSym: s += lex;
             case ')':    break;
             case '(':    var y = eval(s + valFormula());  return y;
             default:  return eval(s);
           }
        } //while
      } //valSimple
    //-->
  </script>
</HEAD>
<BODY bgcolor="F8F8FF">
  <H4>Значение постоянной логической формулы</H4>
  <FORM name=myForm>
    <textarea cols=40 rows=3 name=logFormula></textarea><br>
    <input type=button value="true" onClick="add('true')">
    <input type=button value="false" onClick="add('false')">
    <input type=button value=" ! "  onClick="add('!')">
    <input type=button value=" & "  onClick="add(' && ')">
    <input type=button value=" v "  onClick="add(' || ')">
    <input type=button value="=>"   onClick="add(' => ')">
    <input type=button value="<=>"  onClick="add(' <=> ')">
    <input type=button value=" ( " onClick="add('(')">
    <input type=button value=" ) " onClick="add(')')"><br><br>
    <input type=button value="Вычислить" onClick="evaluateFormula()">
    <input type=button value="Очистить"
           onClick=" document.myForm.logFormula.value = '';
```

```
      document.myForm.result.value = '' "><br><br>
      <B>Значение формулы:</B> <input type=text name=result size=5>
    </FORM>
  </BODY>
</HTML>
```

Если задать значения всем входящим в формулу переменным, то можно вычислить результат всей формулы. В этом случае говорят, что задана *интерпретация*. В исчислении высказываний каждой формуле соответствует конечное число интерпретаций.

Вычисление значения формулы в заданной интерпретации

Необходимо написать сценарий вычисления значения логической формулы в заданной интерпретации. Пусть при построении формулы используются только три переменные *a*, *b*, *c*, значения которых указывает пользователь. Для ввода логической формулы применяется построитель формул так, как показано на рис. 20.2.

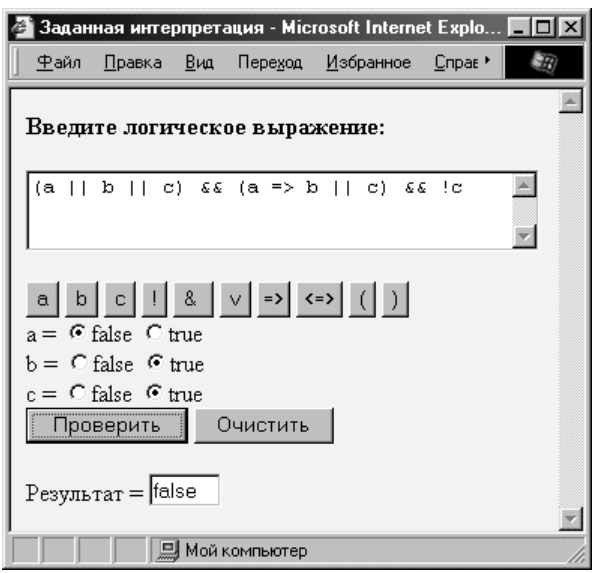


Рис. 20.2. Значение формулы

Полностью написать сценарий предлагается в качестве упражнения.

Соотношение формул

В общем случае при анализе формулы следует учесть количество различных переменных, входящих в формулу. Так как каждая переменная может принимать только два значения, то число различных интерпретаций конечно.

Формулы бывают *тождественно истинными* (или *общезначимыми*) — это формулы истинные в любой интерпретации. *Тождественно ложными* (или *противоречивыми*) называются формулы, ложные в любой интерпретации. Наконец, *выполнимыми* называются формулы, допускающие указание интерпретации, в которой эта формула истинна. Если две формулы имеют одинаковые значения при любых возможных интерпретациях, то говорят, что они *равнозначны* или *эквивалентны*. Равнозначность формул обозначают знаком $\leftrightarrow$. Обозначим знаком $\blacksquare$ любую общезначимую формулу и знаком $\square$ любую противоречивую формулу. Наиболее распространенные равнозначные формулы приведены в табл. 20.2.

Таблица 20.2. *Равнозначные формулы*

Формула	Формула
$A \leftrightarrow B \leftrightarrow (\sim A \vee B) \,\&\, (\sim B \vee A)$	
$(A \Rightarrow B) \leftrightarrow \sim A \vee B$	
$A \vee (B \vee C) \leftrightarrow (A \vee B) \vee C$	$A \,\&\, (B \,\&\, C) \leftrightarrow (A \,\&\, B) \,\&\, C$
$A \vee B \leftrightarrow B \vee A$	$A \,\&\, B \leftrightarrow B \,\&\, A$
$A \vee (B \,\&\, C) \leftrightarrow (A \vee B) \,\&\, (A \vee C)$	$A \,\&\, (B \vee C) \leftrightarrow (A \,\&\, B) \vee (A \,\&\, C)$
$\sim (\sim A) \leftrightarrow A$	
$A \vee \sim A \leftrightarrow \blacksquare$	$A \,\&\, \sim A \leftrightarrow \square$
$A \vee \square \leftrightarrow A$	$A \,\&\, \square \leftrightarrow \square$
$A \vee \blacksquare \leftrightarrow \blacksquare$	$A \,\&\, \blacksquare \leftrightarrow A$
$\sim (A \vee B) \leftrightarrow \sim A \,\&\, \sim B$	$\sim (A \,\&\, B) \leftrightarrow \sim A \vee \sim B$

Нетрудно доказать, что любую формулу исчисления высказываний можно преобразовать к равнозначной формуле, которая содержит только три логические связки: отрицание, конъюнкцию и дизъюнкцию. Действительно, доказав равнозначность формул $A \Rightarrow B$ и $\sim A \vee B$, а также формул $A \Leftrightarrow B$ и $(A \,\&\, B) \vee (\sim A \,\&\, \sim B)$, получим то, что требуется. Некоторые часто используемые правила получили специальные названия, например:

- закон двойного отрицания: $\sim \sim A \leftrightarrow A$;
- закон контрапозиции: $A \Rightarrow B \leftrightarrow \sim B \Rightarrow \sim A$;
- законы де Моргана: $\sim (A \vee B) \leftrightarrow \sim A \,\&\, \sim B$ и $\sim (A \,\&\, B) \leftrightarrow \sim A \vee \sim B$.

Пусть заданы две формулы $(A \Rightarrow B) \Rightarrow C$ и $A \Rightarrow (B \Rightarrow C)$. Требуется определить, эквивалентны ли они. Для того чтобы решить задачу, мы переберем все интерпретации и в каждой из интерпретаций вычислим значение формулы. Результаты приведены в табл. 20.3.

Таблица 20.3. *Таблица истинности для формул $(A \Rightarrow B) \Rightarrow C$ и $A \Rightarrow (B \Rightarrow C)$*

№	A	B	C	$(A \Rightarrow B) \Rightarrow C$	$A \Rightarrow (B \Rightarrow C)$
1	И	И	И	И	И
2	И	И	Л	Л	Л
3	И	Л	И	И	И
4	И	Л	Л	И	И
5	Л	И	И	И	И
6	Л	И	Л	Л	И
7	Л	Л	И	И	И
8	Л	Л	Л	Л	И

После вычислений видно, что значения формул не совпадают в некоторых интерпретациях, поэтому заданные формулы неравнозначны.

Построение таблицы истинности

Напишем сценарий, который для логической формулы строит таблицу истинности. В формуле разрешено использовать операции отрицания, дизъюнкции и конъюнкции; переменные представляются буквами латинского алфавита. Вид документа представлен на рис. 20.3.

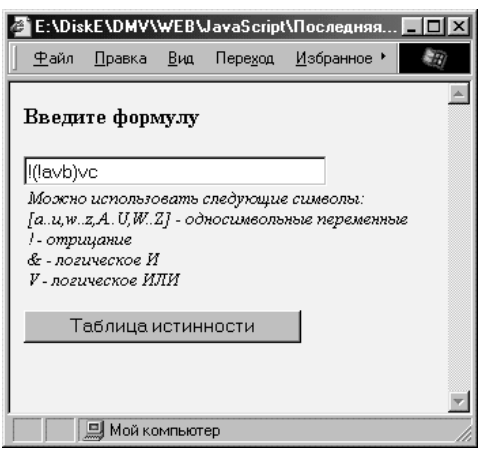


Рис. 20.3. Формирование таблицы истинности для заданной формулы

Если после ввода формулы нажать кнопку **Таблица истинности**, то будет выведена таблица, в которой указаны все возможные интерпретации и значение формулы в каждой интерпретации. Для формулы, введенной в строку на рис. 20.3, будет сформирован следующий документ, содержащий таблицу истинности (рис. 20.4).

Рис. 20.4. Таблица истинности для заданной формулы

HTML-код документа, который по заданной формуле строит таблицу истинности, приведен в листинге 20.2.

Листинг 20.2. Построение таблицы истинности

```
<HTML>
  <HEAD>
    <SCRIPT language="JavaScript" src="term.js"></SCRIPT>
  </HEAD>
  <BODY  bgcolor="F8F8FF">
    <FORM name=data>
      <H4>Введите формулу</H4>
        <input type=text size=30 name=formula value="a&bVc&b"><br>
```

```
        <small><i> Можно использовать следующие символы:<br>
         [a..u,w..z,A..U,W..Z] —  односимвольные переменные<br>
         ! —  отрицание<br>
         & —  логическое И<br>
         V —  логическое ИЛИ<br> <br>
        <input type="button" value="Таблица истинности"
               onclick="TABLE()"><br>
    </FORM><br>
  </BODY>
</HTML>
```

Функция, осуществляющая построение таблицы, хранится во внешнем файле. Содержимое файла приведем позже, а пока рекомендуется написать сценарий в качестве упражнения.

Упражнение

Для каждой из следующих формул определите, является ли она общезначимой, противоречивой или выполнимой:

- $\sim (\sim (P \Rightarrow Q)) \Rightarrow (P \Rightarrow Q)$
- $\sim \ (P \,\&\, T \vee R) \Rightarrow \sim P$
- $(P \Rightarrow Q \,\&\, R \,\&\, T) \Rightarrow (\sim (Q \,\&\, R \,\&\, T) \Rightarrow P)$
- $P \vee (R \Rightarrow Q \,\&\, T)$
- $P \vee (P \,\&\, F \Rightarrow Q \,\&\, B)$
- $\sim P \Rightarrow (P \vee T \vee R \,\&\, F)$

Определение выполнимой, общезначимой и противоречивой формулы

Напишем сценарий, который определяет, является ли заданная формула выполнимой, общезначимой или противоречивой. Для простоты будем считать, что в формуле используются операции отрицания, конъюнкции и дизъюнкции.

При анализе формулы перебираются интерпретации. Вычисляется значение формулы в начальной интерпретации, а затем, если при переходе к следующей интерпретации значение формулы изменилось на противоположное, то она выполнима. Если же при всех интерпретациях значение формулы не изменилось, то она либо общезначима, либо противоречива. Результат выполнения сценария приведен на рис. 20.5.

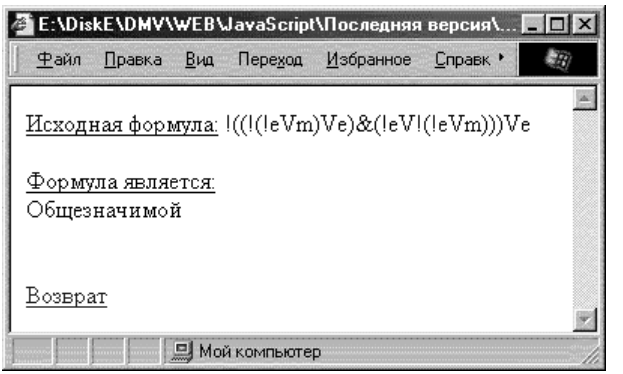


Рис. 20.5. Определение типа формулы

Если формула выполнима, то кроме определения типа формулы требуется построить таблицу истинности.

HTML-код приведен в листинге 20.3.

Листинг 20.3. Формула выполнимая, общезначимая, противоречивая

```
<HTML>
  <HEAD>
    <script language="JavaScript" src="term.js"></SCRIPT>
  </HEAD>
  <BODY>
    <FORM name=data>
      <b>Введите формулу </b>
      <input type=text size=30 name=formula value="a&bVc&b"><br>
      <small><i> Можно использовать следующие символы:<br>
       [a..u,w..z,A..U,W..Z] —  односимвольные переменные<br>
       ! —  отрицание<br>
       & —  логическое И<br>
       V —  логическое ИЛИ<br><br>
      <input type="button" value="Тип формулы"
             onclick="formulaType()"><br>
    </FORM><br>
  </BODY>
</HTML>
```

Как и в предыдущем случае, функция, осуществляющая анализ формулы, хранится во внешнем файле.

Логическое следствие

При решении многих задач нас интересует, следует ли некоторое утверждение из ранее доказанных или уже известных. Мы введем понятие логического следствия и с его помощью решим некоторые задачи.

Пусть даны формулы F_1, F_2, …, F_n и формула G. Говорят, что G *логическое следствие* формул F_1, F_2, …, F_n (или G следует из F_1, F_2, …, F_n) тогда и только тогда, когда для всякой интерпретации I, в которой истинны утверждения F_1, F_2, …, F_n, утверждение G также истинно. Утверждения F_1, F_2, …, F_n называют *аксиомами* (постулатами или посылками), утверждение G — *следствием*.

Представление утверждений формулами

Предположим, что некоторый человек вас спрашивает: "А верно ли, что если вы любите Бетти, то вы также любите и Джейн?" Вы отвечаете: "Если это верно, то я люблю Бетти". Вопрос звучит так: "Любите ли вы Бетти?"

Запишем утверждения формулами исчисления высказываний, обозначив через B утверждение "Я люблю Бетти", через D — "Я люблю Джейн". Тогда посылка (утверждение "Если это верно, то я люблю Бетти") запишется формулой:

$$((B \Rightarrow D) \Rightarrow B).$$

Предполагаемый ответ (следствие) — формулой B. Переберем возможные значения переменных B и D, вычислим значения формул F и G. Полученные данные оформим в виде табл. 20.4.

Таблица 20.4. *Таблица истинности для формул ((B ⇒ D) ⇒ B) и B*

B	D	F: ((B ⇒ D) ⇒ B)	G: B
И	И	И	И
И	Л	И	И
Л	И	Л	–
Л	Л	Л	–

Нас интересуют только те интерпретации, в которых формула F истинна. В этих интерпретациях истинна и формула G. Согласно определению формула G — логическое следствие формулы F. Таким образом, мы выяснили, что ответ на вопрос "Люблю ли я Бетти?" — положительный.

Задача о хищении

На складе совершено хищение. Подозрение пало на трех человек: *A*, *B* и *C*. Они были доставлены для допроса. Установлено следующее:

- никто, кроме *A*, *B*, *C*, не был замешан в деле;
- подозреваемый *A* никогда не ходит на дело без, по крайней мере, одного соучастника;
- *C* невиновен.

Виновен ли *B*?

Запишем приведенные утверждения с помощью формул исчисления высказываний. На основании определения логического следствия выясним, является ли ответ на вопрос логическим следствием этих трех утверждений. Обозначим через *A* утверждение "*A* виновен", через *B* — "*B* виновен", через *C* — "*C* виновен". Запишем утверждения 1—3 с помощью формул F_1—F_3 исчисления высказываний:

- F_1: $A \vee B \vee C$
- F_2: $A \Rightarrow B \vee C$
- F_3: $\sim C$

Предполагаемый ответ — "*B* виновен" — обозначим через *G*. Проверим, является ли формула *G* логическим следствием формул F_1, F_2, F_3. С помощью табл. 20.5 переберем интерпретации и для всех укажем значение каждой из формул F_1, F_2, F_3 в этой интерпретации.

Таблица 20.5. *Таблица интерпретаций для формул F1, F2, F3*

№	A	B	C	F_1: A ∨ B ∨ C	F_2: A ⇒ B ∨ C	F_3: ~ C	G: B
1	И	И	И	И	И	Л	
2	И	И	Л	И	И	И	И
3	И	Л	И	И	И	Л	
4	И	Л	Л	И	Л	И	
5	Л	И	И	И	И	Л	
6	Л	И	Л	И	И	И	И
7	Л	Л	И	И	И	Л	
8	Л	Л	Л	Л	И	И	

Как и в предыдущем примере, нас интересуют только те интерпретации, в которых все посылки (формулы F_1, F_2, F_3) истинны. Таких интерпретаций

только две — вторая и шестая. В этих интерпретациях и значение формулы G истинно. Следовательно, формула G — логическое следствие, т. е. обвиняемый B виновен.

Упражнение

На этот раз на допрос были вызваны четыре подозреваемых: A, B, C, D. Было доказано, что, по крайней мере, один из них виновен и никто кроме A, B, C, D в ограблении не участвовал. Кроме того, удалось установить следующее.

- A безусловно невиновен;
- если B виновен, то у него ровно один соучастник;
- если C виновен, то у него ровно два соучастника.

Кто же из подозреваемых граждан виновен? Докажите это, воспользовавшись понятием логического следствия.

Теоремы о логическом следствии

Теоремы о логическом следствии позволяют решение вопроса о том, является одно утверждение следствием других, свести к анализу общезначимости или противоречивости некоторой формулы.

Теорема 1

Пусть даны формулы F_1, F_2, ..., F_n и формула G. Тогда формула G является логическим следствием формул F_1, F_2, ..., F_n тогда и только тогда, когда формула

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \Rightarrow G$$

общезначима.

Доказательство. Предположим, что G — логическое следствие. Возьмем произвольную интерпретацию I. Формула $F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n$ может быть либо истинной, либо ложной в этой интерпретации.

Рассмотрим сначала случай, когда эта формула истинна. Тогда истинны и все формулы F_i. Так как формула G — логическое следствие, то она истинна, если истинны посылки. Таким образом, и вся импликация

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \Rightarrow G$$

истинна.

Если же формула $F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n$ ложна в выбранной интерпретации, то, независимо от значения формулы G, рассматриваемая нами формула

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_n \Rightarrow G$$

истинна.

Мы выбрали произвольную интерпретацию и доказали, что исследуемая формула в ней истинна. Таким образом, если G — логическое следствие, то формула

$$F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$$

общезначима (или тавтология).

Предположим теперь, что формула

$$F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$$

общезначима. Докажем, что формула G логическое следствие формул F_1, F_2, ..., F_n. Нас интересуют интерпретации, в которых истинны посылки. Выберем одну из них. В этой интерпретации истинна формула F_1 & F_2 & ... & F_n. Так как формула

$$F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$$

общезначима, то она истинна и в выбранной нами интерпретации. Тогда формула G также должна быть истинной. Теорема доказана.

Теорема 2

Пусть даны формулы F_1, F_2, …, F_n и формула G. Тогда формула G является логическим следствием формул F_1, F_2, …, F_n тогда и только тогда, когда формула

$$F_1 \& F_2 \& \ldots \& F_n \& \sim G$$

противоречива.

Доказательство. Согласно теореме 1 формула G является логическим следствием тогда и только тогда, когда формула

$$F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$$

общезначима. Формула

$$F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$$

является общезначимой тогда и только тогда, когда ее отрицание противоречиво. Выполним эквивалентные преобразования формулы $\sim (F_1 \& F_2 \& \ldots \& F_n \Rightarrow G)$:

$$\sim (F_1 \& F_2 \& \ldots \& F_n \Rightarrow G) \leftrightarrow \sim (\sim (F_1 \& F_2 \& \ldots \& F_n) \vee G) \leftrightarrow$$
$$\leftrightarrow F_1 \& F_2 \& \ldots \& F_n \& \sim G.$$

В результате преобразований получили требуемую формулу, и она противоречива. Следовательно, теорема доказана.

