Задание №1.

Цель: Конструирование пользовательского типа данных – структур, создание массива структур в динамической памяти, ввод исходных данных из файла, запись результатов работы в файл.

Задание:

Создать в текстовом редакторе файл с исходными данными в виде строк. Количество строк, их содержимое и способ обработки определяется вашим вариантом.

В соответствии со своим вариантом создать структуру и на ее основе создать массив в динамической памяти.

Ваша программа должна выполнять следующие действия:

- Читать исходные данные из файла в динамический массив
- Иметь возможность просмотреть данные, хранящиеся в массиве.
- Откорректировать данные конкретной заданной строки.
- Выполнить заданные вычисления.
- Сохранить данные в новом файле. Имя файла вводится с клавиатуры.

Вариант задания

В текстовом файле с исходными данными находится таблица, состоящая из n строк. В каждой строке по m слов, образующих соответственно по m столбцов. Между словами расстояние – один пробел. Тип данных в каждом столбце должен соответствовать заданию.

Вариант:

Количество строк: 4. Столбцы: Номер банковской карты, Фамилия владельца, Год окончания действия, Остаток на счете. Определить владельца карты с минимальным остатком средств.

Исходные данные хранятся в файле “myfile1.txt”
Результирующие данные записываем в файл “myfile2.txt”

Задание №2

Конструирование простейшего класса

Цель: познакомиться с основными понятиями объектно – ориентированного программирования (касс, объект, свойство, метод, конструктор, деструктор, полиморфизм, инкапсуляция), выполнить конструирование класса, предназначенного для хранения заданной структуры данных, изучить способы создания объектов.

Задание:
В работе требуется сконструировать класс с заданным набором свойств. Набор свойств следует взять из задания №1 в соответствии со своим вариантом. В класс также должны быть добавлено остаточное количество методов для просмотра редактирования значений любого из свойств.
Требования к конструированию класса: доступ к свойствам – закрытый, к методам – открытый. В классе следует предусмотреть конструктор по умолчанию, конструктор с параметрами.

Действия, выполняемые программой:

- Создание объекта с помощью конструктора по умолчанию.
- Создание объекта с помощью конструктора с параметрами.
- Создание массива объектов.
- Редактирование и просмотр свойств каждого объекта (можно однократное).
- Обработка массива объектов в соответствии с заданием №1.

Ввод исходных данных осуществляется с клавиатуры, вывод на экран.

Теоретический материал, необходимый для решения задания №1:

1 Структуры.
2. Динамическое выделение памяти.
3. Управление потоками данных. Стандартные потоки и файловые потоки. (Работа с файлами)

Справочный материал

Структура

Шаблон	Структурная переменная

Структура – это тип данных, задаваемый пользователем.
Включает в себя:

Задание шаблона структуры . Собственно описание структурной переменной.

- *Шаблон* (pattern)– правила формирования структурной переменной.

Задание шаблона *не связано непосредственно с резервированием памяти* . Шаблон дает компилятору всю необходимую информацию о полях структурной переменной для резервирования памяти и организации доступа к этой памяти *при объявлении структурной переменной и ссылках на отдельные поля структурной переменной.* Фактически шаблон есть *задание нового типа* **struct имя**. Каждый шаблон имеет собственное имя.

Шаблон имеет область определения(видимость). Может быть локальным, если описан внутри блока {}. Если шаблон помещен вне блоков, то он виден во всех функциях ниже точки описания до границы файла.

- *Структурная переменная*. Это место в памяти, где будет располагаться информация о вашей структуре.
  Когда задан шаблон, то можно объявить структурную переменную.
  Компилятор выделяет под структурную переменную число байтов, достаточное для хранения всех ее полей.

ПРИМЕР

```
struct BOOKS{                typedef struct{
  char name[20];               char name[20];
    char title[60];              char title[60];
    int year;                    int year;
    float price;                 float price;
  }                            } BOOKS;

struct BOOKS my_book;          BOOKS my_book;

struct BOOKS mas[25];          BOOKS mas[25];
```

Для того, чтобы программа была более ясной, можно задать типу новое имя с помощью ключевого слова **typedef**.

Структура служит для объединения в одной переменной элементов разных типов.

До начала работы с программой мы в обычном текстовом редакторе создаем файл с именем myfile1.txt с нашими исходными данными..
После редактирования этого файла мы измененный(отредактированный) файл сохраним под именем myfile2.txt.

