УЧЕБНОЕ ПОСОБИЕ

ДЛЯ ВЫСШИХ УЧЕБНЫХ ЗАВЕДЕНИЙ

СПЕЦИАЛЬНОСТЬ

Язык C++ и объектно-ориентированное программирование в C++

Лабораторный практикум

Горячая линия–Телеком

Ашарина И. В.
Крупская Ж. Ф.

Ашарина И. В.
Крупская Ж. Ф.

Язык C++ и объектно-ориентированное программирование в C++

Лабораторный практикум

Рекомендовано федеральным государственным бюджетным образовательным учреждением высшего профессионального образования «Московский государственный технический университет имени Н. Э. Баумана» в качестве учебного пособия для студентов вузов, обучающихся по направлению подготовки 09.03.01 – «Информатика и вычислительная техника»

Москва
Горячая линия – Телеком
2016

УДК 681.3
ББК 32.97
А98

Р е ц е н з е н т ы : доктор техн. наук, профессор *А. И. Гусева*;
доктор техн. наук *А. В. Лобанов*

Ашарина И. В., Крупская Ж. Ф.

А98 Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум. Учебное пособие для вузов. – М.: Горячая линия – Телеком, 2016. – 232 с.: ил.

ISBN 978-5-9912-0464-4.

Пособие содержит 21 лабораторную работу, которые позволят читателю освоить язык программирования С++ в его классическом представлении, а также овладеть технологией объектно-ориентированного программирования в С++. Каждая лабораторная работа включает теоретические сведения, сопровождающиеся большим количеством примеров, работающих в среде MS Visual Studio. Для самостоятельной работы в большинстве лабораторных работ предлагаются наборы заданий двух уровней сложности – для начинающих изучать язык программирования и для тех, кто хочет повысить свой уровень в этой области.

Для студентов и аспирантов высших учебных заведений, изучающих программирование на языке С++ и объектно-ориентированное программирование, а также преподавателей, читающих эти дисциплины. Будет полезно читателям, самостоятельно изучающим программирование на языке С++.

ББК 32.97

Адрес издательства в Интернет WWW.TECHBOOK.RU

Введение

Язык С++ относится к классу универсальных языков, поскольку с его помощью можно решить очень широкий круг задач, выполняемых на ЭВМ. Среди современных алгоритмических языков язык С++ является, пожалуй, одним из самых популярных и распространенных, но наиболее эффективно его применение в написании системных программ-трансляторов, операционных систем, экранных интерфейсов,в обслуживании инструментальных средств.

В большинстве случаев программы, выполненные на языке С++, по быстродействию сравнимы с программами, написанными на Ассемблере. С++ является языком высокого уровня, поэтому программы, подготовленные на нем, более наглядны и просты в сопровождении.

Программы на языке С++ легко переносимы с одного типа компьютера на другой.

Основные особенности языка С++

1) в нем реализованы некоторые операции низкого уровня;

2) его базовые типы данных совпадают с типами данных языка Ассемблера;

3) несмотря на присутствие таких составных объектов, как массивы и структуры, язык не допускает обращения с ними как с единым циклом;

4) широко использует указатели на переменные и функции;

5) удобным средством для передачи параметров являются ссылки;

6) считается языком для профессионалов, поэтому многое «доверяет» программисту: даже на такие важные действия, как преобразование типов, налагаются лишь незначительные ограничения;

7) несмотря на широкие возможности, невелик по объему за счет того, что практически все выполняемые функции оформлены в виде подключаемых библиотек.

Язык С служит базовой платформой для изучения языка С++. Эти два языка имеют так много общих черт, что с методической точки зрения оказывается целесообразным первые темы данного конспекта лекций посвятить изучению языка С, а затем перейти к рассмотрению возможностей, предоставляемых языком С++.

Язык С был разработан в США сотрудниками Bell Laboratories Б.Керниганом и Д.Ритчи и использован для создания ОС UNIX. Во избежание неоднозначных трактовок окончательный вариант был утвержден в качестве стандарта ANSI C.

С++ обязан своим появлением сотруднику Bell Laboratories Б.Страуструппу.

Лабораторный практикум содержит 21 лабораторную работу и позволяет освоить язык программирования C++ в его классическом представлении, а также овладеть технологией объектно-ориентированного программирования в языке C++. Каждая лабораторная работа включает теоретические сведения, сопровождающиеся большим количеством примеров, работающих в среде MS Visual Studio 2008 и 2010. Для самостоятельной работы в большинстве лабораторных работ предлагаются наборы заданий двух уровней сложности – для начинающих изучать язык программирования и для тех, кто хочет повысить свою квалификацию в этой области, что делает книгу интересной для преподавателей, читающих курсы, связанные с программированием на языках высокого уровня.

Практикум предназначен для студентов и аспирантов высших учебных заведений, изучающих основы программирования и объектно-ориентированное программирование на языке C++ и преподавателей, читающих эти дисциплины, а также тех, кто хочет самостоятельно освоить программирование на C++.

Авторы выражают благодарность аспиранту НИУ МИЭТ Зо Мин Кхаингу за помощь в подготовке практикума.

Лабораторная работа 1. Начальные сведения об интегрированных средах разработки программ

Персональный компьютер состоит из нескольких основных частей: устройств ввода-вывода, центрального процессора (ЦП), оперативной памяти (ОП) и внешней памяти.

Центральный процессор – это «мозг» компьютера. Он строго следует программным инструкциям и выполняет соответствующие вычисления. *Инструкция* представляет собой набор единиц и нулей.

Программа – это последовательность инструкций, которые должен выполнить ЦП. Создание программ из инструкций, состоящих из нулей и единиц, и описывающих элементарные действия, очень трудоемко. Поэтому были разработаны языки программирования высокого уровня. Одна инструкция, написанная на таком языке, близка к фразе естественного языка или привычной математической нотации. Она преобразуется компилятором в последовательность элементарных команд, понятных процессору. Язык программирования C++ является высокоуровневым языком.

Перевод инструкций с языка высокого уровня на машинный язык осуществляет специальная программа, называемая *компилятором*.

Для подключения стандартных функций к разрабатываемой программе используется *компоновщик*. Этапы создания машинного кода схематично показано на рис. 1.1.