Пример использования теоремы 1 о логическом следствии

Предположим, что вас спрашивают: "А верно ли, что если вы любите Еву, то вы также любите и Маргарет?" А вы отвечаете: "Если это правда, то я люблю Еву, и если я люблю Еву, то это правда". Любите ли вы Маргарет?

Продемонстрируем решение задачи на основании теоремы 1.

Пусть через E обозначено утверждение "Я люблю Еву", через M — "Я люблю Маргарет". Утверждение "Если вы любите Еву, то вы любите Маргарет" запишется формулой: $E \Rightarrow M$. Утверждение: "Если это правда, то вы любите Еву, и если вы любите Еву, то это правда" запишется так:

$$((E \Rightarrow M) \Rightarrow E) \,\&\, (E \Rightarrow (E \Rightarrow M)).$$

Предположим, ответ на вопрос будет таким "Я люблю Маргарет". Тогда по теореме 1 формула

$$((E \Rightarrow M) \Rightarrow E) \,\&\, (E \Rightarrow (E \Rightarrow M)) \Rightarrow M$$

должна быть общезначимой. Проверим это, выполняя эквивалентные преобразования формул:

$$((E \Rightarrow M) \Rightarrow E) \,\&\, (E \Rightarrow (E \Rightarrow M)) \Rightarrow M \leftrightarrow$$

$$\sim (((E \Rightarrow M) \Rightarrow E) \,\&\, (E \Rightarrow (E \Rightarrow M))) \vee M \leftrightarrow$$

$$\sim ((E \Rightarrow M) \Rightarrow E) \vee \sim (E \Rightarrow (E \Rightarrow M)) \vee M \leftrightarrow$$

$$(E \Rightarrow M) \,\&\, \sim E \vee \sim (\sim E \vee \sim E \vee M) \vee M \leftrightarrow$$

$$(\sim E \vee M) \,\&\, \sim E \vee (E \,\&\, \sim M) \vee M \leftrightarrow$$

$$\sim E \,\&\, (M \vee \sim E) \vee (E \vee M) \,\&\, (\sim M \vee M) \leftrightarrow$$

$$\sim E \,\&\, (M \vee \sim E) \vee (E \vee M) \,\&\, \blacksquare \leftrightarrow$$

$$\sim E \,\&\, (M \vee \sim E) \vee (E \vee M) \leftrightarrow$$

$$(\sim E \vee E \vee M) \& (M \vee \sim E \vee E \vee M) \leftrightarrow \blacksquare \,\&\, \blacksquare \leftrightarrow \blacksquare.$$

Последняя формула общезначима, поэтому и рассматриваемая формула

$$((E \Rightarrow M) \Rightarrow E) \,\&\, (E \Rightarrow (E \Rightarrow M)) \Rightarrow M$$

также общезначима, и мы нашли ответ на вопрос задачи.

Упражнение

Расследуется простое дело о хищении со склада. Преступник (или преступники) вывезли награбленное имущество на машине. Подозрение пало на трех человек A, B, C, которые были доставлены в Скотланд-Ярд для допроса. Было установлено следующее:

- никто, кроме A, B, C, в хищении не замешан;
- C не ходит на дело без A;
- B не умеет водить машину.

Виновен или не виновен A?

Задание по программированию

1. Напишите сценарий, который определяет, является ли общезначимой некоторая заданная формула.
2. Напишите сценарий, который определяет, является ли некоторая формула следствием заданных пользователем формул, опираясь на теорему 1 о логическом следствии.

Пример использования теоремы 2 о логическом следствии

Рассмотрим предыдущий пример и ответим на вопрос: "Люблю ли я Еву?" На основании теоремы 2 о логическом следствии нас интересует, является ли противоречивой формула:

$$((E \Rightarrow M) \Rightarrow E) \ \& \ (E \Rightarrow (E \Rightarrow M)) \ \& \sim E.$$

Убедимся в этом, выполняя эквивалентные преобразования формул:

$$((E \Rightarrow M) \Rightarrow E) \ \& \ (E \Rightarrow (E \Rightarrow M)) \ \& \sim M \leftrightarrow$$

$$(\sim (E \Rightarrow M) \vee E) \ \& \ (\sim E \vee \ (E \Rightarrow M)) \ \& \sim M \leftrightarrow$$

$$(E \ \& \sim M \vee E) \ \& \ (\sim E \vee M) \ \& \sim E \leftrightarrow$$

$$E \ \& \ (\sim M \vee E) \ \& \ (\sim E \vee M) \ \& \sim E \leftrightarrow \ \square.$$

Последняя формула противоречива, следовательно, справедливо утверждение "Я люблю Еву".

Задача о рыцарях и лжецах

На некотором острове, населенном рыцарями и лжецами (напомним, что рыцари говорят только правду, лжецы — ложь), разнесся слух о том, что на нем зарыты сокровища. Вы прибываете на остров и спрашиваете одного из местных жителей (назовем его A), есть ли на острове золото. В ответ на ваш вопрос A заявляет: "Сокровища на этом острове есть в том и только том случае, если я рыцарь". Можно ли определить:

- кто такой A — рыцарь или лжец;
- есть ли сокровища на острове?

Ответим на второй вопрос. Обозначим через A утверждение "A — рыцарь", через L — "Сокровища на острове есть". Тогда высказывание жителя A "Сокровища на острове есть в том и только том случае, если я рыцарь" запишется формулой:

$$(A \Leftrightarrow L).$$

Если A — рыцарь, то его высказывание истинно; если лжец, то ложно, таким образом, наша посылка — формула вида

$$(A \Leftrightarrow (A \Leftrightarrow L)).$$

Для того чтобы доказать, что утверждение L — следствие нашей посылки, т. е. сокровища на острове есть, нам достаточно на основании теоремы 2 доказать, что формула $(A \Leftrightarrow (A \Leftrightarrow L)) \ \& \sim L$ противоречива. Проверим это, перебрав все интерпретации и вычислив значение формул в каждой из них. В табл. 20.6 приведены значения формул во всех интерпретациях.

Таблица 20.6. *Таблица истинности для формулы* $(A \Leftrightarrow (A \Leftrightarrow L)) \ \& \sim L$

A	L	(A ⇔ L)	(A ⇔ (A ⇔ L))	(A ⇔ (A ⇔ L)) & ~ L
И	И	И	И	Л
И	Л	Л	Л	Л
Л	И	Л	И	Л
Л	Л	И	Л	Л

Формула

$$(A \Leftrightarrow (A \Leftrightarrow L)) \ \& \sim L$$

ложна в любой интерпретации, следовательно, она противоречива, и поэтому верно утверждение, что сокровища на острове есть.

Упражнения

По обвинению в ограблении перед судом предстали A, B и C. Установлено следующее:

- по крайней мере, один из трех подсудимых виновен;
- если A виновен в ограблении и B невиновен, то C невиновен.

Этих данных недостаточно, чтобы доказать виновность каждого из трех подсудимых в отдельности, но эти данные позволяют отобрать двух подсудимых, о которых известно, что один из них заведомо виновен. Определите и докажите, о каких подсудимых идет речь.

Мы видим, что если задача такова, что ее посылки (постулаты) и следствие записываются формулами исчисления высказываний, то решить задачу можно одним из следующих способов:

1. На основании определения логического следствия рассмотреть все интерпретации, в которых истинна формула $F_1 \ \& \ F_2 \ \& \ \ldots \ \& \ F_n$, и проверить значение формулы G.

2. На основании теоремы 1 проверить, является ли общезначимой формула $F_1 \& F_2 \& \ldots \& F_n \Rightarrow G$.
3. На основании теоремы 2 проверить, является ли противоречивой формула $F_1 \& F_2 \ \ldots \& F_n \ \& \sim G$.

Конъюнктивная нормальная форма и ее построение

Во многих случаях удобно рассматривать формулы специального вида или, как говорят, формулы в канонической форме. В математической логике и в теории автоматического доказательства теорем рассматриваются формулы в конъюнктивной нормальной форме.

Литералом называется атом или отрицание атома.

Дизъюнктом называется литерал или дизъюнкция литералов.

Формула находится в *конъюнктивной нормальной форме (КНФ)*, если она представляет собой либо дизъюнкт, либо является конъюнкцией дизъюнктов.

Элементарной конъюнкцией называется конъюнкция литералов.

Дизъюнктивной нормальной формой называется дизъюнкция элементарных конъюнкций.

Теорема

По любой формуле F может быть построена эквивалентная ей формула G, находящаяся в КНФ.

Доказательство. Идею доказательства теоремы и построения КНФ поясним на примере формулы F от трех переменных. Возьмем следующую формулу F:

$$(X \Rightarrow Y) \Leftrightarrow Z.$$

Приведем табл. 20.7, в которой перечислим интерпретации и значение формулы в каждой из интерпретаций. Правый столбец таблицы будем использовать для построения дизъюнктов конъюнктивной нормальной формы, соответствующей формуле.

Таблица 20.7. *Построение конъюнктивной нормальной формы*

№	X	Y	Z	$(X \Rightarrow Y) \Leftrightarrow Z$	Дизъюнкты формулы
1	И	И	И	И	
2	И	И	Л	Л	$\sim X \vee \sim Y \vee Z$
3	И	Л	И	Л	$\sim X \vee Y \vee \sim Z$

Таблица 20.7 *(окончание)*

№	X	Y	Z	$(X \Rightarrow Y) \Leftrightarrow Z$	Дизъюнкты формулы
4	И	Л	Л	И	
5	Л	И	И	И	
6	Л	И	Л	Л	$X \vee \sim Y \vee Z$
7	Л	Л	И	И	
8	Л	Л	Л	Л	$X \vee Y \vee Z$

Выделим те интерпретации, в которых формула принимает значение "ложь", и для этих интерпретаций построим дизъюнкт по правилу: если значение переменной — "ложь", то включаем в дизъюнкт эту переменную, в противном случае включаем переменную с отрицанием. Тогда по построению в интерпретациях, в которых значение F ложно, значение G также ложно. В нашем случае формула G является конъюнкцией построенных формул:

$$(\sim X \vee \sim Y \vee Z) \ \& \ (\sim X \vee Y \vee \sim Z) \ \& \ (X \vee \sim Y \vee Z) \ \& \ (X \vee Y \vee Z).$$

Формулы F и G эквивалентны и G имеет КНФ. Эквивалентность формул можно доказать в качестве упражнения.

Упражнение

Преобразуйте следующие формулы в конъюнктивную нормальную форму:

- $(A \Leftrightarrow (A \Leftrightarrow S)) \ \& \ S$
- $((E \Rightarrow M) \Rightarrow E) \ \& \ (E \Rightarrow (E \Rightarrow M))$

Построение формулы в конъюнктивной нормальной форме

Напишем сценарий, который на основании рассмотренной теоремы по произвольной формуле строит эквивалентную формулу, находящуюся в КНФ.

В текстовое поле вводится формула, после щелчка по кнопке, формируется эквивалентная формула, находящаяся в КНФ. Один из тестов продемонстрирован на рис 20.6.

При построении КНФ перебираются все интерпретации, исследуется значение формулы в текущей интерпретации, и, если значение ложно, то соответствующая формула добавляется в строку результата. Вместе с формулой в конъюнктивной нормальной форме выдается таблица истинности так, как показано на рис. 20.7.

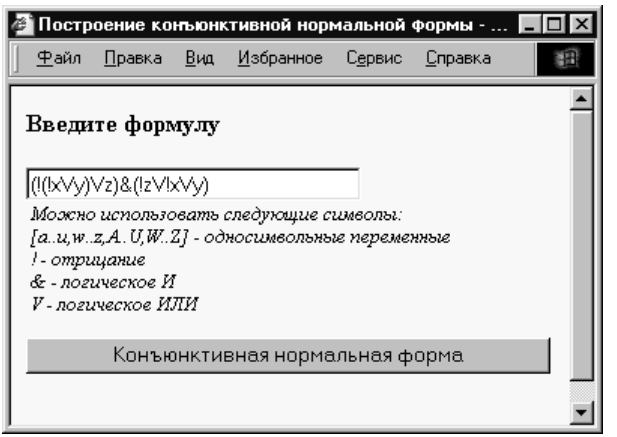


Рис. 20.6. Построение конъюнктивной нормальной формы

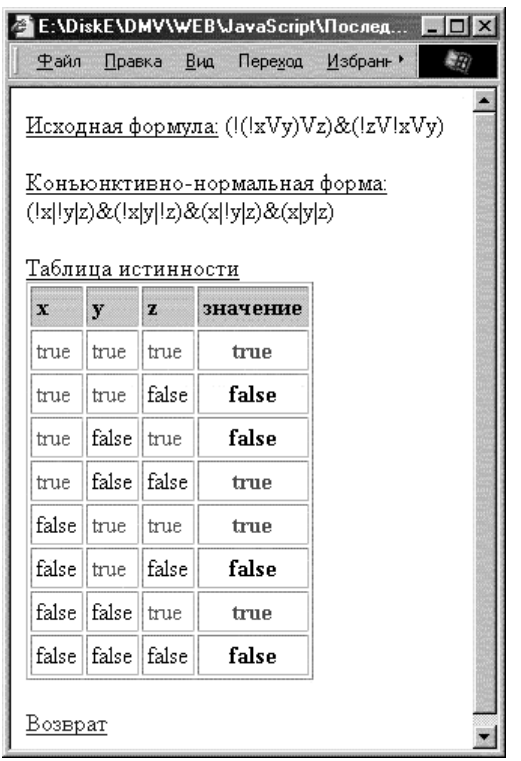


Рис. 20.7. Конъюнктивная нормальная форма заданной пользователем формулы

Три функции (построение таблицы истинности, анализ типа формулы и построение КНФ) хранятся во внешнем файле. Приведем содержимое HTML-документа, содержащего требуемые функции (листинг 20.4).

Листинг 20.4. Конъюнктивная нормальная форма

```
// Функции построения таблицы истинности и КНФ. Korostelyov Andrey
var srcStr = "";                // Исходная строка
var let = new Array();          // Массив значений переменных формулы
var letName = new Array();      // Массив имен переменных
var resStr = "";                // Строка результата
var TABLEStr = "";              // Строка формирования элементов таблицы
var knfStr = null;              // Формула в КНФ
var trueVal = 0;                // Формула общезначима == TRUE
var falseVal = 0;               // Формула противоречива == FALSE
// Формирование заголовка документа и таблицы
function init()
 { srcStr=document.data.formula.value;
   parseLine(srcStr);
   document.writeln("<u>Исходная формула:</u> "+srcStr+"<br><br>");
   TABLEStr += "<u>Таблица истинности</u>\n<TABLE border=1
cellpadding=3>\n"+
   "<TR bgcolor=#cccccc>\n";
   for(i=0; i<letName.length; i++)
    { TABLEStr += "\t<TD><b>"+letName[i]+"</b></TD>\n"}
   TABLEStr += "<TD align=center><b>значение</b></TD>\n</TR>\n";
   prepareTABLE(let, 0);
   TABLEStr += "</TABLE>";
 }
// Формирование строки таблицы
function parseLine(str)
 { var cur;
   var k=0;
   for(i=0; i<str.length; i++){
     cur = str.charAt(i);
     if(cur=="="||cur=="!"||cur=="("||cur==")")
      { resStr+=cur}
     else if (cur=="&")
      { resStr+="&&"}
     else if (cur=="v"||cur=="V")
```

```
      { resStr+="||"}
     else
      { var elemIndex = indexOfElement(letName,cur);
        if(elemIndex==-1)
         { elemIndex = k
           let[k] = true;
           letName[k]=cur;
           k++;
         }
        resStr+="lettr["+elemIndex+"]";
      }
   }
 }
// Возврат индекса найденного элемента или -1, если элемента нет
function indexOfElement(arr, element)
 { for(var l=0; l<arr.length; l++)
    { if (arr[l]==element )
      { return l }
    }
   return -1;
 }
// Формирование строки таблицы значений
function prepareTABLE(lettr, pos)
 { var resStrVal;
   var tmpLettr;
  // pos — номер переменной
  //j==0 — true
  //j==1 — false
  for(var j=0; j<2; j++)
   { if((pos!=0 && j!=0)||(pos==0))
      { TABLEStr += "<TR>\n";
        for(i=0; i<lettr.length; i++)
         { if (lettr[i]!=false){ tmpLettr="<font
color=#555555>"+lettr[i]+"</font>" }
           else {tmpLettr=lettr[i]}
           TABLEStr += "\t<TD>"+tmpLettr+"</TD>\n";
         }
        resStrVal = eval(resStr);
        if (resStrVal!=false)
```

```
         { resStrVal="<font color=#555555>"+resStrVal+"</font>" }
        TABLEStr += "\t<TD align=center><b>"+resStrVal+"</b></TD>\n";
        TABLEStr += "</TR>\n";
        if(resStrVal){ trueVal++; }
        else {
          falseVal++;
          if (knfStr != null){prepareKNF(lettr)}
      }
   }
   if(pos<lettr.length-1)
   { prepareTABLE(lettr,pos+1)}
   lettr[pos]=false;
 }
    lettr[pos]=true;
}
// Построение конъюнктивной нормальной формы
function prepareKNF(lettr)
 { if (knfStr != ""){ knfStr += "&"; }
   knfStr += "(";
   for(i=0; i<lettr.length; i++)
    { if (lettr[i])
       { knfStr += "!"+ letName[i] + "|"; }
      else
       { knfStr += letName[i] + "|"; }
    }
   knfStr = knfStr.substring(0, knfStr.length-1)+")";
 }
// Формирование ссылки перехода к основному документу
function docFooter()
 { document.writeln("<br><A href=index.HTML>Возврат</A>")}
// Вывод таблицы
function TABLE()
 { init();
   document.writeln(TABLEStr);
   docFooter();
 }
// КНФ
function knf()
```

```
 { knfStr = "";
   init();
   if (knfStr==""){knfStr = "Отсутствует"}
   knfStr = "<u>Конъюнктивно-нормальная форма:</u> <BR>\n" +
            knfStr+"<br>";
   document.writeln(knfStr+"<br>");
   document.writeln(TABLEStr);
   docFooter();
 }
// Определение типа формулы
function formulaType()
 { init();
   document.writeln("<u>Формула является:</u><br>\n");
   if (trueVal==0)
    { document.writeln("Противоречивой<br><br>") }
   else if (falseVal==0)
    { document.writeln("Общезначимой<br><br>")}
   else
    { document.writeln("Выполнимой<br><br>");
      document.writeln(TABLEStr);
    }
   docFooter();
 }
```

Алгоритм построения КНФ

Один из способов построения формулы в КНФ мы уже продемонстрировали. Заметим, что КНФ G формулы F можно получить, выполняя последовательно эквивалентные преобразования формулы F. Алгоритм преобразования может быть таким:

1. Сначала следует исключить из формулы знаки логической эквивалентности и импликации, применяя последовательно правила эквивалентности:
   - $A \Leftrightarrow B \leftrightarrow (\sim A \vee B) \ \& \ (\sim B \vee A)$
   - $(A \Rightarrow B) \leftrightarrow \sim A \vee B$
2. Затем необходимо подтянуть логическое отрицание к атомам, используя правило $\sim (\sim F) \leftrightarrow F$, и применяя правила де Моргана:
   - $\sim (A \vee B) \leftrightarrow \sim A \ \& \sim B$
   - $\sim (A \ \& \ B) \leftrightarrow \sim A \vee \sim B$

3. Потом применить несколько раз дистрибутивные законы:
   - $A \vee (B \,\&\, C) \leftrightarrow (A \,\&\, B) \vee (A \,\&\, C)$
   - $A \,\&\, (B \vee C) \leftrightarrow (A \,\&\, B) \vee (A \,\&\, C)$

Продемонстрируем в качестве примера те преобразования, которые необходимо выполнить, чтобы построить КНФ для рассмотренной нами ранее формулы $(X \Rightarrow Y) \Leftrightarrow Z$:

- ❒ $(X \Rightarrow Y) \Leftrightarrow Z$
- ❒ $((X \Rightarrow Y) \Rightarrow Z) \,\&\, (Z \Rightarrow (X \Rightarrow Y))$
- ❒ $(\sim (X \Rightarrow Y) \vee Z) \,\&\, (\sim Z \vee \sim X \vee Y)$
- ❒ $(\sim (\sim X \vee Y) \vee Z) \,\&\, (\sim Z \vee \sim X \vee Y)$
- ❒ $(X \,\&\, \sim Y \vee Z) \,\&\, (\sim Z \vee \sim X \vee Y) \,\&\, (\sim Z \vee \sim X \vee Y)$
- ❒ $(X \vee Z) \;\&\; (\sim Y \vee Z) \;\&\; (\sim X \vee Y \vee \sim Z)$

Полученная формула G_1

$$(X \vee Z) \,\&\, (\sim Y \vee Z) \,\&\, (\sim X \vee Y \vee \sim Z)$$

эквивалентна исходной формуле F, находится в конъюнктивной нормальной форме и не совпадает с G.