Создадим наш проект для решения поставленной задачи. Разобьем проект на файлы, в которых разместятся код и ресурсы нашей программы. Часть этих файлов будет создана системой автоматически. Файлы, реализующие алгоритм решения задачи, будут созданы нами.

Проект lab1_struct

funclab1.h	funclab1.cpp	mainlab1.cpp
файл,содержащий шаблон созданной нами структуры и прототипы наших функций	файл, содержащий определения наших функций	файл содержащий функцию main().

Работа с файлом.

Для обмена информацией с файлом приложение должно создать объект класса ifstream для чтения из файла, или объект класса ofstream для вывода в файл. После этого файл должен быть открыт, а при завершении обмена – закрыт. Описание классов ifstream и ofstream находится в заголовочном файле <fstream>, отвечающим за потоковый ввод-вывод в файлы.

Потоковые классы предназначены для управления потоками данных между ОП и внешними устройствами. Потоки описаны в заголовочном файле <iostream>. Элементы заголовочных файлов определены в пространстве имен std. Поток – это перенос данных от источника к приемнику. Потоки определяются последовательностью байтов и не зависят от конкретного устройства, с которым производится обмен. Входной поток – это данные, которые вводятся в ОП. Выходной поток – это данные, которые выводятся из ОП.
Стандартные потоки- от клавиатуры и на зкран.
Файловые потоки – обмен с внешними носителями.

Операторы ввода >> и вывода << выбирают необходимые функции для преобразования данных в поток байтов или обратное преобразование.

!! При вводе строк извлечение происходит до ближайшего пробела.

:: - оператор разрешения области видимости

std :: cout << . . . - идентификатор из пространства имен стандартной библиотеки C++. (namespace – пространство имен)

Динамические массивы

Динамические массивы создаются с помощью операции new ,при этом необходимо указать тип и размерность:

```
subscriber *psub;
psub=new subscriber[n];
```

*psub- указатель на массив структур. Указатели предназначены для хранения адресов области памяти. Не являются самостоятельным типом, но всегда связаны с каким-либо другим конкретным типом. Чаще всего используется для работы с динамической памятью. Доступ к динамической памяти производится только через указатель. Время жизни указателя – от момента создания до конца программы, или до явного освобождения памяти.

Объем памяти, выделяемый под массив, определяется на этапе выполнения программы. Доступ к элементам ДМ такой же, как и к статическим.

ДМ нельзя при создании инициализировать и нельзя обнулять.

Память освобождается оператором delete[]. Размерность в [] не указывается, но сами [] обязательны.

Задание №3

Конструирование класса на основе принципа наследования

Цель: изучить механизм открытого (public) наследования в C++, познакомиться с понятием «виртуальная функция», освоить технологию конструирования и способы документирования программ, включающей в себя *класс - наследники*, изучить возможности инструментальных сред разработки по автоматической генерации кода.

Задание :

1. В соответствии с вариантом задания разработать базовый класс. В базовый класс следует включить свойства и методы, общие для заданных классов-наследников. Базовый класс должен включать в себя не менее двух свойств и двух методов, один из которых – виртуальная функция.
2. Разработать классы, производные от базового класса (наследники). Классы – наследники должны наследовать от базового класса хотя бы одно свойство, а также должны иметь хотя бы одно собственное свойство. В классы - наследники должны быть включены следующие методы:
   - Метод, наследуемый от базового класса
   - Виртуальная функция базового класса, переопределенная в производном классе.
   - Собственные методы производного класса. В состав производного класса должен быть включен хотя бы один метод, изменяющий хотя бы одно свойство какого-нибудь класса.
3. Разрабатываемая вами программа должна выполнять следующие действия:
   - Создание нескольких объектов на основе классов – наследников.
   - Объединение объектов в массив (массив указателей на базовый класс).
   - Отображение значений и свойств объектов на экране в цикле
   - Изменение свойств объектов по номеру элемента массива
   - Вычисление заданного параметра
   - Выход из программы
4. Объявление инициализацию каждого класса поместить в отдельном модуле.
5. Действия из гр. 2 и 3 должны быть доступные через меню; последовательность выполнения действий - произвольная, в цикле.
6. Отчет по заданию 3 должен содержать:
   - Постановка задачи
   - Анализ задачи

- Текст программы
- Диаграмма классов, с указанием свойств и методов класса
- Диаграмма товаров

Вариант задания

Базовый класс	Производные классы	Вычисляемый параметр
Компьютер	Настольный компьютер, ноутбук	Самый дешевый компьютер