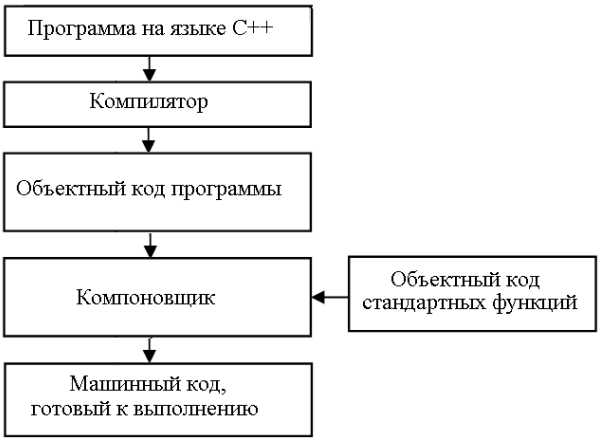


Рис. 1.1. Этапы создания машинного кода программы

Выполнение машинного кода программы осуществляет ЦП после его размещения в ОП. Программа, называемая *загрузчиком*, размещает код в ОП и запускает программу на выполнение.

Работа в интегрированной среде разработки программ MS Visual Studio 2008, работающей под управлением MS Windows

Интегрированная среда разработки программ (ИСРП) включает:

- редактор текста;
- компилятор;
- компоновщик;
- загрузчик.

После того, как среда ИСРП будет загружена, на экране компьютера появится ее окно (рис. 1.2). В верхней части окна располагается строка – меню ИСРП, в нижней части располагается строка, содержащая краткую информацию о положении курсора и режиме вставки/замены (он переключается нажатием клавиши Insert). В левой части экрана – окно проекта, в правой – окно для ввода и редактирования текста программы.

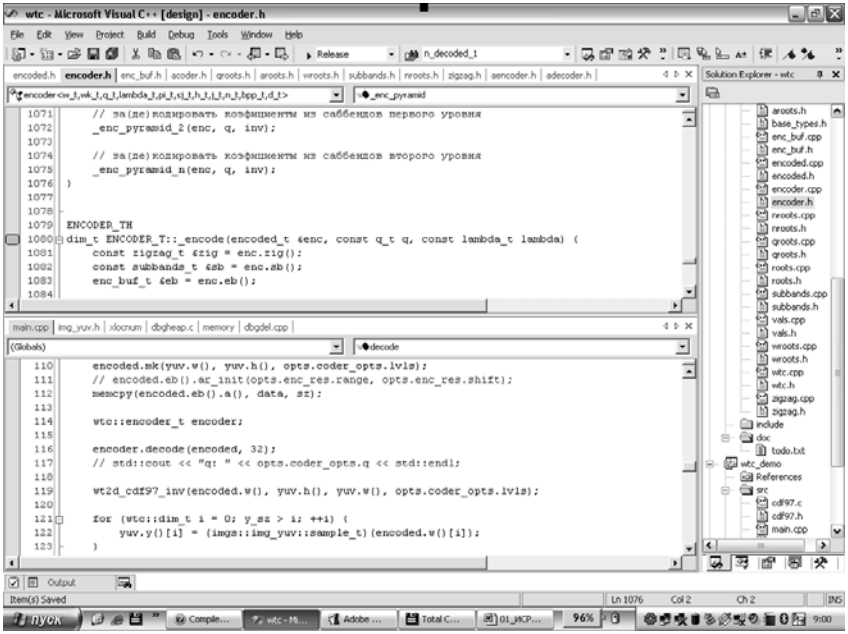


Рис. 1.2. Изображение экрана после входа в ИСРП

Основные пункты меню и их назначение

Пункт меню **File** (рис. 1.3) предназначено для операций с файлами (file), проектами (project) и расширениями (solution). Файл – любой файл на диске. Проект – это набор файлов, объединенных общей идеей. Решение – набор проектов, призванных решать одну или несколько проблем.

New предназначен для создания новых файлов. Однако удобнее пользоваться другим способом (будет описан далее) создания и добавления файлов в проект.

Open открывает ранее созданный файл. При работе с проектами для открытия файлов и для их создания удобнее использовать Solution Explorer .

Close закрывает ранее открытый файл.

Add Project добавляет новый или ранее созданный проект к открытому решению.

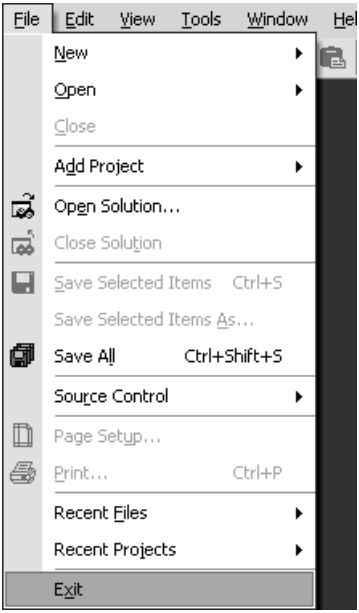


Рис. 1.3. Пункт меню ***File***

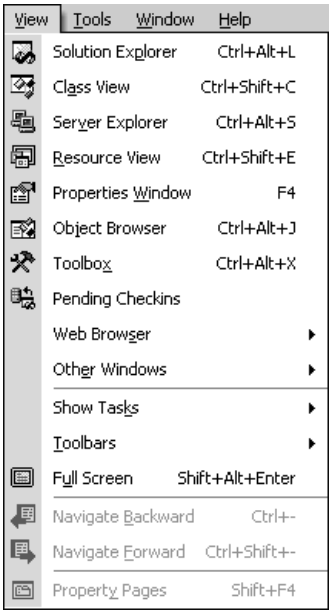


Рис. 1.4. Пункт меню ***View***

Open Solution открывает ранее созданное решение.

Close Solution закрывает открытое решение.

Save… сохраняет сделанные изменения в файлах, проектах, решениях.

Exit завершает выполнение ИСРП.

Пункт меню **Edit** содержит хорошо знакомые всем текстовые операции (вставка, копирование, вырезание, поиск).

Пункт меню **View** (см. рис. 1.4) заведует отображением тех или иных окон, элементов управления и инструментов.

Solution Explorer отображает окно «исследователя решений», своеобразного проводника по решению.

Class View отображает окно с иерархией классов с подробной информацией и об их методах и свойствах.

Properties Window показывает окно свойств выделенного элемента.

Other Windows отвечает за показ окон, не вошедших в корень меню **View**.

Task List – список заданий, который также содержит список ошибок компиляции и компоновки, если таковые были.

Output – это окно вывода компилятора, компоновщика и других программ, работающих под управлением MS VS 2008.

«Горячей» клавишей называют функциональную клавишу или несколько одновременно нажимаемых клавиш клавиатуры. Этими клавишами можно пользоваться для выполнения того или иного действия без входа в соответствующий пункт меню, непосредственно в основном окне ИСРП.

Создание нового проекта

Для создания нового проекта в ИСРП MS Visual Studio 2008 (2010) нужно совершить ряд действий.

- Выбрать пункт меню **File→New→Project** (рис. 1.5).