Упражнения

1. Преобразуйте следующие формулы в конъюнктивную нормальную форму:
   - $\sim (P \,\&\, Q) \Rightarrow (R \Rightarrow T)$
   - $((E \Rightarrow M) \Rightarrow \sim E) \vee (E \Rightarrow (E \Rightarrow \sim M))$
   - $(P \Rightarrow (R \Rightarrow T)) \;\Leftrightarrow \sim (P \,\&\, Q)$
2. Преобразуйте следующие формулы в дизъюнктивную нормальную форму:
   - $\sim (P \,\&\, Q) \Rightarrow (R \Rightarrow T) \,\&\, \sim R$
   - $E \,\&\, ((E \Rightarrow M) \Rightarrow \sim E) \vee (E \Rightarrow (E \Rightarrow \sim M))$
   - $\sim P \Rightarrow (P \Rightarrow (R \Rightarrow T)) \;\Leftrightarrow \sim (P \,\&\, Q)$

Нетрудно доказать, что произвольная пропозициональная формула эквивалентна дизъюнкции элементарных конъюнкций, соответствующих тем интерпретациям, при которых данная формула истинна.

Автоматизация ввода логических формул

Необходимо создать сценарий, который облегчает ввод логических формул. При нажатии кнопок, сопоставленных переменным, скобкам или логическим операциям, соответствующая операция появляется в поле ввода. После

формирования формулы в текстовом поле при щелчке по кнопке вычисляется таблица истинности, определяется тип формулы, строится соответствующая формуле конъюнктивная или дизъюнктивная нормальные формы так, как показано на рис. 20.8.

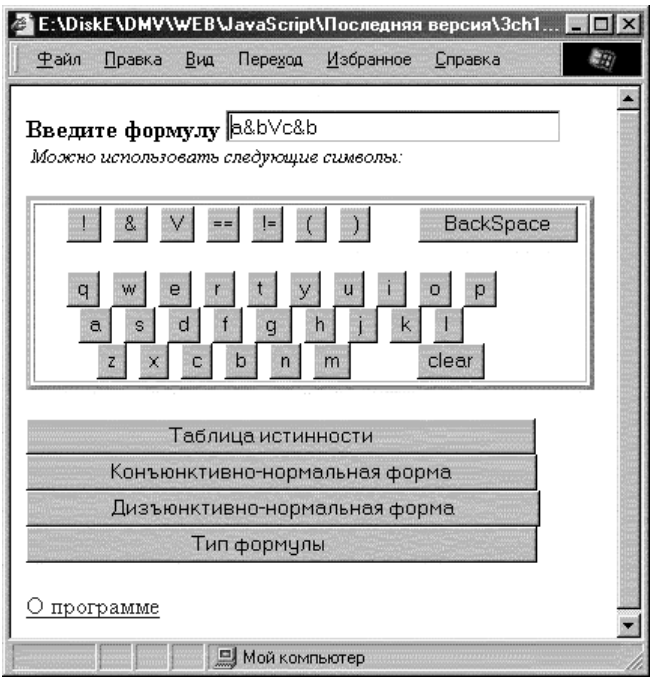


Рис. 20.8. Обработка логических формул

Полностью написать сценарий с помощью уже описанных функций предлагается в качестве упражнения.

В последних рассмотренных примерах использовались три логические операции. Это облегчало написание алгоритмов анализа формулы, т. к. позволяло для вычисления значения формулы воспользоваться функцией `eval`. Рассмотреть варианты решения предложенных задач в случае, когда в формуле используются импликация и эквивалентность, читателю предлагается самостоятельно.

Метод резолюций

Теперь нам надо показать, как из уже имеющихся знаний можно извлекать новые утверждения. Введем *правило резолюции*, согласно которому из двух рассуждений можно вывести третье.

Литерал L_1 образует *контрарную пару* с литералом L_2, если

$$L_1 = \sim L_2.$$

Пусть C_1 и C_2 — два дизъюнкта. Литерал L_1 из дизъюнкта C_1, литерал L_2 из дизъюнкта C_2. Причем литералы L_1 и L_2 образуют контрарную пару. *Резольвентой дизъюнктов* C_1 и C_2 называется дизъюнкт C, составленный из C_1 вычеркиванием L_1 и из C_2 вычеркиванием L_2.

Таким образом, если у нас есть два дизъюнкта и один из них содержит литерал L, а другой — литерал $\sim L$, то мы можем рассмотреть дизъюнкт, который содержит литералы первого за исключением литерала L и литералы второго за исключением $\sim L$. Такой дизъюнкт называется резольвентой первых двух. Например, если первый дизъюнкт $\sim A \vee C$, второй $\sim C \vee D$, то их резольвентой будет формула $\sim A \vee D$. Если первый дизъюнкт состоит из одного литерала A, а второй из литерала $\sim A$, то их резольвентой будет тождественно ложный дизъюнкт, который принято обозначать символом $\square$.

Выводом в методе резолюций дизъюнкта C из множества S называется последовательность дизъюнктов $C_1, C_2, \ldots, C_k$ такая, что для любого i $(i \le k)$ C_i — либо дизъюнкт из S, либо является резольвентой предыдущих дизъюнктов.

Вывод можно рассматривать как формализованный аналог понятия доказательства.

Вывод тождественно ложного дизъюнкта называется *опровержением*.

Теорема о резольвенте

Пусть даны два дизъюнкта C_1 и C_2. Резольвента C дизъюнктов C_1 и C_2 является их логическим следствием.

Резольвента позволяет переходить от двух рассуждений к третьему, а теорема о резольвенте гарантирует, что если исходные утверждения были истинными, то истинным будет и построенное утверждение, т. е. резольвента.

Доказательство. Пусть

$$C_1 = L \vee P \text{ и } C_2 = \sim L \vee Q,$$

тогда резольвента имеет вид $P \vee Q$. Предположим, что дизъюнкты C_1 и C_2 истинны в произвольно взятой интерпретации I. Нам требуется доказать, что резольвента C также истинна в этой интерпретации. Если значение литерала L в выбранной интерпретации истинно, то значение литерала $\sim L$ ложно, но тогда значение дизъюнкта Q должно быть истинным, т. к. истинен дизъюнкт C_2. А поскольку значение дизъюнкта Q истинно, то и резольвента истинна. Если же значение литерала L в выбранной интерпретации ложно, то истинным должно быть значение дизъюнкта P, т. к. дизъюнкт C_1 истинен. Как и в предыдущем случае, резольвента будет иметь значение "истина". Таким образом, резольвента является логическим следствием дизъюнктов, по которым она построена.

Пример использования теоремы о резольвенте

Предположим, что справедливы следующие утверждения:

- если электрички приходят не по расписанию, то студенты опаздывают на лекцию;
- если студенты опаздывают на лекцию, то преподаватель недоволен.

Введем следующие обозначения: обозначим через P утверждение "Электрички приходят не по расписанию", через Q — "Студенты опаздывают на лекцию" и, наконец, через R — "Преподаватель недоволен". Тогда приведенные рассуждения записываются формулами:

$$(P \Rightarrow Q) \text{ и } (Q \Rightarrow R).$$

После преобразования к КНФ получим дизъюнкты

$$(\sim P \vee Q) \text{ и } (\sim Q \vee R),$$

содержащие контрарную пару. Резольвентой этих двух дизъюнктов будет формула $(\sim P \vee R)$, которая эквивалентна формуле $(P \Rightarrow R)$. Последняя формула, очевидно, соответствует рассуждению: "Если электрички не приходят по расписанию, то преподаватель недоволен".

Упражнения

Найдите все возможные резольвенты (если они есть) следующих пар дизъюнктов:

- $\sim A \vee C \vee R \vee T \vee B$
- $\sim B \vee C \vee \sim M$
- $\sim A \vee B \vee C \vee \sim N$
- $\sim C \vee D \vee \sim R$
- $A \vee D \vee C \vee \sim Q$
- $\sim D$

Задание по программированию

Напишите сценарий, который по двум заданным дизъюнктам строит резольвенту, если это возможно.

Теорема о полноте метода резолюций

Множество дизъюнктов S противоречиво тогда и только тогда, когда существует опровержение в S (или из S).

Доказательство. Приведем доказательство достаточности. Предположим, что мы построили опровержение во множестве дизъюнктов S; вывод имеет вид: D_1, D_2, ..., D_n. Докажем, что множество дизъюнктов S противоречиво. Предположим, что это не так, множество S выполнимо и, следовательно, существует интерпретация I, в которой все дизъюнкты из S истинны. Каждое D_i в опровержении либо из множества S и является истинным, либо представляет из себя логическое следствие предыдущих дизъюнктов и также истинно по теореме о резольвенте. Но тогда и последний дизъюнкт истинен, чего не может быть, потому что последний дизъюнкт в опровержении тождественно ложная формула.

Вернемся к сформулированной ранее задаче. Пусть у нас есть аксиомы или уже доказанные факты F_1, F_2, ..., F_m. Нам требуется определить, следует ли утверждение G из фактов F_1, F_2, ..., F_m. По теореме 2 о логическом следствии для решения этого вопроса нам достаточно определить, является ли формула

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_m \,\&\, \sim G$$

противоречивой. Эту формулу можно преобразовать к КНФ и рассматривать в дальнейшем множество дизъюнктов S:

$$S = \{S_1, S_2, \ldots, S_n\}.$$

По теореме о полноте метода резолюций, для того чтобы определить противоречивость (невыполнимость) множества S, достаточно найти опровержение в S.

Задача о поиске виновного

В совершении преступления подозреваются четверо человек: A, B, C, D. Установлено следующее:

- если A или B виновны, то и подозреваемый C виновен;
- если A виновен, то, по крайней мере, один из двух B или C тоже виновны;
- если C виновен в ограблении, то и D виновен;
- если A невиновен, то D виновен.

Виновен ли D?

Будем считать, как и раньше, что через A обозначено утверждение "A виновен", через B — "B виновен", через C — "C виновен" и, наконец, через D — "D виновен". Запишем четыре известных факта и ответ на вопрос формулами исчисления высказываний:

- F_1: $A \vee B \Rightarrow C$
- F_2: $A \Rightarrow B \vee C$
- F_3: $C \Rightarrow D$
- F_4: $\sim A \Rightarrow D$

Следствие G: D (D виновен).

По теореме 2 о логическом следствии построим для наших конкретных высказываний формулу вида $F_1 \& F_2 \& F_3 \& F_4 \& \sim G$:

$$(A \vee B \Rightarrow C) \& (A \Rightarrow B \vee C) \& (C \Rightarrow D) \& (\sim A \Rightarrow D) \& \sim D.$$

Приведем ее к КНФ, получим множество дизъюнктов, которые обозначим S_1—S_6:

$$S = \{\sim A \vee C, \sim B \vee C, \sim A \vee B \vee C, \sim C \vee D, A \vee D, \sim D\}.$$

Построим опровержение в S:

- S_7: $\sim A \vee D$ резольвента S_1 и S_4
- S_8: D резольвента S_7 и S_5
- S_9: $\square$ резольвента S_8 и S_6

Построив опровержение в методе резолюций, мы по теореме о полноте метода резолюций доказали, что множество дизъюнктов противоречиво, следовательно, рассматриваемая формула противоречива. По теореме 2 утверждение G — логическое следствие, т. е. подозреваемый в преступлении D виновен.

Задача о влюбленном логике

Перед нами три девушки: Сью, Марция и Диана. Предположим, что юноша, занимающийся математической логикой, высказывает следующее.

- Я люблю, по крайней мере, одну из этих трех девушек.
- Если я люблю Сью, а не Диану, то я также люблю Марцию.
- Я либо люблю Диану и Марцию, либо не люблю ни одну из них.
- Если я люблю Диану, то я также люблю Сью.

Требуется определить, любит ли автор высказываний Диану?

Запишем формулами четыре высказывания и предполагаемое следствие:

- F_1: $C \vee M \vee D$
- F_2: $C \& \sim D \Rightarrow M$
- F_3: $D \& M \vee \sim D \& \sim M$
- F_4: $D \Rightarrow C$
- G: D

Формулу, построенную согласно теореме 2 о логическом следствии:

$$(C \vee M \vee D) \& (C \& \sim D \Rightarrow M) \& (D \& M \vee \sim D \& \sim M) \& (D \Rightarrow C) \& \sim D$$

приведем к КНФ и получим интересующее нас множество дизъюнктов:

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, \ M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

Считаем, что наши исходные дизъюнкты пронумерованы от S_1 до S_6. Построим опровержение в S:

- S_7: $M \vee D$ резольвента S_1 и S_2
- S_8: D резольвента S_7 и S_5
- S_9: $\square$ резольвента S_8 и S_6

На основании теоремы о полноте метода резолюций делаем вывод о том, что автор высказываний любит Диану.

Упражнения

1. Постройте опровержение для решения следующей задачи. На острове рыцарей и лжецов за преступление судили двух жителей. Обвинитель, однако, оказался также жителем острова. Он заявил следующее.
   - X и Y не могут быть виновны оба;
   - X виновен.

   Кто обвинитель: рыцарь или лжец?
2. Постройте опровержение для решения следующей задачи. Три человека, A, B и C, проживают на острове рыцарей и лжецов. Условимся называть двух людей однотипными, если они оба рыцаря или оба лжеца. Пусть A и B высказывают следующие утверждения:
   - A: B — лжец;
   - B: A и C однотипны.

   Кто такой C: рыцарь или лжец?

Задание по программированию

1. Напишите сценарий, который по заданной последовательности дизъюнктов определяет, может ли она быть выводом в данном множестве дизъюнктов.
2. Напишите сценарий построения опровержения для заданного множества дизъюнктов.

Методы сокращения множества дизъюнктов

Мы исследуем вопрос, является ли противоречивым множество дизъюнктов S. Однако это множество может быть достаточно велико, поэтому искать вывод из большого множества дизъюнктов иногда затруднительно. Возникает вопрос, можно ли как-нибудь это множество сократить? Математики Девис

и Патнем предложили несколько правил, которые позволяют сократить исходное множество дизъюнктов.

Правило тавтологии

Из множества дизъюнктов S можно вычеркнуть все дизъюнкты, которые являются тавтологиями. Оставшееся множество дизъюнктов противоречиво тогда и только тогда, когда противоречиво множество S.

Правило однолитерных дизъюнктов

Если во множестве S существует однолитерный дизъюнкт L, то множество S_1 получается из S вычеркиванием всех дизъюнктов, содержащих L. Если построенное S_1 пусто, то исходное S выполнимо. Для доказательства выполнимости S достаточно рассмотреть интерпретацию, в которой значение L истинно. В противном случае по S_1 строим множество дизъюнктов S_2, вычеркивая из S_1 все вхождения $\sim L$. Множество дизъюнктов S_2 противоречиво тогда и только тогда, когда противоречиво множество S.

Задача о влюбленном логике с применением правила однолитерного дизъюнкта

Мы рассмотрели ранее задачу о влюбленном логике и трех девушках: Сью, Марции и Диане. Для решения вопроса о том, любит ли человек, высказывающий четыре утверждения, Диану, мы строили опровержение в методе резолюций. Однако решить эту задачу можно, применяя к построенному множеству дизъюнктов правило однолитерного дизъюнкта.

Рассмотрим множество дизъюнктов S:

$$S = \{C \vee M \vee D, \sim C \vee D \vee M, D \vee \sim M, \sim D \vee M, \sim D \vee C, \sim D\}.$$

Дизъюнкт $\sim D$ — однолитерный; применив к множеству S правило однолитерного дизъюнкта, получим множество S_1:

$$S_1 = \{C \vee M \vee D, \sim C \vee D \vee M, D \vee \sim M\}.$$

Теперь вычеркиванием из S_1 всех вхождений D получим множество S_2:

$$S_2 = \{C \vee M, \sim C \vee M, \sim M\}.$$

В построенном множестве также есть единичный дизъюнкт $\sim M$. После применения правила единичного дизъюнкта к этому множеству получаем $\{C, \sim C\}$. В последнем множестве опять есть единичный дизъюнкт, после применения правила получаем противоречивое множество, следовательно, и исходное множество S противоречиво, т. е. D — логическое следствие.

Правило чистых литералов

Литерал L называется *чистым литералом*, если в S не входит литерал $\sim L$.

Если L — чистый литерал, то вычеркиваем из S все дизъюнкты, содержащие L. Оставшееся множество дизъюнктов обозначим через S_1. Множество дизъюнктов S_1 противоречиво тогда и только тогда, когда противоречиво множество S.

Правило расщепления

Пусть множество S можно представить в виде:

$S = \{A_1 \vee L, \ldots, A_m \vee L, B_1 \vee \sim L, \ldots, B_k \vee \sim L, R_1, \ldots, R_q\}$, причем дизъюнкты A_i, B_i, R_i не содержат ни L, ни $\sim L$. Рассмотрим два множества:

- $S_1 = \{A_1, A_2, \ldots, A_m, R_1, .., R_q\}$
- $S_2 = \{B_1, B_2, \ldots, B_k, R_1, \ldots, R_q\}$

Множество S противоречиво тогда и только тогда, когда оба множества S_1 и S_2 противоречивы.

Пример использования правила чистого литерала

Рассмотрим множество дизъюнктов S:

$$S = \{A \vee B \vee C, A \vee B \vee \sim C, \sim B, \sim C\}.$$

В этом множестве есть чистый литерал A. Применим к множеству дизъюнктов правило чистого литерала. Множество после применения правила чистого литерала будет следующим:

$$S_1 = \{\sim B, \sim C\}.$$

Пример использования правила расщепления

Рассмотрим множество дизъюнктов S и применим к нему правило расщепления по литералу A:

$$S = \{\sim A \vee B, \sim A \vee \sim B \vee C, A \vee D, D \vee \sim C, A \vee \sim D, \sim D\}.$$

Полученные после применения правила расщепления множества:

$$S_1 = \{B, \sim B \vee C, D \vee \sim C, \sim D\}, \qquad S_2 = \{D, \sim D, D \vee \sim C\}.$$

Множество S_1 противоречиво, доказать это можно, построив, например, опровержение в S_1:

- S_5: C резольвента дизъюнктов B и $\sim B \vee C$
- S_6: D резольвента дизъюнктов C и $D \vee \sim C$
- S_7: $\square$ резольвента дизъюнктов D и $\sim D$

Множество S_2 противоречиво, т. к. опровержение строится по дизъюнктам D и $\sim D$.

В некоторых случаях удается решить задачу, используя лишь правила сокращения дизъюнктов. Продемонстрируем такую возможность при решении следующей задачи.

Задача об искателе приключений

Предположим, что некоторый искатель приключений хочет добраться до острова сокровищ. Путешественнику предложили три карты: X, Y и Z. Только одна из карт правильная и указывает путь к острову сокровищ. В комнате, где лежали карты, находились пятеро колдунов: A, B, C, D и E. Каждый колдун был либо рыцарем, либо лжецом и каждый дал путешественнику совет:

- A: X — правильная карта;
- B: Y — правильная карта;
- C: неверно, что A и B — оба лжеца;
- D: либо A — лжец, либо B — рыцарь;
- E: либо A — лжец, либо C и D — однотипны (либо оба рыцари, либо оба лжецы).

Какая же из карт правильная?

Докажем, что Y — правильная карта. Будем, как ранее, обозначать через A утверждение "A — рыцарь" (аналогично для B, C, D и E). Обозначим через X утверждение "X — правильная карта" (аналогично для Y и Z). Запишем сначала советы колдунов и предполагаемое следствие формулами исчисления высказываний:

- F_1: $A \Leftrightarrow X$
- F_2: $B \Leftrightarrow Y$
- F_3: $C \Leftrightarrow \sim (\sim A \,\&\, \sim B)$
- F_4: $D \Leftrightarrow \sim A \vee B$
- F_5: $E \Leftrightarrow \sim E \vee (C \Leftrightarrow D)$
- G: Y

Построим КНФ вида

$$F_1 \,\&\, F_2 \,\&\, F_3 \,\&\, F_4 \,\&\, F_5 \,\&\, \sim G.$$

Для этого приведем каждое из F_i к КНФ:

- F_1: $A \Leftrightarrow X \leftrightarrow (\sim A \vee X) \,\&\, (\mathrm{A} \vee \sim X)$
- F_2: $B \Leftrightarrow Y \leftrightarrow (\sim B \vee Y) \,\&\, (B \vee \sim Y)$

- F_3: $C \Leftrightarrow \sim (\sim A \,\&\, \sim B) \leftrightarrow (\sim C \vee A \vee B) \,\&\, (C \vee \sim A \,\&\, \sim B) \leftrightarrow$
  $(A \vee B \vee \sim C) \,\&\, (C \vee \sim A) \,\&\, (C \vee \sim B)$
- F_4: $D \Leftrightarrow \sim A \vee B \leftrightarrow (\sim D \vee \sim A \vee B) \,\&\, (A \,\&\, \sim B \vee D) \leftrightarrow$
  $(\sim D \vee \sim A \vee B) \,\&\, (A \vee D) \,\&(\sim B \vee D)$
- F_5: $E \Leftrightarrow \sim E \vee (C \Leftrightarrow D) \leftrightarrow$
  $(\sim E \vee \sim E \vee (\sim C \vee D) \,\&\, (C \vee \sim D)) \,\&\, (E \,\&\, \sim (C \Leftrightarrow D) \vee E) \leftrightarrow$
  $(\sim C \vee D \vee \sim E) \,\&\, (C \vee \sim D \vee \sim E) \,\&\, E \,\&\, (\sim C \vee \sim D \vee E) \,\&\, (C \vee D \vee E)$

Рассматриваемое нами далее множество дизъюнктов S имеет вид

$$S = \{\sim A \vee X, A \vee \sim X, \sim B \vee Y, B \vee \sim Y, A \vee B \vee \sim C, C \vee \sim A, C \vee \sim B, \sim D \vee \sim A \vee B, A \vee D, \sim B \vee D, \sim C \vee D \vee \sim E, C \vee \sim D \vee \sim E, E, \sim C \vee \sim D \vee E, C \vee D \vee E, \sim Y\}.$$

Применим к множеству S правило единичного дизъюнкта (дизъюнкт E), получим множество S_1:

$$S_1 = \{\sim A \vee X, A \vee \sim X, \sim B \vee Y, B \vee \sim Y, A \vee B \vee \sim C, C \vee \sim A, C \vee \sim B, \sim D \vee \sim A \vee B, A \vee D, \sim B \vee D, \sim C \vee D, C \vee \sim D, \sim Y\}.$$

В полученном множестве есть единичный дизъюнкт $\sim Y$, применим к множеству S_1 правило единичного дизъюнкта, получим множество S_2:

$$S_2 = \{\sim A \vee X, A \vee \sim X, \sim B, A \vee B \vee \sim C, C \vee \sim A, C \vee \sim B, \sim D \vee \sim A \vee B, A \vee D, \sim B \vee D, \sim C \vee D, C \vee \sim D\}.$$

Во множестве S_2 есть единичный дизъюнкт $\sim B$, после применения правила по этому дизъюнкту получим множество S_3:

$$S_3 = \{\sim A \vee X, A \vee \sim X, A \vee \sim C, C \vee \sim A, \sim D \vee \sim A, A \vee D, \sim C \vee D, C \vee \sim D\}.$$

К построенному множеству S_2 применим правило расщепления по дизъюнкту A, получим два множества S_4 и S_5:

$$S_4 = \{X, C, \sim D, \sim C \vee D, C \vee \sim D\};$$

$$S_5 = \{\sim X, \sim C, D, \sim C \vee D, C \vee \sim D\}.$$

Во множестве S_4 есть чистый литерал X. По правилу чистого литерала все дизъюнкты, содержащие этот литерал, можно исключить из рассматриваемого множества. Получим множество

$$S_6 = \{C, \sim D, \sim C \vee D, C \vee \sim D\}.$$

К множеству S_6 можно применить правило единичного дизъюнкта по литералу C, получим множество $S_7 = \{\sim D, D\}$. Множество S_7 противоречиво. Во множестве S_5 есть чистый литерал $\sim X$. Применяя к S_5 правило чистого литерала, а затем правило единичного дизъюнкта, получаем противоречивое

множество. Следовательно, утверждение Y — логическое следствие утверждений F_1—F_5, таким образом, карта Y — правильная карта и поиск сокровищ можно продолжать.

Упражнения

1. Перед нами три островитянина, A, B, C, каждый из которых либо рыцарь, либо лжец. Двое из них высказывают утверждения.
   - A: все мы лжецы;
   - B: ровно один из нас лжец.

   Кто такой островитянин C?

   Постройте множество дизъюнктов и примените для сокращения к нему правила Девиса и Патнема.
2. Путешественник проснулся на острове рыцарей и лжецов, название которого он не помнит. Он разговорился с двумя местными жителями A и B, которые сообщили ему следующее.
   - A: B — рыцарь и остров называется Майя;
   - B: A — лжец и остров называется Майя.

   Можно ли утверждать, что остров действительно называется Майя?

   Постройте множество дизъюнктов и примените для сокращения к нему правила Девиса и Патнема.
3. Внимание трех девушек Екатерины, Елены и Евгении привлек проезжающий мимо автомобиль.
   - Екатерина сказала: "Эта машина изготовлена в США, марка ее — Форд";
   - Елена возразила: "По-моему, эта машина из Германии, ее марка — Мерседес";
   - Евгения добавила: "Марка машины — Ауди, изготовлена не в США".

   Оказалось, что каждая из трех девушек права только в одном из своих высказываний. Какой марки автомобиль, и в какой стране изготовлен?

   Постройте множество дизъюнктов и примените для сокращения к нему правила Девиса и Патнема.

Задания по программированию

1. Напишите сценарий, который сокращает заданное множество дизъюнктов по правилам Девиса и Патнема.
2. Напишите сценарий, который проверяет правильность построения формулы исчисления высказываний методом рекурсивного спуска.

3. Напишите сценарий, который для заданной пропозициональной формулы находит все подформулы, значения которых совпадают с заданным.
4. Напишите сценарий, который по формуле исчисления высказываний, представленной в инфиксной нотации, строит формулу в префиксной нотации.
5. Напишите сценарий, который вычисляет значение пропозициональной формулы, находящейся в постфиксной нотации.
6. Напишите сценарий, при выполнении которого пропозициональная формула преобразуется таким образом, что знак отрицания встречается лишь перед переменной.
7. Заданы две пропозициональные формулы. Напишите сценарий, который проверяет, являются ли формулы эквивалентными.
8. Переменная x в формуле A называется *фиктивной*, если существует формула B, эквивалентная формуле A, и не содержащая вхождений переменной x. Напишите сценарий, который находит в заданной формуле все фиктивные переменные.
9. Напишите сценарий, который по заданной пропозициональной формуле определяет, верно ли, что на наборах противоположных значений переменных формула принимает противоположное значение.
10. Пара скобок в пропозициональной формуле называется избыточной, если после ее удаления формула остается эквивалентной исходной формуле. Напишите сценарий, который удаляет все избыточные пары скобок.

Занимательные задачи

Если задача описывается с помощью формул исчисления высказываний, то многие этапы решения задачи (определение общезначимости, противоречивости или выполнимости формул, приведение к КНФ, сокращение множества дизъюнктов и др.) можно автоматизировать, в чем мы могли убедиться, выполняя различные задания по написанию сценариев.

Полученные знания рекомендуется применить при создании сайта, позволяющего пользователю решать логические задачи и проверять правильность решения. Для выбранной из предложенного списка задачи пользователю требуется ввести формулы, соответствующие условию задачи, и указать предполагаемое следствие. Созданный сценарий должен проверять правильность решения задачи любым из описанных способов. Предлагается следующий список задач, который может быть дополнен или изменен разработчиком.

Прекраснейшая богиня

В одном старинном задачнике описан суд Париса следующим образом. Богини Гера, Афродита и Афина пришли к Парису, чтобы он решил, кто из них прекраснее. Богини (обозначим их α, β, χ) высказали следующие утверждения.

- α: я самая прекрасная;
- β: Афродита не самая прекрасная;
- χ: я самая прекрасная;
- α: Гера не самая прекрасная;
- β: я самая прекрасная.

Парис предположил, что высказывания прекраснейшей из богинь истинны, а все утверждения двух остальных ложны. Мог ли Парис вынести свое решение? Кто прекраснейшая из богинь? Какие богини скрываются под именами α, β, χ?

Игроки на скачках

Перед началом бегов на ипподроме четверо знатоков из числа зрителей обсуждали шансы трех фаворитов: *A*, *B* и *C*. Во время обсуждения были высказаны следующие утверждения.

- заезд выиграет *A* или *C*;
- если *A* придет вторым, то выиграет *B*;
- если *A* придет третьим, то *C* не выиграет;
- вторым придет *A* или *B*.

После заезда выяснилось, что фавориты *A*, *B* и *C* действительно заняли первые три места и что все четыре высказывания знатоков оказались истинными. Как фавориты поделили между собой места?

Поиск пути выхода из лабиринта

Одинокий путник заблудился в лабиринте и попал в комнату, из которой можно выйти в одну из четырех дверей: *X*, *Y*, *Z* и *W*. По крайней мере одна из дверей ведет к выходу из лабиринта. Того, кто выходит через другую дверь, ожидает дракон, живущий в лабиринте.

К удивлению путника в комнате находятся восемь жрецов, каждый из которых либо рыцарь и говорит всегда только правду, либо лжец и всегда лжет: *A*, *B*, *C*, *D*, *E*, *F*, *G* и *H*. Нашему путнику жрецы сообщили следующее:

- *A*: *X* — дверь, ведущая к выходу из лабиринта;
- *B*: по крайней мере одна из дверей *X* и *Y* ведет к выходу из лабиринта;

- C: A и B — рыцари;
- D: обе двери X и Y ведут к выходу из лабиринта;
- E: обе двери X и Z ведут к выходу из лабиринта;
- F: либо D, либо E — рыцарь;
- G: если C — рыцарь, то F — рыцарь;
- H: если G и я сам — рыцари, то A — рыцарь.

Какую же дверь выбрать одинокому путнику?

Расследование преступления

Какие выводы вы бы сделали из следующих фактов?

- Если A виновен и B невиновен, то C виновен.
- C никогда не действует в одиночку.
- A никогда не ходит на дело вместе с C.
- Никто, кроме A, B и C, в преступлении не замешан и, по крайней мере, один из этой тройки виновен.

Чья виновность не вызывает сомнений?

Обвинитель на острове рыцарей и лжецов

На одном из островов, населенных рыцарями и лжецами, за совершение преступления судили двух местных жителей X и Y. Об обвинителе было известно, что он либо рыцарь, либо лжец. На суде обвинитель сделал два заявления:

- X виновен;
- X и Y не могут быть виновны оба.

Кто обвинитель: рыцарь или лжец?

Суд на острове рыцарей и лжецов

Предположим, что обвинитель сделал следующих два заявления:

- либо X не виновен, либо Y виновен;
- X виновен.

К какому заявлению вы бы пришли на основании этих заявлений?

Любовь рыцаря или лжеца

Предположим, что некоторый человек, который может быть либо рыцарем, либо лжецом, делает следующие заявления:

- я люблю Линду;

- если я люблю Линду, то я люблю Диану.

Кто же он: рыцарь или лжец?

Интервью на острове рыцарей и лжецов

У трех обитателей *A*, *B* и *C* острова рыцарей и лжецов взяли интервью, в ходе которого они высказали следующие утверждения:

- *A*: *B* — рыцарь;
- *B*: если *A* — рыцарь, то *C* — рыцарь.

Можно ли определить, кто из *A*, *B* и *C* рыцарь и кто лжец?

Путешественник на островах

Путешественник ищет остров с названием Майя. Исследуя третий остров, путешественник встретил двух жителей *A* и *B*, которые заявили следующее:

- *A*: по крайней мере, один из нас лжец, и этот остров называется Майя;
- *B*: совершенно верно!

Можно ли утверждать, что третий остров действительно называется Майя?

Исследование шестого острова

Путешественник ищет остров с названием Майя. Исследуя шестой остров, путешественник встретил двух жителей *A* и *B*, которые заявили следующее:

- *A*: либо *B* — рыцарь, либо этот остров называется Майя;
- *B*: либо *A* — лжец, либо этот остров называется Майя.

Можно ли утверждать, что шестой остров действительно называется Майя?

Поиск виновных

В этом случае подсудимых четверо: *A*, *B*, *C* и *D*. Установлено следующее:

- если *A* и *B* оба виновны, то *C* был соучастником;
- если *A* виновен, то, по крайней мере, один из обвиняемых *B* или *C* был соучастником;
- если *C* виновен, то *D* был соучастником;
- если *A* невиновен, то *D* виновен.

Кто из подсудимых виновен вне всякого сомнения, и чья вина остается под сомнением?

Искатель приключений

Предположим, что некоторый искатель приключений хочет добраться до острова сокровищ. Путешественнику предложили три карты: X, Y и Z. Только одна из карт правильная и указывает путь к острову сокровищ. В комнате, где лежали карты, находились пятеро колдунов: A, B, C, D и E. Каждый колдун был либо рыцарем, либо лжецом и каждый дал путешественнику совет:

- A: X — правильная карта;
- B: Y — правильная карта;
- C: неверно, что A и B — оба лжеца;
- D: либо A — лжец, либо B — рыцарь;
- E: либо A — лжец, либо C и D — однотипны (либо оба рыцари, либо оба лжеца).

Какая же из карт правильная?

Глава 21

Поиск доказательств

Данная глава посвящена рассмотрению методов автоматизации поиска доказательства. В *главе 20* при решении задач использовалась следующая схема. Пусть у нас имеются аксиомы или уже доказанные факты

$$F_1, F_2, \ldots, F_m.$$

Нам требуется определить, следует ли утверждение G из утверждений F_1, F_2, ..., F_m. Согласно теореме 2 о логическом следствии для решения этого вопроса нам достаточно определить, является ли формула

$$F_1 \,\&\, F_2 \,\&\, \ldots \,\&\, F_m \,\&\, \sim G.$$

противоречивой. Данную формулу можно преобразовать к равнозначной формуле, находящейся в КНФ, и в дальнейшем рассматривать множество дизъюнктов

$$S = \{S_1, S_2, \ldots, S_n\}.$$

По теореме о полноте метода резолюций для того, чтобы определить противоречивость (невыполнимость) множества S, достаточно найти опровержение в S. Правило резолюции позволяет извлекать новые данные из уже имеющихся, а теорема о резолюции гарантирует, что, если исходные дизъюнкты были истинными, то истинным будет и построенная по ним резольвента. Понятие вывода — формализованный аналог понятия доказательства.

Рассмотрим автоматические методы поиска доказательства. Теоретический материал иллюстрируется примерами, и предлагаются задания для создания сценариев, реализующих методы поиска решений.

Стратегия насыщения уровней

Согласно теореме о полноте метода резолюций для того, чтобы доказать, что множество дизъюнктов противоречиво, достаточно построить опровержение в S.

При поиске опровержения строятся всевозможные резольвенты и добавляются к множеству дизъюнктов S. Построение резольвент продолжается до тех пор, пока не будет получен требуемый дизъюнкт □. Далее необходимо проследить в обратном порядке, как был получен дизъюнкт □, и восстановить вывод. При этом может оказаться, что порождено много лишних дизъюнктов,

которые не нужны при построении вывода. *Стратегия насыщения уровней* состоит в следующем: формируются множества дизъюнктов S^0, S^1, …, S^m, где

- $S^0 = S$
- S^n = {резольвента C_1 и $C_2 \mid C_1 \in (S^0 \cup ... \cup S^{n-1})$ и $C_2 \in S^{n-1}$}, $n = 1, 2, 3, \ldots$

Когда резольвента построена, она добавляется к концу списка, сформированному на данный момент.

Пример доказательства противоречивости множества

Рассмотрим следующее множество дизъюнктов:

$$S = \{\sim A \vee C, \sim B \vee C, \sim A \vee B \vee C, \sim C \vee D, A \vee D, \sim D\}.$$

Требуется доказать, что оно противоречиво, и продемонстрировать на этом примере использование стратегии насыщения уровней.