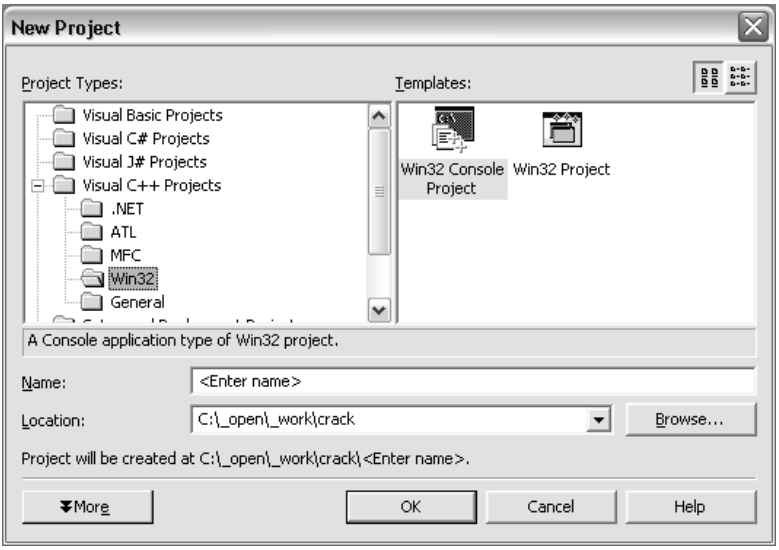


Рис. 1.5. Создание нового проекта

- В **Visual C++ Project** выбираем пункт **Win32**. В правом списке выбираем **Win32 Console Project**. В поле **Name** вводим имя будущего проекта, а в поле **Location** папку, где он будет располагаться. Нажимаем кнопку **Ok**.
- Далее в появившемся окне выбираем слева пункт **Application Settings** (рис. 1.6) и ставим галочку **Empty Project**. Нажимаем кнопку **Finish**.
- Затем в окне **View→Solution Explorer** нажимаем правую кнопку на **Source Files**. В появившемся меню выбираем пункт **Add→Add New Item (**рис. 1.6).
- В появившемся окне выбираем в списке справа **C++ File (.cpp)** (рис. 1.7). В поле **Name** вводим имя создаваемого файла. Поле **Location** не трогаем, так как в нем уже написан верный путь к проекту. Нажимаем **Open**.

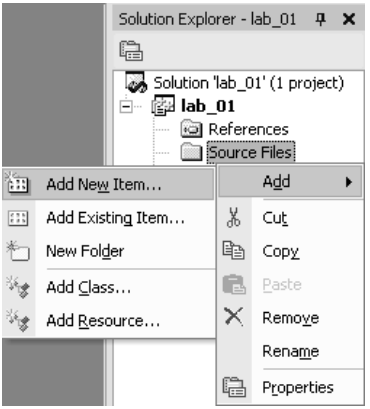


Рис. 1.6. Создание нового проекта. Продолжение

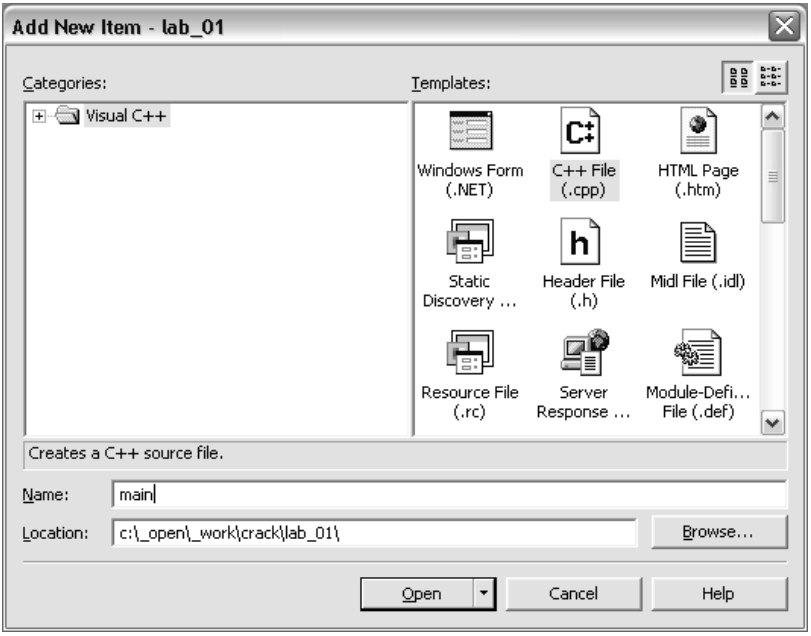


Рис. 1.7. Создание нового проекта. Продолжение

В следующем окне выбрать начальный состав проекта. Выбираем «A simple program». Нажимаем «Finish»: новый проект создан (рис. 1.8).

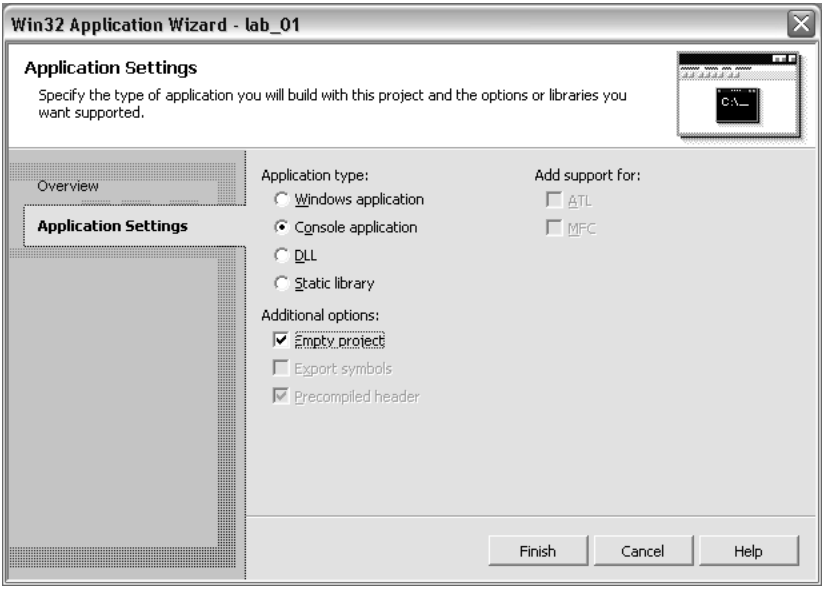


Рис. 1.8. Создание нового проекта. Завершение

Рассмотрим окно нового проекта.

Слева от рабочего окна находится панель управления проектом, состоящая из двух закладок: закладки «Class view», отображающей информацию о функциях и переменных проекта (двойной щелчок по названию функции открывает в рабочем файле место с объявлением этой функции), и закладки «File view», отображающей информацию о рабочих файлах проекта.

Двойной щелчок по названию рабочего файла проекта откроет окно редактирования этого файла. Можно приступать к работе.