S^0: (1) $\sim A \vee C$

(2) $\sim B \vee C$

(3) $\sim A \vee B \vee C$

(4) $\sim C \vee D$

(5) $A \vee D$

(6) $\sim D$

S^1: (7) $\sim A \vee D$ из (1) и (4)

(8) $C \vee D$ из (1) и (5)

(9) $\sim A \vee C$ из (2) и (3)

(10) $\sim B \vee D$ из (2) и (4)

(11) $\sim A \vee B \vee D$ из (3) и (4)

(12) $B \vee C \vee D$ из (3) и (5)

(13) $\sim C$ из (4) и (6)

(14) A из (5) и (6)

(15) $\sim A$ из (1) и (13)

S^2: (16) C из (1) и (14)

(17) $\sim A \vee C \vee D$ из (2) и (11)

(18) $C \vee D$ из (2) и (12)

(19) $\sim B$ из (2) и (13)

(20) $\sim A \vee B$ из (3) и (13)

(21) $B \vee C$ из (3) и (14)

(22) D из (4) и (8)
(23) $\sim A \vee D$ из (4) и (9)
(24) $B \vee D$ из (4) и (12)
(25) D из (5) и (7)
(26) $C \vee D$ из (5) и (9)
(27) $\sim A$ из (6) и (7)
(28) C из (6) и (8)
(29) $\sim B$ из (6) и (10)
(30) $\sim A \vee B$ из (6) и (11)
(31) $B \vee C$ из (6) и (12)

S^3: (32) $\sim A \vee C$ из (2) и (21)
(33) $C \vee D$ из (2) и (24)
(34) $\sim A \vee C$ из (2) и (30)
(35) C из (2) и (31)
(36) $\sim A \vee C$ из (3) и (19)
(37) $\sim A \vee C$ из (3) и (29)
(38) D из (4) и (16)
(39) $\sim A \vee D$ из (4) и (17)
(40) D из (4) и (18)
(41) $B \vee D$ из (4) и (21)
(42) D из (4) и (29)
(43) $C \vee D$ из (5) и (17)
(44) $B \vee D$ из (5) и (20)
(45) D из (5) и (23)
(46) D из (5) и (27)
(47) $B \vee D$ из (5) и (30)
(48) $\square$ из (6) и (22)

Нетрудно видеть, что при поиске тождественно ложного дизъюнкта были порождены лишние дизъюнкты, не относящиеся к выводу. Многие дизъюнкты порождались неоднократно. На самом деле для того чтобы построить опровержение, требовалось породить дизъюнкты (8), (22), (48). На рис. 21.1 приведено дерево опровержения для S, построенное согласно стратегии насыщения уровней.

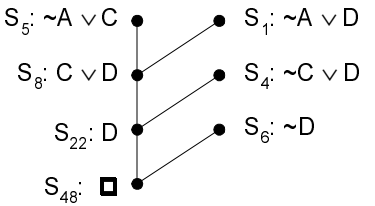


Рис. 21.1. Дерево опровержения согласно стратегии насыщения уровней

Упражнение

Для следующего множества дизъюнктов

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}$$

найдите тождественно ложный дизъюнкт, воспользовавшись методом насыщения уровней.

Задание по программированию

Напишите сценарий, который по заданному множеству дизъюнктов находит дизъюнкт □, используя стратегию насыщения уровней. При этом требуется определить число порожденных дизъюнктов, длину построенного вывода и время, затраченное на поиск доказательства.

Стратегия вычеркивания

Для решения проблемы порождения избыточного числа дизъюнктов применяется *стратегия вычеркивания*.

Дизъюнкт C является *поддизъюнктом* дизъюнкта D, если верно $C \subseteq D$. Дизъюнкт D в этом случае называется *наддизъюнктом* для C.

Стратегия вычеркивания используется вместе со стратегией насыщения уровней следующим образом: сначала выписываются дизъюнкты из множества

$$(S^0 \cup \ldots \cup S^{n-1})$$

по порядку, затем вычисляются резольвенты путем сравнения каждого дизъюнкта $C_1 \in (S^0 \cup \ldots \cup S^{n-1})$ с дизъюнктом $C_2 \in S^{n-1}$.

Если резольвента является тавтологией или наддизъюнктом какого-либо дизъюнкта из списка уже построенных, она вычеркивается, в противном случае добавляется к порождаемому списку дизъюнктов.

Пример применения стратегии вычеркивания

Применим стратегию вычеркивания к поиску дизъюнкта □ из множества дизъюнктов S, рассмотренного в предыдущем примере.

В методе насыщения уровней при построении резольвенты (9) получили дизъюнкт $\sim A \vee C$, который совпадает с дизъюнктом (1), построенную резольвенту в список не включаем. При построении резольвенты (17) был получен дизъюнкт $\sim A \vee C \vee D$, он также не включается в список, потому что является наддизъюнктом дизъюнкта $\sim A \vee C$. Приведем дальнейший список без комментариев.

S^0: (1) $\sim A \vee C$

(2) $\sim B \vee C$

(3) $\sim A \vee B \vee C$

(4) $\sim C \vee D$

(5) $A \vee D$

(6) $\sim D$

S^1: (7) $\sim A \vee D$ из (1) и (4)

(8) $C \vee D$ из (1) и (5)

(9) $\sim B \vee D$ из (2) и (4)

(10) $\sim A \vee B \vee D$ из (3) и (4)

(11) $B \vee C \vee D$ из (3) и (5)

(12) $\sim C$ из (4) и (6)

(13) A из (5) и (6)

(14) $\sim A$ из (1) и (12)

S^2: (15) C из (1) и (13)

(16) $\sim B$ из (2) и (12)

(17) D из (4) и (8)

(18) □ из (6) и (17)

В этот список занесено 12 порожденных резольвент в отличие от 42 дизъюнктов из списка, полученного в результате использования стратегии насыщения уровней. Для построения опровержения требовалось породить дизъюнкты (8) и (17). Дерево вывода такое же, как и дерево, полученное по методу насыщения уровней (см. рис. 21.1), лишь нумерация дизъюнктов иная.

Упражнение

Для следующего множества дизъюнктов

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}$$

найдите тождественно ложный дизъюнкт, используя стратегию вычеркивания.

Задание по программированию

Напишите сценарий, который по заданному множеству дизъюнктов находит дизъюнкт □, используя стратегию вычеркивания. При этом требуется определить число порожденных дизъюнктов, длину построенного вывода и время, затраченное на поиск доказательства.

Линейная резолюция

Линейная резолюция соответствует следующей схеме поиска доказательства. Мы выбрали некоторое утверждение из исходного множества, применили к нему и к другому утверждению правило резолюции, получили резольвенту. К полученной резольвенте и какому-либо другому дизъюнкту опять применили правило резолюции, получили следующее высказывание, и т. д. На каждом этапе построения доказательства одно из высказываний — это только что полученное в результате применения правила резолюции утверждение. Процесс повторяется до тех пор, пока не будет получен тождественно ложный дизъюнкт.

Рассмотрим множество дизъюнктов $S = \{S_1, S_2, \ldots, S_n\}$. Возьмем дизъюнкт C_0 из S. Линейный вывод дизъюнкта C из множества дизъюнктов S — это вывод, изображенный с помощью схемы на рис. 21.2, в котором выполняются условия:

1. Для $i = 0, 1, \ldots, k-1$ дизъюнкт C_{i+1} есть резольвента дизъюнкта C_i (называемого *центральным* дизъюнктом) и B_i (называемого *боковым*).
2. Каждый B_i либо принадлежит S, либо есть C_j для некоторого j, такого, что $j < i$.

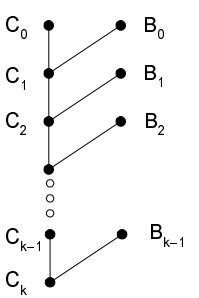


Рис. 21.2. Схема построения линейного вывода

Задача об автомобилях

Внимание трех девушек Екатерины, Елены и Евгении привлек проезжающий мимо автомобиль.

- Екатерина сказала: "Эта машина изготовлена в США, марка ее — Форд";
- Елена возразила: "По-моему, эта машина из Германии, ее марка — Мерседес";
- Евгения добавила: "Марка машины — Ауди, изготовлена не в США".

Оказалось, что каждая из трех девушек права только в одном из своих высказываний. Какой марки автомобиль, и в какой стране изготовлен?

Введем следующие обозначения:

- R — автомобиль изготовлен в США;
- G — автомобиль изготовлен в Германии;
- F — марка автомобиля Форд;
- M — марка автомобиля Мерседес;
- A — марка автомобиля Ауди.

Высказывания трех девушек с учетом того, что одно из них истинно, а одно ложно, записываются с помощью формул исчисления высказываний следующим образом:

- Екатерина: $R \,\&\, \sim F \vee \sim R \,\&\, F$
- Елена: $G \,\&\, \sim M \vee \sim G \,\&\, M$
- Евгения: $G \,\&\, \sim A \vee \sim G \,\&\, A$

Утверждение, что автомобиль изготовлен в одной из двух стран, соответствует формуле

$$R \,\&\, \sim G \vee \sim R \,\&\, G.$$

И, наконец, утверждение, что проезжающий автомобиль принадлежит какой-либо из трех марок:

$$F \,\&\, \sim M \,\&\, \sim A \vee \sim F \,\&\, M \,\&\, \sim A \vee \sim F \,\&\, \sim M \,\&\, A.$$

Мы хотим определить, следует ли утверждение $F \,\&\, G$ (автомобиль марки Форд, изготовлен в Германии) из рассматриваемых пяти утверждений. Формулу

$$(R \,\&\, \sim F \vee \sim R \,\&\, F) \,\&\, (G \,\&\, \sim M \vee \sim G \,\&\, M) \,\&\, (G \,\&\, \sim A \vee \sim G \,\&\, A) \,\&$$
$$(F \,\&\, \sim M \,\&\, \sim A \vee \sim F \,\&\, M \,\&\, \sim A \vee \sim F \,\&\, \sim M \,\&\, A) \,\&\, \sim (F \,\&\, G)$$

преобразуем к КНФ и получим следующее множество дизъюнктов:

$$S = \{R \vee F,\ \sim R \vee \sim F,\ G \vee M,\ \sim G \vee \sim M,\ G \vee A,\ \sim G \vee \sim A,\ R \vee G,\ \sim R \vee \sim G,$$
$$\sim F \vee \sim M \vee \sim A,\ \sim F \vee \sim M \vee A,\ \sim F \vee M \vee \sim A,\ F \vee \sim M \vee \sim A,$$
$$F \vee M \vee A,\ \sim F \vee \sim G\}.$$

Дизъюнкты множества S будем обозначать S_1, S_2, ..., S_{14}. В качестве верхнего дизъюнкта возьмем дизъюнкт S_1. Построим линейный вывод, изображенный на рис. 21.3.

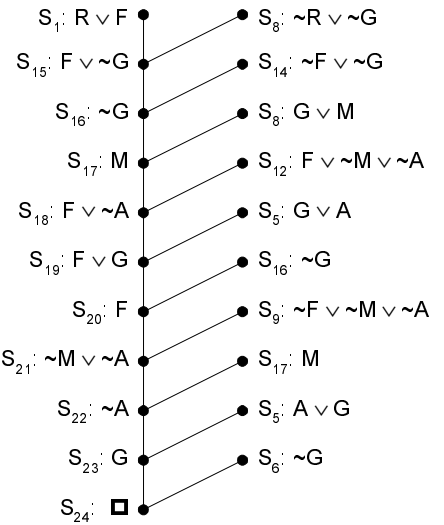


Рис. 21.3. Линейное опровержение для задачи об автомобилях

Упражнения

1. Постройте линейный вывод для решения задачи об автомобилях, используя в качестве верхнего дизъюнкта дизъюнкт S_{11}.
2. Постройте опровержение согласно стратегии насыщения уровней для решения задачи об автомобилях.
3. Постройте опровержение согласно стратегии вычеркивания для решения задачи об автомобилях.

Задание по программированию

Напишите сценарий, который по заданному множеству дизъюнктов и выводу определяет, является ли этот вывод линейным.

Входная резолюция

Рассмотрим некоторые частные случаи линейной резолюции.

Каждый дизъюнкт исходного множества S назовем *входным* дизъюнктом.

Входная резолюция — это резолюция, в которой один из дизъюнктов входной дизъюнкт.

Входной вывод — линейный вывод, в котором каждое применение резолюции является входной резолюцией.

Входное опровержение — входной вывод дизъюнкта $\square$ из S.

Задача о формировании экипажа

Формируется экипаж для кругосветного путешествия. Каждый кандидат проходит тестирование. Окончательное слово остается за командиром экипажа. Три человека Семен, Максим и Дмитрий проходят тестирование. После беседы с ними командир высказал свое мнение помощнику необычным образом:

- я возьму в путешествие одного из кандидатов;
- если я возьму Семена, но не возьму Максима, то возьму также и Дмитрия;
- членами экипажа станут либо Дмитрий и Максим, либо ни один из них;
- если в экипаж будет принят Дмитрий, то также будет принят и Семен.

Никто не усомнился в том, что высказывания командира экипажа истинны. Дмитрий после собеседования на вопрос друга о том, отправляется ли он в кругосветное путешествие, ответил утвердительно. Прав ли Дмитрий?

Введем следующие обозначения:

- C — Семен отправится в путешествие;
- M — Максим отправится в путешествие;
- D — Дмитрий отправится в путешествие.

Запишем формулами четыре высказывания командира экипажа и ответ Дмитрия:

- F_1: $C \vee M \vee D$
- F_2: $C \,\&\, \sim D \Rightarrow M$
- F_3: $D \,\&\, M \vee \sim D \,\&\, \sim M$
- F_4 :$D \Rightarrow C$
- G: D

Формулу, построенную согласно теореме 2 о логическом следствии:

$$(C \vee M \vee D) \,\&\, (C \,\&\, \sim D \Rightarrow M) \,\&\, (D \,\&\, M \vee \sim D \,\&\, \sim M) \,\&\, (D \Rightarrow C) \,\&\, \sim D$$

приведем к КНФ и получим интересующее нас множество дизъюнктов:

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

Считаем, что наши исходные дизъюнкты пронумерованы от S_1 до S_6. Построим входное опровержение в S (рис. 21.4).

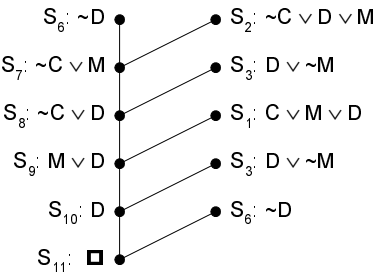


Рис. 21.4. Линейное опровержение для задачи о формировании экипажа

Обратите внимание на то, что в линейном выводе на рис. 21.4 все боковые дизъюнкты являются входными дизъюнктами, поэтому и опровержение — входное.

Упражнение

Постройте входное опровержение для задачи об автомобилях, используя дизъюнкт, отличный от S_1, в качестве верхнего дизъюнкта.

Задание по программированию

Напишите сценарий, который по заданному множеству дизъюнктов и выводу определяет, является ли этот вывод входным.

Единичная резолюция

Рассмотрим еще один частный случай линейной резолюции, так называемую единичную резолюцию.

Единичная резолюция — это резолюция, в которой, по крайней мере, один из дизъюнктов является единичным.

Единичный вывод — линейный вывод, в котором каждое применение резолюции является единичной резолюцией.

Единичное опровержение — единичный вывод дизъюнкта □ из S.

Пример единичного опровержения

Построим единичное опровержение для задачи о формировании экипажа. Напомним, что рассматривается множество дизъюнктов

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

На рис. 21.5 представлено единичное опровержение. Заметим, что это опровержение не является входным. При получении дизъюнкта □ использовались дизъюнкты S_{11} и S_7, и дизъюнкт S_7 не является входным.

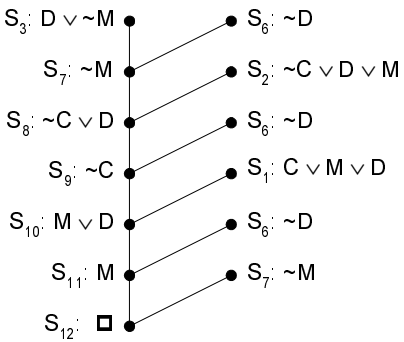


Рис. 21.5. Единичное опровержение для задачи о формировании экипажа

Упражнение

Постройте единичное опровержение для задачи об автомобилях, используя дизъюнкт, отличный от S_1, в качестве верхнего дизъюнкта.

Задание по программированию

Напишите сценарий, который по заданному множеству дизъюнктов и выводу определяет, является ли этот вывод единичным.

Эквивалентность входной и единичной резолюций

Теорема об эквивалентности входной и единичной резолюций

Для противоречивого множества дизъюнктов S единичное опровержение имеется тогда и только тогда, когда имеется входное.

Доказательство. Пусть A — множество пропозициональных переменных, входящих во множество дизъюнктов S (база S). Если A состоит из единственного элемента Q, то среди множества дизъюнктов S имеются единичные дизъюнкты Q и ~ Q. Резольвентой дизъюнктов Q и ~ Q является дизъюнкт □. Вывод дизъюнкта □ из S является как входным, так и единичным. Следовательно, для этого случая теорема верна.

Предположим, что теорема верна для случая, когда множество A состоит из i элементов, где $1 \le i \le n$.

Пусть имеется единичное опровержение в S. Тогда множество дизъюнктов S должно содержать хотя бы один единичный дизъюнкт L, где L — литерал. Множество дизъюнктов S^* получаем из S исключением дизъюнктов, содержащих L, и вычеркиванием литерала $\sim L$ из остальных дизъюнктов. Поскольку множество S противоречиво, то противоречивым будет и множество S^*, т. к. S^* мы получили по правилу единичного дизъюнкта. Раз имеется единичное опровержение в S, то должно существовать единичное опровержение в S^*.

База множества дизъюнктов S^* содержит n или менее элементов. По индукционному предположению имеется S^*-входное опровержение Z_1. Пусть Z — вывод, получающийся из Z_1 возвращением литерала $\sim L$ в те дизъюнкты, из которых он был вычеркнут.

Пусть литерал T — дизъюнкт, получившийся в корне дерева для вывода Z. Очевидно, что Z является S-входным выводом дизъюнкта T из S. Дизъюнкт T — это либо $\square$, либо $\sim L$. Если T есть $\square$, то мы построили входное опровержение в S. Если же T есть $\sim L$, то строим резольвенту T с дизъюнктом L, который является входным дизъюнктом. Таким образом, и в этом случае мы построили входное опровержение.

Пример исключения дизъюнктов

Для задачи о формировании экипажа на рис. 21.6 приведено единичное опровержение. Рассматриваемое множество дизъюнктов

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}$$

содержит единичный дизъюнкт $\sim D$.

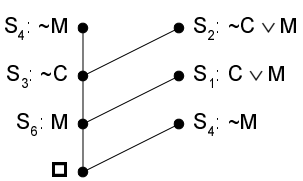


Рис. 21.6. Единичное опровержение для задачи о формировании экипажа после сокращения дизъюнктов

Построим множество S^* исключением из S всех дизъюнктов, содержащих литерал $\sim D$, и вычеркиванием из оставшихся дизъюнктов литерала D. Получим

$$S^* = \{C \vee M, \sim C \vee M, \sim M\}.$$

На рис. 21.6 приведем единичное опровержение Z_1 в S^*. Это опровержение входное. Вывод Z получим, возвращая литерал D в те дизъюнкты, из которых он был вычеркнут. Получим вывод Z, изображенный на рис. 21.7.

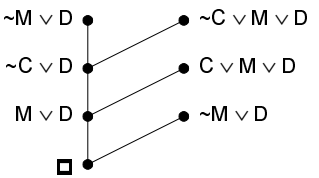


Рис. 21.7. Входной вывод сокращенного дизъюнкта

Вывод Z является S-входным выводом дизъюнкта D из S. На рис. 21.8 построено S-входное опровержение.

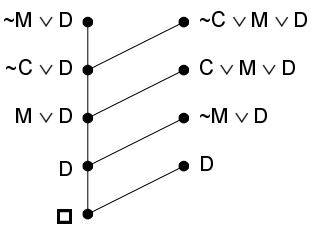


Рис. 21.8. Входное опровержение для задачи о формировании экипажа

Предположим, что имеется входное опровержение в S. Множество S должно содержать, по крайней мере, один единичный дизъюнкт L. Такой литерал необходим при получении дизъюнкта □, один из исходных дизъюнктов должен быть входным. Обозначим через N множество дизъюнктов, в которых содержатся литералы L или $\sim L$, через $\mathrm{Res}(Y, C)$ — резольвенту дизъюнктов Y и C. Построим множество

$$S^* = S \cup \{\mathrm{Res}(L, C) \mid C \in S\} - N.$$

Так как существует S-входное опровержение, то должно существовать S^*-входное опровержение. Но множество S^* содержит n или менее пропозициональных переменных. По индукционному предположению можно построить единичное опровержение в S^*. Каждый дизъюнкт из S^* является либо дизъюнктом из S, либо резольвентой, полученной применением единичной резолюции к дизъюнкту L и некоторому дизъюнкту из S. Поэтому можно построить единичное опровержение в S. Теорема доказана.

Пример построения входного опровержения

Рассматриваемое множество дизъюнктов таково:

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

На рис. 21.3 построено входное опровержение в S. Множество S содержит, по крайней мере, один единичный дизъюнкт. Обратите внимание на построение резольвенты по дизъюнктам S_{10} и S_6. В нашем случае это дизъюнкт $\sim D$. Множество N совпадает с S, т. к. каждый дизъюнкт из S содержит либо D, либо $\sim D$.

Есть множество

$$\{\text{Res}(L,\ C) \mid C \in S\} = \{C \vee M,\ \sim C \vee M,\ \sim M\}.$$

Построим множество

$$S_1 = S \cup \{\text{Res}(L,\ C) \mid C \in S\} - N = \{C \vee M,\ \sim C \vee M,\ \sim M\}.$$

На рис. 21.9 изображено S_1-входное опровержение. Заметим, что оно одновременно является и единичным опровержением.

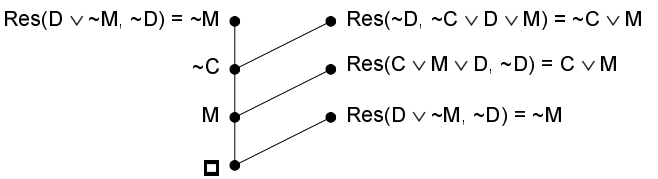


Рис. 21.9. Входное опровержение для множества S_1

На рис. 21.10 показано, как по S_1-входному опровержению можно построить единичное опровержение в S.

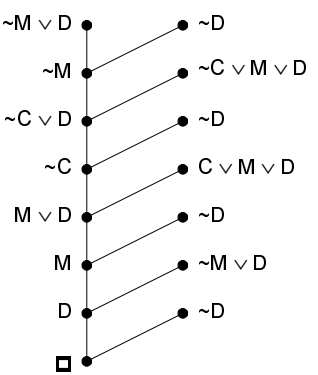


Рис. 21.10. Единичное опровержение, построенное по входному

Упражнение

Определите, можно ли построить единичное опровержение для задачи о формировании экипажа.

Задания по программированию

1. Напишите сценарий, который по заданному входному *S*-опровержению строит единичное *S*-опровержение.
2. Напишите сценарий, который по заданному единичному *S*-опровержению строит входное *S*-опровержение.

Свойства входной и единичной резолюций

Единичную и входную резолюцию легче реализовать, чем другие стратегии поиска доказательства, но у входной и единичной резолюций есть один существенный недостаток: обе стратегии не являются полными.

Задача о кандидатах в министры

В одной стране проживают рыцари, которые говорят только правду, лжецы, которые всегда лгут, и так называемые нормальные люди. Нормальный человек может говорить как правду, так и ложь. Три различные партии (обозначим их *E*, *F* и *G*) выдвигают на пост министра трех кандидатов: *A*, *B* и *C* соответственно. Каждый из кандидатов, выступая от имени своей партии, предложил программу развития экономики. После встречи с президентом во время пресс-конференции на вопрос журналиста: "Программа какой партии получила одобрение президента?" каждый из кандидатов ответил:

- кандидат *A*: одобрение получила не наша программа;
- кандидат *B*: одобрение получила не та программа, которую предложил кандидат *A*;
- кандидат *C*: к сожалению, одобрена не наша программа.

В этот же день во время прямого телеэфира телеведущая задала кандидатам тот же самый вопрос: "Программа какой партии получила одобрение президента?" и получила уже другие ответы:

- кандидат *A*: одобрение получила программа не партии *F*;
- кандидат *B*: одобрение получила программа партии *G*;
- кандидат *C*: одобрение получила программа, предложенная кандидатом *A*.

Известно, что один из кандидатов — рыцарь, один — лжец, и один — нормальный человек, который в одном случае сказал правду, а в другом ложь. Программа какой партии получила одобрение президента?

Для решения задачи введем обозначения. Обозначим через *E* (соответственно *F* и *G*) утверждение, что получила одобрение президента программа,

предложенная партией E (соответственно F и G). Тогда высказывания кандидатов журналисту и ведущему имеют следующий вид:

- кандидат A: $\sim E, F$;
- кандидат B: $\sim E, G$;
- кандидат C: $\sim G, E$.

Так как известно, что один из кандидатов — рыцарь, один — лжец и один — нормальный человек, то возможны следующие шесть вариантов, которые представлены в табл. 21.1.

Таблица 21.1. *Варианты кандидатов*

№	Кандидат A	Кандидат B	Кандидат C
1	рыцарь	лжец	нормальный человек
2	рыцарь	нормальный человек	лжец
3	лжец	рыцарь	нормальный человек
4	лжец	нормальный человек	рыцарь
5	нормальный человек	рыцарь	лжец
6	нормальный человек	лжец	рыцарь

Учитывая, что высказывания рыцаря истинны, лжеца ложны, а из двух высказываний нормального человека одно ложно, а одно истинно, получаем следующую формулу, описывающую условие задачи и соответствующую рассмотренным шести вариантам:

$$\sim E \,\&\, \sim F \,\&\, E \,\&\, \sim G \,\&\, (\sim G \,\&\, \sim E \vee G \,\&\, E) \vee$$
$$\sim E \,\&\, \sim F \,\&\, (\sim E \,\&\, \sim G \vee E \,\&\, G) \,\&\, G \,\&\, \sim E \vee$$
$$E \,\&\, F \,\&\, \sim E \,\&\, G \,\&\, (\sim G \,\&\, \sim E \vee G \,\&\, E) \vee$$
$$E \,\&\, F \,\&\, (\sim E \,\&\, \sim G \vee E \,\&\, G) \,\&\, \sim G \,\&\, E \vee$$
$$(\sim E \,\&\, F \vee E \,\&\, \sim F) \,\&\, \sim E \,\&\, G \,\&\, G \,\&\, \sim E \vee$$
$$(\sim E \,\&\, F \vee E \,\&\, \sim F) \,\&\, E \,\&\, \sim G \,\&\, \sim G \,\&\, E.$$

Эту формулу можно упростить, выполняя эквивалентную замену формул вида

$$f \,\&\, \square \leftrightarrow \square \quad \text{и} \quad f \vee \square \leftrightarrow f.$$

Получим следующую формулу:

$$\sim E \,\&\, \sim F \,\&\, (\sim E \,\&\, \sim G \vee E \,\&\, G) \,\&\, G \,\&\, \sim E \vee$$
$$E \,\&\, F \,\&\, (\sim E \,\&\, \sim G \vee E \,\&\, G) \,\&\, \sim G \,\&\, E \vee$$
$$(\sim E \,\&\, F \vee E \,\&\, \sim F) \,\&\, \sim E \,\&\, G \,\&\, G \,\&\, \sim E \vee$$
$$(\sim E \,\&\, F \vee E \,\&\, \sim F) \,\&\, E \,\&\, \sim G \,\&\, \sim G \,\&\, E.$$

Приведем последнюю формулу к КНФ. Для этого вычислим ее значения во всех интерпретациях и результаты представим в табл. 21.2.

Таблица 21.2. *Таблица истинности для формул задачи*

E	F	G	Значение формулы
И	И	И	Л
И	И	Л	Л
И	Л	И	Л
И	Л	Л	И
Л	И	И	И
Л	И	Л	Л
Л	Л	И	Л
Л	Л	Л	Л

Заметим, что одобрение президента получила только одна программа, поэтому рассмотрим формулу

$$E \,\&\, \sim F \,\&\, \sim G \vee \sim E \,\&\, F \,\&\, \sim G \vee \sim E \,\&\, \sim F \,\&\, G.$$

Построим для этой формулы КНФ, а результаты представим в табл. 21.3.

Таблица 21.3. *Таблица истинности*

E	F	G	Значение формулы
И	И	И	Л
И	И	Л	Л
И	Л	И	Л
И	Л	Л	И
Л	И	И	Л
Л	И	Л	И
Л	Л	И	И
Л	Л	Л	Л

Предположим, что мы хотим показать, что одобрение получила программа партии *E*. После построения КНФ по табл. 21.2 и 21.3, и добавления предполагаемого следствия, интересующее нас множество дизъюнктов будет следующим:

$$S = \{\sim E \vee \sim F \vee \sim G, \sim E \vee \sim F \vee G, \sim E \vee F \vee \sim G, E \vee \sim F \vee \sim G,$$
$$E \vee \sim F \vee G, E \vee F \vee \sim G, E \vee F \vee G, \sim E\}.$$

Сократим множество S по правилу единичного дизъюнкта, получим множество S_1:

$$\{\sim F \vee \sim G, \sim F \vee G, F \vee \sim G, F \vee G\}.$$

Это множество противоречиво, если противоречиво исходное множество S. Линейное опровержение с верхним дизъюнктом $\sim F \vee \sim G$ изображено на рис. 21.11. Заметим, что линейный вывод на рис. 21.11 не является входным, т. к. боковой дизъюнкт $\sim F$ не содержится в S.

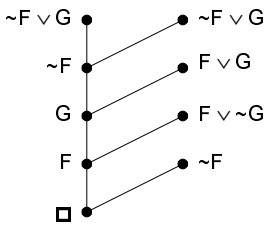


Рис. 21.11. Линейное опровержение для задачи о кандидатах в министры

Для множества дизъюнктов

$$\{\sim F \vee \sim G, \sim F \vee G, F \vee \sim G, F \vee G\}$$

нельзя построить единичного опровержения, а, следовательно, по теореме об эквивалентности входной и единичной резолюции нельзя построить и входное опровержение. Таким образом, методы входной и единичной резолюций не являются полными.

Теорема о полноте метода линейной резолюций

Пусть C — дизъюнкт, принадлежащий противоречивому множеству дизъюнктов S, если $S - \{C\}$ выполнимо, то существует линейное опровержение из S с верхним дизъюнктом C.

Доказательство. Будем доказывать теорему индукцией по числу элементов базы A множества дизъюнктов S. Если A состоит из одного элемента Q, то во множестве S должен быть и элемент $\sim Q$. Их резольвентой является дизъюнкт □. Так как $S - \{C\}$ выполнимо, то либо литерал Q, либо литерал $\sim Q$ должен совпадать с литералом C. Теорема для этого случая доказана.

Предположим, что теорема верна для случая, когда A состоит из i элементов при $1 \le i \le n$. Рассмотрим множество A, состоящее из $n+1$ элемента.

Пусть C — единичный дизъюнкт, совпадающий с литералом L. Пусть S имеет вид:

$$S = \{L, A_1 \vee L, A_2 \vee L, \ldots, A_m \vee L, B_1 \vee \sim L, \ldots, B_k \vee \sim L, R_1, \ldots, R_q\}.$$

Дизъюнкты R_1, ..., R_q не содержат ни литерала L, ни литерала $\sim L$. Построим множество S_1 удалением из S дизъюнктов, содержащих L, и вычеркиванием литерала $\sim L$ из остальных дизъюнктов. Множество S_1 имеет вид

$$S_1 = \{B_1, \ldots, B_k, R_1, \ldots, R_q\}.$$

Так как множество S — противоречиво, то и множество дизъюнктов S_1 — противоречиво. Пусть T_1 — противоречивое подмножество S_1 такое, что каждое собственное подмножество T_1 выполнимо. Для того чтобы получить такое подмножество, надо перебрать всевозможные подмножества множества S_1.

Множество T_1 должно содержать дизъюнкт E_1, который был получен из некоторого дизъюнкта вида $B_i \vee \sim L$, принадлежащего множеству дизъюнктов S, до вычеркивания литерала $\sim L$. Если бы это было не так, то все дизъюнкты множества T_1 принадлежали бы множеству дизъюнктов $\{R_1, ..., R_q\}$. По условию теоремы множество дизъюнктов

$$S - \{C\} = \{A_1 \vee L, A_2 \vee L, ..., A_m \vee L, B_1 \vee \sim L, ..., B_k \vee \sim L, R_1, ..., R_q\}$$

выполнимо. Следовательно, существует интерпретация, в которой выполнимо множество дизъюнктов $\{R_1, \ldots, R_q\}$. Но тогда и множество T_1 являлось бы выполнимым, а это противоречит построению множества T_1. Итак, во множестве T_1 есть дизъюнкт E_1, совпадающий с B_i.

Множество дизъюнктов $T_1 - \{E_1\}$ выполнимо, т. к. T_1 строилось таким образом, чтобы каждое его собственное подмножество было выполнимым. Множество T_1 содержит n или менее элементов, по индукционному предположению существует линейное опровержение D_1 из T_1 с верхним дизъюнктом E_1. Преобразуем вывод D_1: возвратим литерал $\sim L$ в те дизъюнкты, из которых он был вычеркнут, за исключением верхнего дизъюнкта E_1. Дизъюнкт E_1 получим в результате выполнения резолюции дизъюнкта L с дизъюнктом $E_1 \vee \sim L$. Таким образом, построили вывод D либо дизъюнкта $\square$, либо дизъюнкта $\sim L$ в S. Если построили вывод дизъюнкта $\sim L$, то строим резольвенту дизъюнктов $\sim L$ и L, получаем линейный вывод дизъюнкта $\square$ из S. Теорема в этом случае доказана.

Предположим, что C — не единичный дизъюнкт. Пусть L — первый (самый левый) литерал в C, т. е. $C = L \vee C_1$, где C_1 — непустой дизъюнкт. Пусть S_1 — множество, получаемое из S удалением дизъюнктов, содержащих дизъюнкт $\sim L$, и вычеркиванием литерала L из всех оставшихся дизъюнктов. Если множество S имеет вид:

$$S = \{L \vee C_1, L \vee C_2, \ldots, L \vee C_m, \sim L \vee B_1, \ldots, \sim L \vee B_k, R_1, \ldots, R_q\}.$$

то множество S_1 будет выглядеть так:

$$S_1 = \{C_1, C_2, \ldots, C_m, R_1, \ldots, R_q\}.$$

Так как множество S — противоречиво, то и множество дизъюнктов S_1 будет противоречивым. Рассмотрим множество

$$S_1 - \{C_1\} = \{C_2, \ldots, C_m, R_1, \ldots, R_q\}.$$

Докажем, что множество $S_1 - \{C_1\}$ выполнимо. Пусть J — интерпретация, в которой выполнимо множество дизъюнктов

$$S - \{C\} = \{L \vee C_2, \ldots, L \vee C_m, \sim L \vee B_1, \ldots, \sim L \vee B_k, R_1, \ldots, R_q\}.$$

Такая интерпретация существует по условию теоремы. Так как S невыполнимо, то дизъюнкт C должен быть ложен в J. Поэтому литерал L ложен в интерпретации J. В интерпретации J истинны дизъюнкты $C_1, C_2, \ldots, C_m, R_1, \ldots, R_q$ следовательно, множество $S_1 - \{C_1\}$ истинно в J, т. е. $S_1 - \{C_1\}$ выполнимо.

Поскольку множество элементов базы S_1 содержит n или менее символов, то по индукционному предположению существует линейный вывод D_1 дизъюнкта □ из S_1 с верхним дизъюнктом C_1. Возвращая литерал L во все дизъюнкты, из которых он ранее был вычеркнут, мы получим линейный вывод D_1 дизъюнкта L из S с верхним дизъюнктом C.

Рассмотрим множество

$$\{L\} \cup (S - \{C\}) = \{L, L \vee C_1, L \vee C_2, \ldots, L \vee C_m, \sim L \vee B_1, \ldots, \sim L \vee B_k, R_1, \ldots, R_q\}.$$

Докажем, что оно противоречиво. Предположим, что это не так, т. е. существует интерпретация I, в которой $\{L\} \cup (S - \{C\})$ выполнимо. В этой интерпретации значение литерала L истинно, но тогда в интерпретации I множество S истинно, что противоречит условию теоремы. Итак, множество дизъюнктов $\{L\} \cup (S - \{C\})$ противоречиво. Множество $S - \{C\}$ выполнимо, L — единичный дизъюнкт. Ранее доказали, что существует линейное опровержение D_2 с верхним дизъюнктом L из множества $\{L\} \cup (S - \{C\})$. Объединяя выводы D_1 и D_2, получим линейное опровержение из S с верхним дизъюнктом C. Теорема доказана.

Задача о поиске острова сокровищ

Предположим, что некоторый искатель приключений хочет добраться до острова сокровищ. Путешественнику предложили три карты: X, Y и Z. Только одна из карт правильная и указывает путь к острову сокровищ. В комнате, где лежали карты, находились пятеро колдунов: A, B, C, D и E. Каждый колдун был либо рыцарем, либо лжецом и каждый дал путешественнику совет:

- E: либо A — лжец, либо C и D — однотипны (либо оба рыцари, либо оба лжеца);
- A: X — правильная карта;
- B: Y — правильная карта;
- C: неверно, что A и B — оба лжеца;
- D: либо A — лжец, либо B — рыцарь.

Какая же из карт правильная?

Докажем, что Y — правильная карта. Будем обозначать через A утверждение "A — рыцарь" (аналогично для B, C, D и E). Обозначим через X утверждение "X — правильная карта" (аналогично для Y и Z). Запишем сначала советы колдунов и предполагаемое следствие формулами исчисления высказываний:

- F_1: $E \Leftrightarrow \sim E \vee (C \Leftrightarrow D)$
- F_2: $A \Leftrightarrow X$
- F_3: $B \Leftrightarrow Y$
- F_4: $C \Leftrightarrow \sim (\sim A \,\&\, \sim B)$
- F_5: $D \Leftrightarrow \sim A \vee B$
- G: Y

Построим КНФ для формулы вида

$$F_1 \,\&\, F_2 \,\&\, F_3 \,\&\, F_4 \,\&\, F_5 \,\&\, \sim G.$$

После преобразований получено множество дизъюнктов S, приведенное в табл. 21.4.