Структура программы на языке C++

Программа на языке C++ состоит из одной или более функций, причем одна из них должна называться main или _tmain:

```
int _tmain (int argc, _TCHAR* argv[]) // или так:
int  main()
    {
     операторы языка C++
     return 0;
    }
```

Если при написании программы использовались стандартные функции, то перед заголовком main необходимо указать компилятору, какие файлы с описанием заголовков этих функций требуется подключить. Такое указание называется *директивой препроцессору* и имеет вид:

```
#include <имя подключаемого файла>
```

Угловые скобки < > указывают, что подключаемый файл находится в системной папке. Для подключения файлов, находящихся в личной папке используются символы кавычек (").

При использовании в программе стандартных потоков ввода/вывода необходимо подключить файл iostream. В нем определены необходимые нам объекты «cout» – поток, связанный с выводом на консоль, объект «cin» – поток, связанный с вводом с консоли. Кроме этого, указать для использования стандартное пространство имен (std).

```
#include <iostream>
using namespace std;
```

Идентификаторы языка C++

Идентификатор – это имя переменной, константы и т. д. Оно должно начинаться с буквы латинского алфавита или знака подчеркивания (_), далее могут следовать цифры, буквы или знак подчеркивания. В языке C++ коды прописных и строчных букв различны, поэтому, например, «A» и «a» – это разные символы.

Оператор присваивания

Общий вид: **идентификатор** = **выражение**; где выражение – это любое выражение на языке C++, например c = a + b;

Пример 1.1. Демонстрация операторов ввода/вывода (рис. 1.9).

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
   setlocale (LC_ALL, "Russian"); // подключение
                                  // русификатора
   int num=5;                // определение переменной
                             // целого типа num с
//присваиванием значения = 5
   cout << "Я простая ";
   cout << "вычислительная машина \n";
   cout << "Мое любимое число  " << num << endl;
   cout << "А какое твое любимое число?   ";
   cin >> num;
   cout << "Теперь я знаю твое любимое число.
        << Оно равно " <<  num << endl;
```

```
    _getch();
    return 0;
}
```

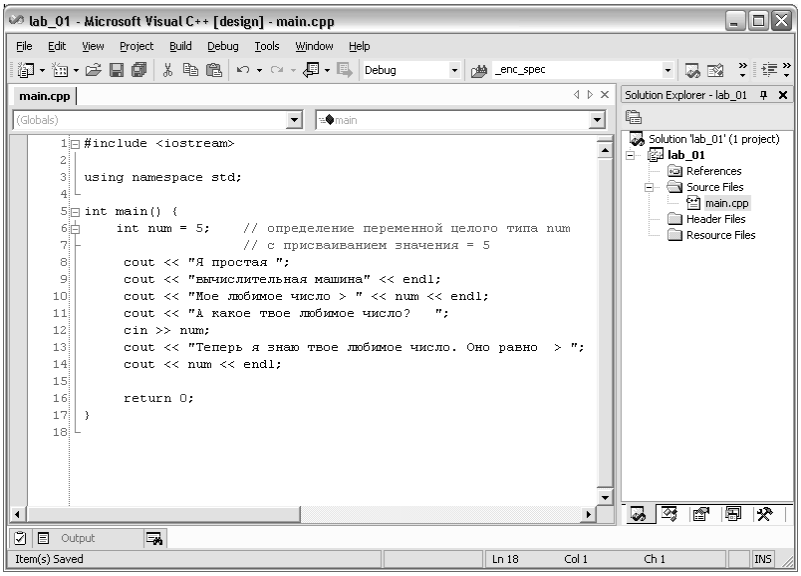


Рис. 1.9. Результат работы программы из примера 1.1

Результат выполнения программы:

```
Я простая вычислительная машина
Мое любимое число 5
А какое твое любимое число? <ожидание ввода числа>
Теперь я знаю твое любимое число. Оно равно <введенное
число>
```

Пример 1.2. Написать программу нахождения суммы введенных чисел.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
    setlocale (LC_ALL, "Russian"); // подключение
                                   // русификатора
    int   a,                       // слагаемое a,
          b,                       // слагаемое b,
          summa;                   // сумма
    cout << "Введите слагаемое a ";
    cin >> a;
    cout << "Введите слагаемое b ";
    cin >> b;
```

```
    summa = a + b;
    cout << "Сумма = " << summa << endl;
    _getch();
    return 0;
}
```

Запуск программы

Для запуска написанной программы, ее необходимо сначала скомпилировать. Для этого в меню **Build** выберите пункт **Build Solution**, чтобы скомпилировать и скомпоновать все решение целиком, или **Build lab_01**, чтобы скомпилировать и скомпоновать текущий проект. Далее для запуска программы в меню **Debug** выберите пункт **Start Debug** или **Start Without Debugging**. В первом случае процессор выполняет программу в режиме отладки, что позволяет выполнять ее в пошаговом режиме, устанавливать точки останова и выполнять другие действия, связанные с ее отладкой. Во втором случае программа просто запускается на выполнение.

Контрольные вопросы

1. Перечислите этапы подготовки к выполнению программы на языке C++?
2. Как войти в ИСРП?
3. Перечислите основные пункты меню и их назначение?
4. Какова структура программы на языке C++?
5. Для чего необходима директива препроцессору #include?
6. Каким образом можно осуществить вывод информации на экран?
7. Как можно организовать ввод данных с клавиатуры?
8. Что называют идентификатором?

Задания

Таблица 1.1. Простые варианты

Вариант	*Задание*

1	Исправить ошибки:
	`includ <iostream.h>`
	`main {}`
	`( cout << "Problems, problems \ n";`
	`cout << Problems all day long! \ n;`
	`cout << "The Everly Brothers \ n"`
	`)`

Вариант	*Задание*

2	Исправить ошибки:
	`include <io stream.h>`
	`main {} // эта программа печатает число недель в году`
	`( int s`
	`S := 56;`
	`cin << В году S недель;`

3	Исправить ошибки:
	`# include <iostream.h>`
	`main()`
	`( cout ("Здравствуйте \  n ");`
	`cout ( 'Я - программа на С' \n);`
	`cout ("которая причинит кучу неприятностей \ ?")`
	`)`

4	Исправить ошибки:
	`# include <iostream.c>`
	`main()`
	`Begin`
	`int S;`
	`S :=56;`
	`print ( 'В году' , S, 'недель')  End`

5	Исправить ошибки:
	`# include <ioctrim.c>;`
	`main()`
	`{ int include, j, k`
	`SUM = include + j;`
	`cout ('Сумма будет равна % d, sum');`
	`cout ('Ну что? Опять ошибки?')}`

6	Исправить ошибки:
	`include <iostream.h>`
	`main()`
	`( cout («Программирование - это сказка: /n»);`
	`cout («чем дальше, тем страшнее! \ u);`
	`cout («Станем сказочными героями ! \  n”)`
	`)`

7	Исправить ошибки:
	`include <include.h>`
	`main(); // эта программа печатает число месяцев`
	`// в году`
	`( int m;`
	`m := 12;`
	`cout << «В году m месяцев» )`

8	Исправить ошибки:
	`# include <iostream.h>`
	`main()`
	`( cout << 'Доброе утро, \  y ';`
	`cout << 'если, конечно, оно действительно`
	`// доброе.' \7;`
	`cout << 'И ослик Иа грустно махнул хвостом\n'`
	`)`

Вариант	*Задание*

9	Исправить ошибки:
	`# include <stream.c>`
	`main();`
	`Begin`
	`int d;`
	`D :=29;`
	`cin >> «В январе»' , d, «недель»`
	`End`

10	Исправить ошибки:
	`# include <iostream.c>;`
	`main()`
	`{ int I, j, k`
	`SUM = I + j;`
	`cout ('Сумма будет равна % d, sum');`
	`cout ('Ну что? Опять ошибки?') }`

11	Исправить ошибки:
	`#inclyde <stream.h>`
	`main;`
	`{cin ("Хотел бы я знать, зачем звезды светятся.\N");`
	`cin ("Наверно, чтобы рано или поздно" \N);`
	`cin (" каждый мог опять найти свою\N") }`

12	Исправить ошибки:
	`$includ <iostream.h>`
	`main()`
	`int min`
	`min= 60;`
	`cout (В часу &min минут);`

13	Исправить ошибки:
	`# include <iostream.cpp>`
	`MAIN();`
	`Begin`
	`cout << ("Снова тянет с берега снегом`
	`и туманом,\ff ");`
	`cout << ( "Снова ночь нелетная`
	`даже для луны" \l);  End`

14	Исправить ошибки:
	`# includ <iostream.c>`
	`main()`
	`Begin  int S;`
	`scanf (S);`
	`print ( 'В году' , %S, 'недель')`
	`End`

15	Исправить ошибки:
	`# include <studio.c>;`
	`main();`
	`{ int i, j, MUL`
	`MUL = i * j;`
	`printf ('Произведение равно &MUL')`
	`printf ('Оно действительно такое?') }`

Таблица 1.2. Сложные варианты

Вариант	*Задание*
1	Разработать программу, которая вычисляет среднее арифметическое значение трех введенных с клавиатуры чисел
2	Разработать программу, которая вычисляет площадь круга по введенному с клавиатуры радиусу
3	Разработать программу, которая вычисляет объем цилиндра по введенным с клавиатуры радиусу и высоте
4	Разработать программу, которая вычисляет объем конуса по введенным с клавиатуры радиусу и высоте
5	Разработать программу, которая вычисляет сумму цифр введенного с клавиатуры четырехзначного числа
6	Разработать программу, которая вычисляет объем шара по введенному с клавиатуры радиусу
7	Разработать программу, которая формирует число по введенным с клавиатуры цифрам, трактуемым как число сотен, десятков и единиц
8	Разработать программу, которая «переворачивает» введенное с клавиатуры четырехзначное число
9	Разработать программу, которая вычисляет произведение цифр введенного с клавиатуры четырехзначного числа
10	Разработать программу, которая вычисляет объем куба по введенной с клавиатуры длине ребра
11	Разработать программу, которая вычисляет площадь поверхности правильного тетраэдра по введенной с клавиатуры длине ребра
12	Разработать программу, которая вычисляет объем усеченного конуса по введенной с клавиатуры высоте и двум радиусам
13	Разработать программу, которая вычисляет площадь треугольника по введенным с клавиатуры двум сторонам и углу между ними
14	Разработать программу, которая вычисляет площадь трапеции по введенным с клавиатуры основаниям и высоте
15	Разработать программу, которая вычисляет площадь кольца по введенным с клавиатуры внешнему и внутреннему радиусам

Лабораторная работа 2. Программирование ветвящихся алгоритмов

Часто возникает необходимость выбора одного из альтернативных путей решения в зависимости от значения некоторого выражения. Оператор, который позволяет осуществить такой выбор, называется *условным оператором*.

Условный оператор if

Синтаксис полной формы условного оператора:

```
if (выражение) оператор1;
else оператор2;
```

Если **выражение** истинно, т. е. не равно нулю, то выполняется **оператор1**, иначе выполняется **оператор2** (рис. 2.1).

Синтаксис сокращенной формы условного оператора:

```
if (выражение) оператор;
```

Рис. 2.1. Обозначение условного оператора в схеме алгоритма

Оператор выполняется только в том случае, если **выражение** не равно нулю, т. е. истинно.

Как правило, **выражение** является выражением отношения или совокупностью операций отношения и логических операций (табл. 2.1).

Таблица 2.1. Основные операции отношения и логические операции

Операция	*Значение*	*Пример*
= =	Равно	value==0
!=	Не равно	value!=0
<	Меньше	i<count
>	Больше	i>count
> =	Больше или равно	i>=count
< =	Меньше или равно	i<=count
||	Логическое ИЛИ	!a || b
&&	Логическое И	a>8 && c<5
!	Логическое НЕ	!EOF

Часто каждая из альтернативных возможностей оператора if требует выполнения более одного оператора. В этом случае необходимо заключить группу операторов в фигурные скобки { }.

Список операторов, заключенный в фигурные скобки, называется *составным оператором* или *блоком*. Составные операторы выполняются так же, как и простые операторы, и могут находиться в любом месте программы.

Оператор множественного выбора

Когда необходимо выбрать один из нескольких вариантов, можно воспользоваться конструкцией if else if…else или оператором множественного выбора switch.

Синтаксис оператора switch:

```
switch (выражение)
{
    case константное выражение:
                    оператор или группа операторов;
                    break;
    case константное выражение:
                    оператор или группа операторов;
                    break;
    case константное выражение:
                    оператор или группа операторов;
                    break;
    …
    defaulf: оператор или группа операторов;
}
```

Результат вычисленного **выражения** сравнивается с каждым из **константных выражений**. Если находится совпадение, то управление передается оператору, связанному с данным case. Исполнение продолжается пока не встретится оператор break, который передает управление из тела switch оператору, следующему за switch или до конца тела оператора switch. **Оператор** или **группа операторов**, стоящая после ключевого слова default, выполняется, если **выражение** не соответствует ни одному из **константных выражений** в case.

Например, в операторе

```
switch (c)
      { case ‘A’: capa++;
      case ‘a’: letter++;
      default: total++;
      }
```

при c=’A’ инкрементируются переменные capa, letter, total, при c=’a’ – letter и total, при всех остальных значениях переменной c – только total.