Таблица 21.4. *Исходное множество дизъюнктов*

S_1: $E \vee C \vee D$	S_{11}: $\sim C \vee A \vee B$
S_2: $E \vee \sim C \vee D$	S_{12}: $C \vee \sim A \vee B$
S_3: $\sim E \vee \sim C \vee D$	S_{13}: $C \vee A \vee \sim B$
S_4: $E \vee \sim C \vee \sim D$	S_{14}: $C \vee \sim A \vee \sim B$
S_5: $\sim E \vee C \vee \sim D$	S_{15}: $D \vee A \vee B$
S_6: $E \vee C \vee \sim D$	S_{16}: $\sim D \vee \sim A \vee B$
S_7: $\sim A \vee X$	S_{17}: $D \vee A \vee \sim B$
S_8: $A \vee \sim X$	S_{18}: $D \vee \sim A \vee \sim B$
S_9: $\sim B \vee Y$	S_{19}: $\sim Y$
S_{10}: $B \vee \sim Y$	

В табл. 21.4 выделим те дизъюнкты и литералы, которые будем исключать из множества после преобразования по правилу единичного дизъюнкта (дизъюнкт $\sim Y$). Получаем множество дизъюнктов, размещенных в табл. 21.5.

Таблица 21.5. *Дизъюнкты после сокращения по ~Y*

$E \vee C \vee D$	$\sim C \vee A \vee B$
$E \vee \sim C \vee D$	$C \vee \sim A \vee B$
$\sim E \vee \sim C \vee D$	$C \vee A \vee \sim B$
$E \vee C \vee D$	$\sim C \vee A \vee B$

Таблица 21.5 (окончание)

$E \vee \sim C \vee \sim D$	$C \vee \sim A \vee \sim B$
$\sim E \vee C \vee \sim D$	$D \vee A \vee B$
$E \vee C \vee \sim D$	$\sim D \vee \sim A \vee B$
$\sim A \vee X$	$D \vee A \vee \sim B$
$A \vee \sim X$	$D \vee \sim A \vee \sim B$
$\sim B$	

После преобразования по правилу единичного дизъюнкта (дизъюнкт $\sim B$) получим множество дизъюнктов, размещенных в табл. 21.6.

Таблица 21.6. *Дизъюнкты после сокращения по ~B*

S_1: $E \vee C \vee D$	S_7: $\sim A \vee X$
S_2: $E \vee \sim C \vee D$	S_8: $A \vee \sim X$,
S_3: $\sim E \vee \sim C \vee D$	S_9: $C \vee \sim A$
S_4: $E \vee \sim C \vee \sim D$	S_{10}: $\sim C \vee A$
S_5: $\sim E \vee C \vee \sim D$	S_{11}: $\sim D \vee \sim A$
S_6: $E \vee C \vee \sim D$	S_{12}: $D \vee A$

Линейное опровержение, построенное из последнего множества дизъюнктов, может быть таким:

$$< S_1\colon E \vee C \vee D,\ S_2\colon E \vee \sim C \vee D > \rightarrow S_{13}\colon E \vee D;$$

$$< S_{13}\colon E \vee D,\ S_4\colon E \vee \sim C \vee \sim D > \rightarrow S_{14}\colon E \vee C;$$

$$< S_{14}\colon E \vee C,\ S_5\colon \sim E \vee C \vee \sim D > \rightarrow S_{15}\colon C \vee \sim D;$$

$$< S_{15}\colon C \vee \sim D,\ S_6\colon E \vee C \vee \sim D > \rightarrow S_{16}\colon E \vee \sim D;$$

$$< S_{16}\colon E \vee \sim D,\ S_{13}\colon E \vee D > \rightarrow S_{17}\colon E;$$

$$< S_{17}\colon E,\ S_3\colon \sim E \vee \sim C \vee D > \rightarrow S_{18}\colon \sim C \vee D;$$

$$< S_{18}\colon \sim C \vee D,\ S_9\colon C \vee \sim A > \rightarrow S_{19}\colon \sim A \vee D;$$

$$< S_{19}\colon \sim A \vee D,\ S_{12}\colon D \vee A > \rightarrow S_{20}\colon D;$$

$$< S_{20}\colon D,\ S_5\colon \sim E \vee C \vee \sim D > \rightarrow S_{21}\colon \sim E \vee C;$$

$$< S_{21}\colon \sim \mathrm{E} \vee \mathrm{C},\ \mathrm{S}_{17}\colon \mathrm{E} > \rightarrow S_{22}\colon C;$$

$$< S_{22}\colon C,\ S_{10}\colon \sim C \vee A > \rightarrow S_{23}\colon A;$$

$$< S_{23}\colon A,\ S_{11}\colon \sim D \vee \sim A > \rightarrow S_{24}\colon \sim D;$$

$$< S_{24}\colon \sim D,\ S_{20}\colon D > \rightarrow \square.$$

Упражнения

1. Определите, какие дизъюнкты множества дизъюнктов S в задаче об автомобилях могут быть использованы в качестве верхних дизъюнктов в линейном выводе.
2. Определите, какие дизъюнкты множества дизъюнктов S в задаче о кандидатах в министры могут быть использованы в качестве верхних дизъюнктов в линейном выводе.

Задания по программированию

3. Напишите сценарий, который по заданному множеству дизъюнктов S определяет те дизъюнкты, которые могут быть использованы в качестве верхних дизъюнктов в линейном выводе.
4. Напишите сценарий, который по заданному множеству дизъюнктов S и некоторому дизъюнкту из S определяет, можно ли построить линейный вывод с этим верхним дизъюнктом, и строит вывод, если это возможно.

Семантическая резолюция

При построении опровержения можно придерживаться определенной стратегии. Каждая из стратегий имеет свои достоинства и недостатки. Один из основных вопросов стратегии состоит в том, полна ли она, т. е. всегда ли можно построить опровержение из противоречивого множества дизъюнктов согласно выбранной стратегии.

При поиске доказательства порождается большое число дизъюнктов, которые в дальнейшем могут не использоваться. Для того чтобы сократить множество порождаемых дизъюнктов, исходное множество S разбивается на два множества Σ_1 и Σ_2. При построении вывода не разрешается строить резольвенту по двум дизъюнктам, принадлежащим одному множеству. При разбиении множества S на два множества Σ_1 и Σ_2 будет использована произвольная интерпретация I.

Пусть S — противоречивое множество дизъюнктов, I — произвольная интерпретация. Во множество Σ_1 включим все дизъюнкты из S, истинные в интерпретации I, а во множество Σ_2 — все дизъюнкты, ложные в I.

Пример разбиения множества дизъюнктов

При решении задачи о кандидатах в министры было построено следующее множество дизъюнктов:

$$S = \{\sim E \vee \sim F \vee \sim G, \sim E \vee \sim F \vee G, \sim E \vee F \vee \sim G, E \vee \sim F \vee \sim G,$$
$$E \vee \sim F \vee G, E \vee F \vee \sim G, E \vee F \vee G, \sim E\}.$$

Рассмотрим интерпретацию $I = \{E, \sim F, G\}$. В этом случае множества Σ_1 и Σ_2 будут следующими:

- $\Sigma_1 = \{\sim E \vee \sim F \vee \sim G, \sim E \vee \sim F \vee G, E \vee \sim F \vee \sim G, E \vee \sim F \vee G, E \vee F \vee \sim G, E \vee F \vee G\}$
- $\Sigma_2 = \{\sim E \vee F \vee \sim G, \sim E\}$

Интерпретация $I = \{\sim E, F, G\}$ разобьет исходное множество S на следующие два множества

- $\Sigma_1 = \{\sim E \vee \sim F \vee \sim G, \sim E \vee \sim F \vee G, \sim E \vee F \vee \sim G, E \vee \sim F \vee G, E \vee F \vee \sim G, E \vee F \vee G, \sim E\}$
- $\Sigma_2 = \{E \vee \sim F \vee \sim G\}$

Лемма. Пусть S — противоречивое множество дизъюнктов, I — произвольная интерпретация. Во множество Σ_1 включим все дизъюнкты из S, истинные в интерпретации I, а во множество Σ_2 — все дизъюнкты, ложные в I. Доказать, что Σ_1 и Σ_2 — непустые множества.

Предположим сначала, что множество Σ_2 пусто, т. е. множество S совпадает с множеством Σ_1. Мы нашли интерпретацию, в которой S выполнимо, а мы рассматриваем противоречивое множество S. Следовательно, множество Σ_2 не может быть пустым.

Предположим теперь, что множество Σ_1 пусто, т. е. мы нашли интерпретацию, в которой все дизъюнкты из S ложны. Но тогда мы должны признать, что в дизъюнктах S_j для $j = 1, \ldots, n$ нет ни одной контрарной пары. Если бы это было не так, то во множестве дизъюнктов S была бы пара дизъюнктов S_{i1} и S_{i2} вида:

$$S_{i1} \leftrightarrow A_{i1} \vee L \vee B_{i1} \quad \text{и} \quad S_{i2} \leftrightarrow A_{i2} \vee \sim L \vee B_{i2}.$$

В выбранной интерпретации I значения дизъюнктов S_{i1} и S_{i2} отличались бы: одно из них должно быть истинным, другое ложным. Мы же предположили, что все дизъюнкты ложны. Следовательно, во множестве S нет контрарных пар. По интерпретации $I = \{m_1, m_2, \ldots, m_k\}$ построим интерпретацию $I^* = \{\sim m_1, \sim m_2, \ldots, \sim m_k\}$. В интерпретации I^* все дизъюнкты из S истинны; как и в первом случае, мы нашли интерпретацию, в которой S выполнимо, а этого не может быть, т. к. S — противоречиво. Следовательно, предположение, что Σ_1 пусто, неверно. Лемма доказана.

С целью дальнейшего сокращения числа порождаемых дизъюнктов на множестве элементов базы задается отношение порядка. При построении резольвенты по дизъюнктам S_1 и S_2 удаляемый литерал в S_1 — наибольший в этом дизъюнкте.

Пример построения резольвент

Рассмотрим следующее множество дизъюнктов:

$$S = \{\sim E \vee \sim F \vee \sim G, \sim E \vee \sim F \vee G, \sim E \vee F \vee \sim G, E \vee \sim F \vee \sim G,$$
$$E \vee \sim F \vee G, E \vee F \vee \sim G, E \vee F \vee G, \sim E\}.$$

Если в качестве одного из дизъюнктов взять дизъюнкт S_1, то можно построить следующие резольвенты:

- $\sim F \vee \sim G$ резольвента S_1 и S_4
- $\sim E \vee \sim G$ резольвента S_1 и S_3
- $\sim E \vee \sim F$ резольвента S_1 и S_2

Если задан порядок $E > F > G$, то допустимой является первая резольвента, если порядок $F > E > G$, то следует строить вторую резольвенту, и, наконец, если порядок определен $G > E > F$, то должна быть построена третья резольвента.

Определение семантического конфликта

Рассмотрим множество дизъюнктов S, выберем некоторую интерпретацию I, на элементах базы S зададим порядок P.

Конечное множество дизъюнктов вида $\{E_1, E_2,, E_q, N\}$, где $q \geq 1$ называется *семантическим конфликтом*, если E_1, E_2, ..., E_q и N удовлетворяют следующим условиям:

1. Дизъюнкты E_1, E_2, ..., E_q ложны в интерпретации I.
2. Пусть $R_1 = N$, для каждого $I = 1, ..., q$ имеется резольвента R_{I+1} дизъюнктов R_i и E_I (рис. 21.12).
3. Тот литерал в E_i, по которому строится резольвента, наибольший в E_i.
4. Дизъюнкт R_{q+1} ложен в интерпретации I.

Дизъюнкт R_{q+1} называется семантической или *PI*-резольвентой, обозначать семантическую резольвенту будем следующим образом:

$$\{E_1, E_2, ..., E_q, N\} \rightarrow R_{q+1}.$$

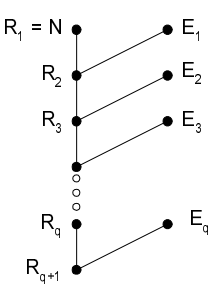


Рис. 21.12. Схема построения семантической резольвенты

Пример построения семантического конфликта

Рассмотрим множество дизъюнктов, построенное при решении задачи об автомобилях:

$$S = \{R \vee F, \sim R \vee \sim F, G \vee M, \sim G \vee \sim M, G \vee A, \sim G \vee \sim A, R \vee G,$$
$$\sim R \vee \sim G, \sim F \vee \sim M \vee \sim A, \sim F \vee \sim M \vee A, \sim F \vee M \vee \sim A,$$
$$F \vee \sim M \vee \sim A, F \vee M \vee A, \sim F \vee \sim G\}.$$

Выберем следующую интерпретацию $I = \{R, F, G, \sim M, A\}$, на множестве элементов базы зададим порядок $R > G > F > M > A$.

Множество дизъюнктов $\{\sim F \vee \sim G, \sim R \vee \sim F, R \vee G\}$ является семантическим конфликтом (рис. 21.13). Дизъюнкт $\sim F$ является построенной по семантическому конфликту резольвентой

$$\{\sim F \vee \sim G, \sim R \vee \sim F, R \vee G\} \rightarrow \sim F.$$

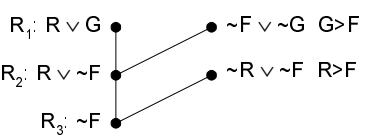


Рис. 21.13. Построение семантического конфликта

Пусть I — интерпретация некоторого множества дизъюнктов S; P — порядок на элементах базы. Вывод из множества S называется *семантическим* или *PI-выводом*, если каждый дизъюнкт вывода либо принадлежит S, либо является PI-резольвентой.

PI-вывод дизъюнкта $\square$ называется *PI-опровержением*.

Об Олимпиаде

В Олимпиаде по математике участвовали три ученика (обозначим их A, B, C). Когда A подошел к учителю узнать об итогах, тот ему сказал: "Если победитель ты, а не B, то и C также победитель". Второй ученик получил следующий ответ: "Если C — победитель, то либо ты, либо A также победители". Встретив ученика C, учитель произнес: "Если A — победитель, то ты точно не победитель". Чтобы утешить своих учеников, преподаватель на прощание сказал: "Среди вас точно есть победитель". Ученики предположили, что преподаватель не шутит и все его высказывания истинны. Является ли ученик B победителем?

Обозначим через A утверждение "A — победитель Олимпиады" (соответственно для B и C). Утверждения учителя и предполагаемое следствие запишутся формулами:

- F_1: $A \,\&\, \sim B \Rightarrow C$
- F_2: $C \Rightarrow A \vee B$
- F_3: $A \Rightarrow \sim C$
- F_4: $A \vee B \vee C$
- G: B

Построим множество дизъюнктов S:

$$S = \{\sim A \vee B \vee C,\ A \vee B \vee \sim C,\ \sim A \vee \sim C,\ A \vee B \vee C,\ \sim B\}.$$

Выберем интерпретацию $I = \{\sim A,\ B,\ \sim C\}$ и определим порядок $A > B > C$. Интерпретация I разбивает исходное множество S на два:

$$\Sigma_1 = \{\sim A \vee B \vee C,\ A \vee B \vee \sim C,\ \sim A \vee \sim C,\ A \vee B \vee C,\};\quad \Sigma_2 = \{\sim B\}.$$

Приведем семантическое опровержение, указывая семантический конфликт и построенную по нему PI-резольвенту:

$$\{\sim B,\ A \vee B \vee C\} \to A \vee C;$$

$$\{A \vee C,\ \sim B,\ \sim A \vee B \vee C\} \to C;$$

$$\{C,\ \sim B,\ A \vee B \vee \sim C\} \to A;$$

$$\{A,\ C,\ \sim A \vee \sim C\} \to \square.$$

На рис. 21.14 изображено дерево, соответствующее построенному PI-опровержению.

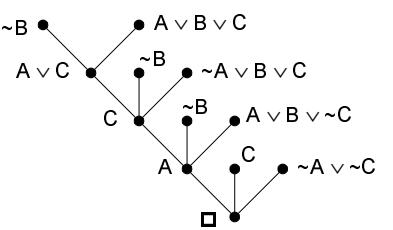


Рис. 21.14. Семантическое опровержение для задачи об Олимпиаде

Теорема о полноте метода семантической резолюции

Пусть S — противоречивое множество дизъюнктов; I — произвольная интерпретация множества S; P —упорядочение на множестве элементов базы A. Существует PI-опровержение в S.

Доказательство. Пусть множество A состоит лишь из одного элемента Q. Так как множество дизъюнктов S противоречиво, то среди дизъюнктов S есть дизъюнкты Q и $\sim Q$. Поскольку один из дизъюнктов Q или $\sim Q$ ложен в I, то можно построить PI-резольвенту дизъюнкта $\square$. Для этого случая теорема доказана.

Допустим, что теорема верна в тех случаях, когда множество A состоит из i элементов, где $1 \le i \le n$. Рассмотрим случай, когда множество A состоит из $n+1$ элемента. При доказательстве исследуем два варианта.

Предположим, что множество S содержит единичный дизъюнкт L, ложный в интерпретации I. Построим множество S^* следующим образом: исключим из множества S все дизъюнкты, содержащие литерал L, а из оставшихся дизъюнктов вычеркнем литерал $\sim L$. Так как S противоречиво, то противоречивым будет и построенное множество S^*. Множество элементов базы для S^* содержит менее $n+1$ элемента, поэтому по индукционному предположению существует PI-опровержение D^* в S^*.

При построении PI-опровержения в S будем исследовать каждый дизъюнкт PI-вывода D^* в S^*. Предположим, что дизъюнкт D_j в D^* получен из семантического конфликта вида:

$$< E_1^*, E_2^*, ..., E_q^*, L, N >.$$

Если дизъюнкт N^* был получен из дизъюнкта N вычеркиванием литерала $\sim L$, то будем рассматривать семантический конфликт

$$< E_1^*, E_2^*, ..., E_q^*, L, N >,$$

который порождает тот же дизъюнкт. Если дизъюнкт E^* получен из дизъюнкта E_i вычеркиванием литерала $\sim L$, то будем рассматривать его как PI-резольвенту семантического конфликта $<L, E_i>$. После описанных преобразований мы получим PI-вывод дизъюнкта D_j из множества S. Выполняя преобразования над каждым дизъюнктом PI-опровержения D^* в S^*, построим PI-опровержение D в S.

Пример построения семантического опровержения: вариант 1

Рассмотрим следующее множество дизъюнктов:

$$S = \{\sim A \vee B, \sim B \vee C \vee \sim A, D \vee A, D \vee \sim C, \sim D \vee A, \sim D\}.$$

База множества S состоит из элементов $\{A, B, C, D\}$. Зададим порядок P: $A > B > C > D$.