Константные выражения должны быть целого или символьного типа. Если нескольким **константным выражениям** соответствует один и тот же **оператор**, то возможна запись:

```
switch (выражение)
{
    case константное выражение:
    case константное выражение:
    case константное выражение:
            оператор или группа операторов;
            break;
};
```

Примеры программирования

Пример 2.1. Написать программу нахождения действительных корней квадратного уравнения общего вида $ax^2+bx+c=0$.

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
 int  a,          // Коэффициенты a,
      b,          //               b,
      c;          //               c
setlocale (LC_ALL, "Russian"); // подключение русификатора
 cout << "Введите коэффициенты a b c > ";
 cin >> a >> b >> c; //Ввод данных
 float d=b*b-4*a*c; //Вычисление дискриминанта
 if ( d>=0)        //Если дискриминант больше или равен 0,
      {            //Вычислить корни x1, x2
      float x1= (-b+sqrt (d))/ (2.0*a);
      float x2= (-b-sqrt (d))/ (2.0*a);
                  //Вывод на экран значений корней:
      cout << "Первый корень = " << x1 << endl;
      cout << "Второй корень = " << x2 << endl;
    }
//Если корней нет, то вывод сообщения "Корней нет"
 else  cout << "Корней нет" << endl;
_getch();
return 0;
}
```

Пример 2.2. Написать программу, которая по введенному с клавиатуры номеру дня недели выводит на экран название этого дня.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;
```

```
int _tmain (int argc, _TCHAR* argv[])
{
 int num;
 setlocale (LC_ALL, "Russian"); // подключение
                                // русификатора
 cout << "Введите номер дня недели >";
 cin >> num;           //Ввод данных
switch (num)           //Выбор варианта
   {  case 1: cout << "Понедельник" << endl;
              break;
      case 2: cout << "Вторник" << endl;
              break;
      case 3: cout << "Среда" << endl;
              break;
      case 4: cout << "Четверг" << endl;
              break;
      case 5: cout << "Пятница" << endl;
              break;
      case 6: cout << "Суббота" << endl;
              break;
      case 7: cout << "Воскресенье" << endl;
              break;
      default: cout << "Номер неверен" << endl;
   }
_getch();
return 0;
}
```

Контрольные вопросы

1. Какие операторы позволяют организовать выбор между несколькими вариантами?
2. Что понимают под логическим выражением?
3. Чем отличается логическое И от логического ИЛИ?
4. Какие операции отношения вы знаете?
5. В чем особенность использования операторов if и switch?

Задания

Таблица 2.2. Простые варианты

Вариант	*Задание*
1	Определить, какое из выражений является меньше: $a*\ln b$, $\sin a*\sqrt{b}$
2	По номеру y ($y > 0$) некоторого года определить c – номер его столетия. Учесть, наример, что началом XXI столетия был 2001, а не 2000 год

Вариант	*Задание*
3	Даны произвольные числа *a*, *b*, *c*. Если нельзя построить треугольник с такими длинами сторон, то выдать соответствующее сообщение; если можно, то вычислить его площадь по формуле Герона
4	Пользуясь оператором switch, по введенному номеру месяца выдать на экран сообщение о времени года и названии введенного месяца. Например: 1 – январь, зима…
5	Составить программу, которая при вводе символа с клавиатуры выводит «цифра», если введена цифра и «не цифра» во всех остальных случаях
6	Даны *a* и *b*. Напечатать максимальное значения выражений $a \cdot \sin b$ $\cos a$ $\sqrt{a \cdot b}$
7	Пользуясь оператором switch, по введенному номеру месяца вывести на экран сообщение о номере квартала.
8	Найти значение *y* по введенному значению *x*: $y=\begin{cases} 2x^3+\sqrt[3]{x} & при \quad x \ge 0 \\ \dfrac{e^x}{1+x^2}-1 & при \quad x<0. \end{cases}$
9	Составить программу, которая при вводе символа *c* клавиатуры выводит «латинская буква» при вводе латинской буквы и «не латинская буква» во всех остальных случаях
10	При вводе с клавиатуры символа: «+» вывести сообщение «сложение», «–» вывести сообщение «вычитание», «*» вывести сообщение «умножение», «/» вывести сообщение «деление»
11	Составить программу, которая при вводе символа «{» или «}» выводит сообщение «фигурная скобка»;. при вводе «[» или «]» – «квадратная скобка»; при вводе «(» или «)» – «круглая скобка»; в остальных случаях – сообщение «не скобка»
12	Даны произвольные числа *a* и *b*. Поменять их значения так, чтобы стало *a*>=*b*
13	Составить программу, которая при вводе оценки в виде цифры выводит оценку в виде слова: 5 – отлично, 4 – хорошо, 3 – удовлетворительно, 2 – неудовлетворительно

Вариант	*Задание*
14	Найти значение y по введенным значениям x и a: $y=\begin{cases} 2ax+\lvert a-1\rvert & при \quad a \ge 0 \\ \dfrac{e^x}{1+a^2}-1 & при \quad a<0. \end{cases}$

Таблица 2.3. Варианты повышенной сложности

Вариант	*Задание*
1	Расположить в порядке возрастания значения, полученные в результате вычисления выражений: $a*\ln b$, $\mathrm{tg}a+b/a$, $\sin a*\sqrt{b}$ с указанием формул, по которым производились вычисления
2	По номеру y ($y > 0$) некоторого года определить c – номер его столетия. Учесть, например, что началом XXI столетия был 2001, а не 2000 год. Задачу решить двумя способами: с использованием операторов принятия решения и без использования
3	Даны произвольные числа a, b, c. Если нельзя построить треугольник с такими длинами сторон, то выдать соответствующее сообщение; если можно, то напечатать какой он: равносторонний, равнобедренный, разносторонний и вычислить его площадь по формуле Герона
4	Составить программу, которая по введенному году и номеру месяца определяет число дней в этом месяце
5	По введенному времени и известному расписанию занятий вывести сообщение о том, что это: пара (с указанием ее номера) или перемена
6	Даны a и b. Напечатать максимальное и минимальное значения выражений $a\cdot\sin b$, $\cos a$, $\sqrt{a\cdot b}$ с указанием формул, по которым производились вычисления
7	Элементы окружности пронумерованы таким образом: 1– радиус (R), 2 – диаметр (D), 3 – длина (L), 4 – площадь круга (S). По номеру элемента и его значению вычислить значения остальных элементов
8	Составить программу, которая при вводе символа с клавиатуры выводит «цифра», если введена цифра; «латинская буква» – при вводе латинской буквы и «не цифра и не латинская буква» во всех остальных случаях

Вариант	*Задание*
9	При вводе с клавиатуры символа: «+» вывести сообщение «сложение», «–» вывести сообщение «вычитание», «*» вывести сообщение «умножение», «/» вывести сообщение «деление» с указанием формулы и примера с конкретными значениями, введенными с клавиатуры
10	Составить программу, которая при вводе символа «{» или «}» выводит сообщение «фигурная скобка»; при вводе «[» или »]» – «квадратная скобка»; при вводе «(» или «)» – «круглая скобка»; в остальных случаях – сообщение «не скобка». Разработать программу для двух вариантов – с использованием оператора *if* и *switch*
11	Даны произвольные числа *a*, *b* и *c*. Определить максимальное, минимальное и среднее число
12	Даны произвольные числа *a* и *b*. Поменять их значения так, чтобы стало *a*<=*b*<=*c*
13	Составить программу, которая выводит введенный возраст человека с добавлением слов «год», «года», «лет» (41 год, 3 года, 20 лет)
14	Для заданного числа a найти корень уравнения $f(x) = 0$, где $\begin{cases} 2ax + \lvert a-1 \rvert & при \quad a \ge 0 \\ \dfrac{e^x}{1+a^2} - 1 & при \quad a < 0. \end{cases}$

Лабораторная работа 3. Разработка программ с использованием циклов

Цикл – это многократное повторение оператора или блока операторов пока выполняется некоторое условие.

Оператор цикла for

Общая форма записи оператора цикла for имеет вид:

```
for (инициализация управляющих переменных цикла;
проверка на продолжение цикла;
закон изменения управляющих переменных цикла)
оператор;
```

Оператор цикла for содержит три выражения, каждое из которых не является обязательным (рис. 3.1). Первое выражение инициализирует управляющие переменные цикла. Второе выражение описывает условие, которое определяет, будет ли выполняться следующая итерация цикла. Благодаря третьему выражению оператора цикла происходит увеличение или уменьшение значения управляющей переменной цикла.

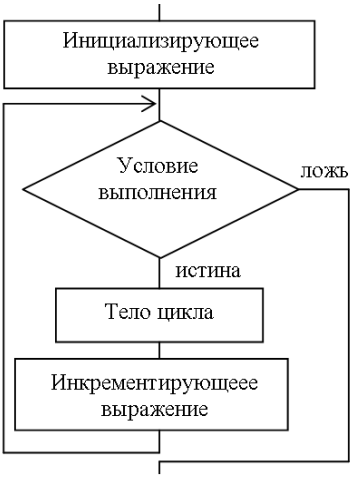


Рис. 3.1. Обозначение цикла for в схеме алгоритма

В состав цикла for может входить операция «*запятая*», которая увеличивает гибкость его использования, позволяя включать в описание оператора несколько инициализирующих или корректирующих выражений.

Пример записи оператора цикла for:

```
for (int y=1, int n=1; n<=10; n++)  y*=n;
```

Оператор цикла допускает изменение управляющей переменной в сторону уменьшения:

```
for (int n=10; n>=0; n--)
cout << n << "секунд !" << endl;
cout << " Пуск !" << endl;
```

Параметром оператора цикла может быть переменная не только целого, но и символьного типа:

```
for (char ch='a'; ch<='z'; ch++)
     cout << "Символ " << ch << endl;
```

При выполнении приведенного примера будут выведены на экран все буквы от a до z.

Можно сделать так, чтобы значение некоторой величины возрастало в геометрической, а не в арифметической прогрессии, т. е. чтобы вместо прибавления фиксированного значения на каждом шаге цикла выполнялось умножение:

```
for (int d=1; d<15; d=d*2)
     cout << d << endl;
```

В цикле for может отсутствовать одно или более выражений (но при этом обязательно наличие символов «точка с запятой»). При этом необходимо включить в тело цикла один или несколько операторов, которые рано или поздно приведут к завершению его работы:

```
a=2;
for ( n=3; a<=25;)  a=a * n;
```

Тело цикла

```
for ( ; ; )
   cout << "Мы написали бесконечный цикл" << endl;
```

будет выполняться бесконечное число раз, поскольку пустое условие всегда считается истинным.

Если в цикле оставлены пустыми все три компонента, как в последнем из приведенных примеров, то такой цикл называется *открытым*.

Выход из такого цикла может осуществляться двумя способами:

1. Оператор break служит для продолжения работы программы после окончания текущего цикла и используется, если после выхода из цикла необходимо выполнить оставшуюся часть программы.

2. Функция exit(), объявленная в заголовочном файле stdlib.h, позволяет выйти из программы. Эта функция используется в том случае, если нужно прекратить выполнение шагов цикла и выйти из программы.

Оператор continue служит для завершения текущей итерации цикла и перехода к следующей итерации этого же цикла, но не является способом выхода из цикла.

Оператор цикла do-while

В операторе цикла do-while условие повторения проверяется после каждого прохождения тела цикла, т. е. это цикл с последующим условием. Следовательно, цикл do-while выполняется по крайней мере один раз. Этот цикл повторяется до тех пор, пока выполняется условие, проверяемое в конце цикла.

Форма его записи:

```
do {оператор или группа операторов}
while (условие);
```

Даже в случае, когда тело цикла содержит единственный оператор, фигурные скобки следует писать во избежание неправильной трактовки компилятором служебного слова do. Обозначение цикла do while в схеме алгоритма показано на рис. 3.2.

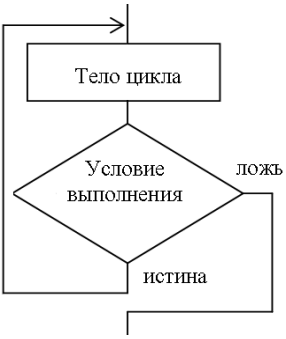


Рис. 3.2. Обозначение цикла do-while в схеме алгоритма

```
. . .
do {    ch=getchar ( );  // принимать с клавиатуры символы
putchar (ch);            // и выводить их на экран
    }   while (ch != ' \n' ); // пока не будет нажата
                              // клавиша Enter
. . .
```

Оператор цикла while

Оператор цикла while является еще одной разновидностью условного цикла, повторяющегося до тех пор, пока выполняется условие, проверяемое перед началом каждой итерации цикла. Таким образом, это цикл с предварительным условием. Обозначение цикла while в схеме алгоритма показано на рис. 3.3.