Возьмем интерпретацию $I = \{A, \sim B, C, D\}$. Множество S содержит единичный дизъюнкт $\sim D$ ложный в I. Построим множество

$$S^* = \{\sim A \vee B, \sim B \vee C \vee \sim A, A, \sim C\}.$$

База множества S^* содержит три элемента. Построим PI-вывод D^* дизъюнкта $\square$ из S^*:

$$<\sim A \vee B, A> \rightarrow B;$$

$$<B, \sim C, \sim A \vee \sim B \vee C> \rightarrow \sim A;$$

$$<\sim A, A> \rightarrow \square.$$

На рис. 21.15 изображено дерево, соответствующее PI-опровержению D^* в S^*. Исследуя каждый дизъюнкт построенного вывода, построим по описанным правилам PI-опровержения D в S:

$$<\sim A \vee B, \sim D, D \vee A> \rightarrow B$$

$$<\sim D, D \vee \sim C> \rightarrow \sim C$$

$$<B, \sim C, \sim A \vee \sim B \vee C> \rightarrow \sim A$$

$$<\sim A, \sim D, D \vee A> \rightarrow \square.$$

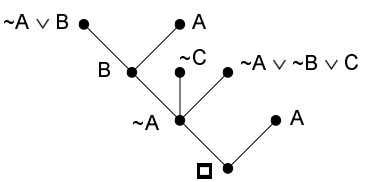


Рис. 21.15. Построение семантического опровержения для S^*

На рис. 21.16 изображено дерево, соответствующее PI-опровержению D в S. В нем содержатся те элементы, которые были добавлены при исследовании вывода D^*.

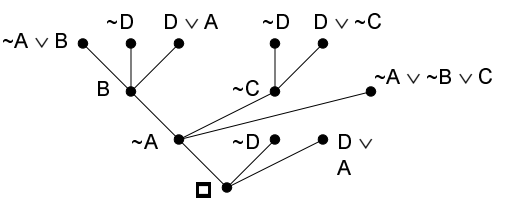


Рис. 21.16. Восстановленное семантическое опровержение для S

Теперь предположим, что множество S не содержит единичного дизъюнкта, ложного в I. Выберем наименьший элемент Z и поступим следующим образом: в качестве L выберем либо Z, либо $\sim Z$, т. е. тот литерал, который ложен в I. Множество S^* строим следующим образом: вычеркиваем все дизъюнкты, содержащие $\sim L$, из оставшихся дизъюнктов вычеркиваем дизъюнкт L. Так как S — противоречиво, то противоречивым будет и построенное

множество S^*. База множества S^* содержит не более, чем n элементов, поэтому по индукционному предположению существует PI-опровержение D_1 в S^*. Пусть D_2 — вывод, полученный из D_1 возвращением литерала L в те дизъюнкты, из которых он был вычеркнут. Вывод D_2 также является семантическим выводом, т. к. L — минимальный литерал, ложный в I. Семантический вывод D_2 является выводом в S дизъюнкта $\square$ или дизъюнкта L. В первом случае теорема доказана.

Рассмотрим случай, когда D_2 является семантическим выводом в S дизъюнкта L: $Z_1, Z_2, ..., L$. Исследуем множество дизъюнктов $S \cup \{L\}$. Это множество содержит единичный дизъюнкт, ложный в интерпретации I. По доказанному ранее (вариант 1) существует семантический вывод D_3 дизъюнкта $\square$ из множества $S \cup \{L\}$: $T_1, T_2, ..., T_m \leftrightarrow \square$.

Выбираем наименьшее j, такое, что T_j совпадает с L. Рассматриваем вывод L в S: $Z_1, Z_2, ..., L$. Так преобразованный вывод D_3 будет PI-выводом $\square$ из множества дизъюнктов S. Теорема доказана.

Пример построения семантического опровержения: вариант 2

Рассмотрим множество дизъюнктов:

$$S = \{\sim A \vee B \vee C, \sim C \vee A \vee B, \sim A \vee \sim C, A \vee B \vee C, \sim B\}.$$

Выберем интерпретацию $I = \{\sim A, \sim B, C\}$, определим порядок $C > B > A$. Рассматриваемое множество S не содержит единичного дизъюнкта ложного в I. Выберем наименьший элемент A, он ложен в I. Множество S^* строим следующим образом: вычеркиваем все дизъюнкты, содержащие $\sim A$, из оставшихся дизъюнктов вычеркиваем дизъюнкт A:

$$S^* = \{\sim C \vee B, B \vee C, \sim B\}.$$

Так как S противоречиво, то противоречивым будет и построенное множество S^*. Семантическое опровержение D_1 в S^*:

$$<B \vee \sim C, B \vee C> \rightarrow B;$$

$$<B, \sim B> \rightarrow \square.$$

При построении вывода D_2 возвращаем литерал A в те дизъюнкты, из которых он был вычеркнут:

$$<A \vee B \vee \sim C, A \vee B \vee C> \rightarrow A \vee B;$$

$$<A \vee B, \sim B> \rightarrow A.$$

Рассмотрим множество $S \cup \{A\}$, в нем есть единичный дизъюнкт A, ложный в интерпретации I. Следовательно, можно построить PI-опровержение D_3 в $S \cup \{A\}$:

$$<A, \sim A \vee \sim C> \rightarrow \sim C.$$

$$<\sim C, A, B \vee C \vee \sim A> \rightarrow B;$$

$$<B, \sim B> \rightarrow \square.$$

Преобразовывая PI-опровержение D_3, построим PI-вывод дизъюнкта □ из множества дизъюнктов S:

$$<A \vee B \vee \sim C, A \vee B \vee C> \rightarrow A \vee B;$$

$$<A \vee B, \sim B> \rightarrow A;$$

$$<A, \sim A \vee \sim C> \rightarrow \sim C;$$

$$<\sim C, A, B \vee C \vee \sim A> \rightarrow B;$$

$$<B, \sim B> \rightarrow \square.$$

Упражнения

1. Постройте семантическое опровержение для задачи о формировании экипажа при заданной интерпретации $I = \{\sim C, M, \sim D\}$.
2. Постройте семантическое опровержение для задачи о поиске острова сокровищ при определенном порядке P: $C > A > B > D > E > X > Y > Z$.
3. Постройте семантическое опровержение для задачи об автомобилях.

Задания по программированию

1. Напишите сценарий, который по заданному множеству дизъюнктов S и интерпретации строит два множества дизъюнктов: в первое множество включаются все дизъюнкты, истинные в заданной интерпретации, во второе — ложные.
2. Напишите сценарий, который по заданному множеству дизъюнктов S, интерпретации I и порядку P строит семантическое опровержение в S.

Положительная гиперрезолюция

Рассмотрим некоторые частные случаи семантической резолюции.

Дизъюнкт называется положительным, если его литералы не содержат знаков отрицания, например, следующие дизъюнкты положительны:

$$A, A \vee C, B \vee C \vee A.$$

Дизъюнкт называется отрицательным, если все его литералы содержат знаки отрицания, например, следующие дизъюнкты отрицательны:

$$\sim A, \sim A \vee \sim C, \sim B \vee \sim C \vee \sim A.$$

Дизъюнкт называется смешанным, если он не является ни положительным, ни отрицательным, например:

$$\sim A \vee C, B \vee C \vee \sim A.$$

Положительной гиперрезолюцией называется семантическая (PI-резолюция), в которой любой литерал интерпретации содержит знак отрицания. Гиперрезолюция называется положительной, потому что все E_I и PI-резольвенты при такой интерпретации положительные дизъюнкты. Положительная гиперрезолюция является частным случаем семантической резолюции.

Расследование хищения

Рассмотрим следующую задачу. При расследовании дела о хищении установлено:

- никто, кроме A, B, C, в деле не замешан;
- A не идет на дело без, по крайней мере, одного соучастника;
- C невиновен.

Спрашивается, виновен ли B?

Запишем известные факты и предполагаемое следствие с помощью формул исчисления высказываний:

- F_1: $A \vee B \vee C$
- F_2: $A \Rightarrow B \vee C$
- F_3: $\sim C$
- G: B

Построенное множество дизъюнктов будет иметь вид:

$$S = \{A \vee B \vee C, \sim A \vee B \vee C, \sim C, \sim B\}.$$

Зададим порядок P: $A > B > C$. Рассмотрим интерпретацию $I = \{\sim A, \sim B, \sim C\}$. Исходное множество S разбивается на два, Σ_1 и Σ_2, следующим образом:

$$\Sigma_1 = \{\sim A \vee B \vee C, \sim C, \sim B\};$$

$$\Sigma_2 = \{A \vee B \vee C\}.$$

Построим вывод дизъюнкта $\square$ из S:

$$<A \vee B \vee C, \sim A \vee B \vee C> \rightarrow B \vee C;$$

$$<B \vee C, \sim B> \rightarrow \sim C;$$

$$<C, \sim C> \rightarrow \square.$$

Вывод является положительной гиперрезолюцией, все E_I в семантическом конфликте положительны, любая семантическая резольвента также положительна.

Дерево вывода для положительной гиперрезолюции представлено на рис. 21.17. Положительная резолюция соответствует построению вывода, начиная с аксиом.

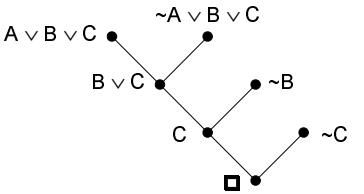


Рис. 21.17. Положительная гиперрезолюция для задачи о расследовании

Если же на множестве элементов задан порядок P: $C > A > B$, то PI-опровержение будет следующим:

$$<A \vee B \vee C, \sim C> \rightarrow A \vee B;$$

$$<A \vee B, \sim A \vee B \vee C> \rightarrow B \vee C;$$

$$<B \vee C, \sim C> \rightarrow B;$$

$$<B, \sim B> \rightarrow \square.$$

Упражнение

Постройте семантическое опровержение, являющееся положительной гиперрезолюцией, для задач:

- о формировании экипажа;
- о поиске острова сокровищ;
- об автомобилях.

Задания по программированию

1. Напишите сценарий, который по заданному множеству дизъюнктов S строит семантическое опровержение, являющееся положительной гиперрезолюцией.
2. Напишите сценарий, который по заданному множеству дизъюнктов S и семантическому выводу определяет, является ли вывод положительной гиперрезолюцией.

Отрицательная гиперрезолюция

Отрицательной гиперрезолюцией называется семантическая (PI-резолюция), в которой любой литерал интерпретации не содержит знак отрицания. Гиперрезолюция называется отрицательной, потому что все E_I и PI-резольвенты при такой интерпретации — отрицательные дизъюнкты. Отрицательная гиперрезолюция является частным случаем семантической резолюции.

Отрицательная гиперрезолюция для задачи о расследовании

В предыдущем примере исследовалось множество дизъюнктов:

$$S = \{A \vee B \vee C, \sim A \vee B \vee C, \sim C, \sim B\}.$$

Зададим порядок P: $A > B > C$. Рассмотрим интерпретацию $I = \{A, B, C\}$. Исходное множество S разбивается на два, Σ_1 и Σ_2, следующим образом:

$$\Sigma_1 = \{A \vee B \vee C, \sim A \vee B \vee C\};$$

$$\Sigma_2 = \{\sim C, \sim B\}.$$

Построим вывод дизъюнкта $\square$ из S, который в данном случае будет отрицательной гиперрезолюцией:

$$<\sim B, \sim C, \sim A \vee B \vee C > \rightarrow \sim A;$$

$$<\sim A, \sim B, \sim C, A \vee B \vee C > \rightarrow \square.$$

Дерево вывода для отрицательной гиперрезолюции представлено на рис. 21.18 и, как видно, вывод строится с отрицания заключения. Допустив, что заключение ложно, выводится противоречие из этого допущения.

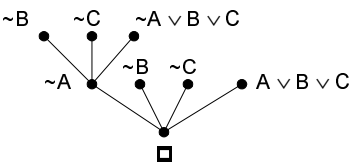


Рис. 21.18. Отрицательная гиперрезолюция для задачи о расследовании

Упражнение

Постройте семантическое опровержение, являющееся отрицательной гиперрезолюцией, для задач:

- о формировании экипажа;
- о поиске острова сокровищ;
- об автомобилях.

Задания по программированию

1. Напишите сценарий, который по заданному множеству дизъюнктов S строит семантическое опровержение, являющееся отрицательной гиперрезолюцией.

2. Напишите сценарий, который по заданному множеству дизъюнктов S и семантическому выводу, определяет, является ли вывод отрицательной гиперрезолюцией.

Стратегия поддержки

Как и рассмотренные стратегии, стратегия поддержки направлена на то, чтобы не порождать лишние дизъюнкты.

Рассмотрим множество дизъюнктов S. Подмножество T множества S называется *множеством поддержки*, если множество $S–T$ выполнимо.

Резолюцией с поддержкой называется резолюция, примененная к двум дизъюнктам, не принадлежащим одновременно множеству $S–T$.

Вывод с поддержкой — это вывод, в котором любая резолюция — это резолюция с поддержкой.

Опровержение с поддержкой — это вывод с поддержкой дизъюнкта $\square$.

Пример использования

Рассмотрим следующее множество дизъюнктов:

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

В качестве множества T возьмем множество $T = \{\sim M \vee D, \sim D \vee C\}$. Множество $S– T$ состоит из дизъюнктов

$$S– T = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim D\}.$$

Оно выполнимо, для того чтобы убедиться в этом, достаточно выбрать интерпретацию, в которой литерал M истинен, а литерал D ложен. Множество T является множеством поддержки для S. Следующий вывод представляет собой опровержение с поддержкой:

$$< C \vee M \vee D, \sim M \vee D > \rightarrow C \vee D;$$

$$<\sim C \vee M \vee D, \sim M \vee D > \rightarrow \sim C \vee D;$$

$$< C \vee D, \sim C \vee D > \rightarrow D;$$

$$<\sim D, D > \rightarrow \square.$$

Теорема о полноте метода поддержки

Пусть S — противоречивое множество дизъюнктов, T — подмножество S, такое, что $S–T$ выполнимо. Существует вывод с поддержкой дизъюнкта $\square$ из множества S, в котором T является множеством поддержки.

Доказательство. Так как множество $S–T$ выполнимо, то существует интерпретация I, в которой оно истинно. Будем рассматривать эту интерпретацию. Выберем любое упорядочивание P для элементов базы. По теореме о полноте метода семантической резолюции существует PI-вывод D дизъюнкта $\square$ из множества S. Рассмотрим произвольный семантический конфликт, порождающий семантическую резольвенту вывода D: $<E_1, E_2, \ldots, E_q, N>$ для $q \geq 1$. Все дизъюнкты E_j ложны в интерпретации I, поэтому в семантическом конфликте при построении резольвенты $R_j + 1$ по дизъюнктам R_j и E_j оба дизъюнкта не принадлежат одновременно множеству $S–T$. Следовательно, по любому семантическому конфликту может быть построен вывод с поддержкой. Теорема доказана.

Применение теоремы в задаче формировании экипажа

В задаче о формировании экипажа было построено и исследовалось следующее множество дизъюнктов:

$$S = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D, \sim M \vee D, \sim D \vee C, \sim D\}.$$

Рассмотрим множество $T = \{\sim M \vee D, \sim D \vee C, \sim D\}$. Для того чтобы убедиться, что оно является множеством поддержки, достаточно показать, что множество

$$S–T = \{C \vee M \vee D, \sim C \vee M \vee D, M \vee \sim D\}$$

выполнимо. Все дизъюнкты множества $S–T$ истинны в любой интерпретации, в которой истинен литерал M. Рассмотрим интерпретацию $I = \{C, M, D\}$, зададим порядок $C > M > D$. Мы можем построить следующий семантический вывод дизъюнкта $\square$ из множества S:

$$<\sim D, \sim M \vee D> \rightarrow \sim M;$$

$$<\sim M, \sim D, \sim C \vee M \vee D> \rightarrow \sim C;$$

$$<\sim C, \sim M, \sim D, C \vee M \vee D> \rightarrow \square.$$

По только что построенному семантическому выводу может быть построен следующий вывод с поддержкой:

$$<\sim D, \sim M \vee D> \rightarrow \sim M;$$

$$<\sim M, \sim C \vee M \vee D> \rightarrow \sim C \vee D;$$

$$<\sim D, \sim C \vee D> \rightarrow \sim C;$$

$$<\sim C, C \vee M \vee D> \rightarrow M \vee D;$$

$$<\sim M, M \vee D> \rightarrow D;$$

$$<\sim D, D> \rightarrow \square.$$

Упражнение

Постройте вывод с поддержкой для задач:

- о поиске острова сокровищ;
- об автомобилях.

Задания по программированию

1. Напишите сценарий, который по заданному множеству дизъюнктов S строит вывод с поддержкой.
2. Напишите сценарий, который по заданному множеству дизъюнктов S и выводу определяет, является ли вывод выводом с поддержкой.

Заключение

Приведенные в книге примеры демонстрируют широкие возможности языка JavaScript. Но многие вопросы, связанные с созданием и использованием сценариев JavaScript, остались не рассмотренными, а задачи не решенными. Однако первые шаги сделаны, и далее читатель может писать собственные сценарии и приступать к самостоятельной разработке Web-приложений.

Сайт разработчиков JavaScript **http://developer.netscape.com/library/documentation/** содержит информацию, которая будет полезна при создании программ.

Для дальнейшего изучения методов современного программирования рекомендуется обратиться к литературе [1, 2, 10, 12].

Список литературы

1. Бранденбау Дж. JavaScript: сборник рецептов для професионалов: Пер. с англ. — СПб.: Питер, 2000.
2. Вайк А., Вагнер Р. JavaScript в примерах: Пер. с англ. — Киев: Диа Софт, 2000.
3. Дмитриева М. В., Кубенский А. А. Турбо Паскаль и Турбо Си: построение и обработка структур данных. — СПб.: Изд-во СПбГУ, 1996.
4. Дмитриева М. В., Павлова М. В. Система автоматизации процесса решения задач//Компьютерные инструменты в образовании. № 2, 1998, с. 32—37.
5. Дмитриева М. В., Павлова М. В. Система автоматизации процесса решения задач: построение доказательства//Компьютерные инструменты в образовании. № 3—4, 1998, с. 62—67.
6. Дмитриева М. В., Павлова М. В. Система автоматизации процесса решения задач: описание системы и примеры ее использования//Компьютерные инструменты в образовании. № 5, 1998, с. 53—58.
7. Кирова Е. В. Компьютерные технологии для пользователя ЭВМ. — Коломна: 1999.
8. Клини С. Математическая логика: Пер. с англ. — М.: Наука, 1973.
9. Косовский Н. К. Элементы математической логики и ее применение к теории субрекурсивных алгоритмов. — Л.: Изд-во ЛГУ, 1981.
10. Матросов А., Сергеев А., Чаунин М. HTML 4.0. — СПб.: БХВ—Санкт-Петербург, 1999.
11. Мендельсон Э. Введение в математическую логику. — М.: Наука, 1984.
12. Морис Б. HTML в действии: Пер. с англ. — СПб.: Питер, 1997.
13. Нильсон Н. Дж. Принципы искусственного интеллекта: Пер. с англ. — М.: Мир, 1985.
14. Рик Дарнел. JavaScript. Справочник: Пер. с англ. — СПб.: Питер, 2000.
15. Смаллиан Р. Как же называется эта книга? — М.: Мир, 1981.
16. Чень Ч., Ли Р. Математическая логика и автоматическое доказательство теорем: Пер. с англ. — М.: Наука, 1983.
17. Шауцукова Л. З. Информатика. — М.: Просвещение, 1997.

Предметный указатель

Н

О

П

Р

С

Т

Ф

Ч