Рис. 3.3. Обозначение цикла while в схеме алгоритма

Форма записи:

```
while (условие) {последовательность операторов}
. . .
    int index=2;
    while (index++<5) cout << "Желаю удачи!" <<endl;
. . .
```

Пример 3.1. Из *n* целых чисел, введенных с клавиатуры, определить максимальную последовательность четных чисел.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
 setlocale (LC_ALL, "Russian"); // подключение русификатора
 int n, x;
 int cnt=0;
 int k=0;
 cout<<"Введите последовательность из ";
 cin>>n;
 cout<<"чисел\n";
 for (int i=1; i<=n; i++)
    {    cin>>x;
        cnt=0;
        while (x%2==0)
        {    cnt++;
            if (i==n) break;
            cin>>x;
            i++;
        }
        if (cnt>k)
            k=cnt;
    }
 cout<<"Количество четных равно "<<k<<endl;
 _getch();
return 0;
}
```

Контрольные вопросы

1. Каким образом цикл **while** может имитировать цикл **for**?
2. Каким образом цикл **while** может имитировать цикл **do-while?**
3. В каких случаях используются операторы **break, continue, exit?**
4. Почему в языке C++ нет необходимости использовать оператор **goto?**

Задания

Таблица 3.1. Простые варианты

Вариант	*Задание*
1	Ввести с клавиатуры целое число, которое будет являться количеством символов во вводимой последовательности. Определить, является ли последовательность упорядоченной по алфавиту

Вариант	*Задание*
2	Ввести с клавиатуры последовательность из *k* целых чисел. Определить, является ли последовательность упорядоченной по возрастанию, если *k* также введено с клавиатуры
3	Ввести с клавиатуры последовательность из *n* действительных чисел. Определить, есть ли в последовательности хотя бы одна пара одинаковых соседних чисел. Если есть, вывести их порядковые номера
4	Ввести с клавиатуры последовательность из *k* упорядоченных по возрастанию действительных чисел и число *n*, не равное ни одному из чисел последовательности. Найти ближайшее к *n* число последовательности (его порядковый номер и значение)
5	Определить, является ли последовательность цифр *n*-значного введенного с клавиатуры целого числа упорядоченной (*n* считается заданным). Для произвольного *n* задача существенно усложняется
6	Определить, какая из цифр введенного с клавиатуры целого числа расположена левее, максимальная или минимальная
7	Ввести с клавиатуры целое число, которое будет являться количеством символов во вводимой последовательности. Найти количество чисел в последовательности
8	Ввести с клавиатуры последовательность из *k* упорядоченных по возрастанию действительных чисел и число *n*, не равное ни одному из чисел последовательности. Вывести все числа последовательности, меньшие *n*
9	Вывести на экран все четырехзначные целые числа, в записи которых нет одинаковых цифр
10	Ввести с клавиатуры целое число, которое будет являться количеством символов во вводимой последовательности. Вывести на экран элементы последовательности до первого повторяющегося из подряд идущих символов
11	Ввести с клавиатуры последовательность целых чисел. Вывести на экран минимальное и максимальное значения введенной последовательности
12	Ввести с клавиатуры последовательность целых чисел. Определить, имеется ли во введенной последовательности упорядоченные по возрастанию подпоследовательности

Таблица 3.2. Варианты повышенной сложности

Вариант	*Задание*
1	Дано начальное значение a_0 =5 и рекуррентная формула $a_i = a_{i-1} + 3i/2$. Найти номер первого элемента, превысившего введенное с клавиатуры число
2	Дано начальное значение a_0 =1 и рекуррентная формула $a_i = \frac{a_{i-1}}{2i}$. Найти наименьший номер элемента последовательности, для которого выполняется условие $\lvert a_i - a_{i-1} \rvert < \varepsilon$, введенное с клавиатуры. Вывести на экран этот номер и все элементы a_i
3	Дано начальное значение a_0 = 2 и рекуррентная формула $a_i = 2i + \frac{1}{a_{i-1}}$. Найти номер первого элемента, превысившего введенное с клавиатуры число
4	Дано начальное значение a_0 = 0 и рекуррентная формула $a_i = e^{-a_{i-1}}$. Найти наименьший номер элемента последовательности, для которого выполняется условие $\lvert a_i - a_{i-1} \rvert < \varepsilon$, введенное с клавиатуры. Вывести на экран этот номер и все элементы a_i
5	Дано начальное значение a_0 =0.5 и рекуррентная формула $a_i = 2i \cos a_{i-1}$. Найти номер первого элемента, превысившего введенное с клавиатуры число
6	Дано начальное значение a_0 =*x* и рекуррентная формула $a_i = \frac{x}{2a_{i-1}^2}$. Найти наименьший номер элемента последовательности, для которого выполняется условие $\lvert a_i - a_{i-1} \rvert < \varepsilon$, введенное с клавиатуры. Вывести на экран этот номер и все элементы a_i. Значение *x* ввести с клавиатуры
7	Дано начальное значение a_0 =2 и рекуррентная формула $a_i = \frac{2 + a_{i-1}^2}{2ia_{i-1}}$. Найти номер первого элемента, превысившего введенное с клавиатуры число
8	Дано начальное значение a_0 =1 и рекуррентная формула $a_i = \frac{i}{2}\left(a_{i-1} + 2/a_{i-1}\right)$. Найти номер первого элемента, превысившего введенное с клавиатуры число

Вариант	*Задание*
9	Для числового ряда, общий член которого имеет вид: $a_n = \frac{n!}{(2n)!}$, найти сумму тех членов ряда, для которых $\lvert a_n \rvert \geq \varepsilon$, введенного с клавиатуры.
10	Дано начальное значение a_0 =1, r (ввести с клавиатуры) и рекуррентная формула $a_i = (1+r)a_{i-1} - ra_{i-1}^2$. Найти наименьший номер элемента последовательности, для которого выполняется условие $\lvert a_i - a_{i-1} \rvert < \varepsilon$, введенное с клавиатуры. Вывести на экран этот номер и все элементы a_i
11	Для числового ряда, общий член которого имеет вид: $a_n = \frac{(-1)^{n-1}}{n^n}$, найти сумму тех членов ряда, для которых $\lvert a_n \rvert \geq \varepsilon$, введенного с клавиатуры
12	Для числового ряда, общий член которого имеет вид: $a_n = \frac{1}{2^n} + \frac{1}{3^n}$, найти сумму тех членов ряда, для которых $\lvert a_n \rvert \geq \varepsilon$, введенного с клавиатуры
13	Для числового ряда, общий член которого имеет вид: $a_n = \frac{2n-1}{2^n}$, найти сумму тех членов ряда, для которых $\lvert a_n \rvert \geq \varepsilon$, введенного с клавиатуры
14	Для числового ряда, общий член которого имеет вид: $a_n = \frac{1}{(3n-2)(3n+1)}$, найти сумму тех членов ряда, для которых $\lvert a_n \rvert \geq \varepsilon$, введенного с клавиатуры

Приложение. Отладка приложений в ИСР Borland C++ Builder и MS Visual C++

«Горячие» клавиши для отладки приложений в ИСР Borland C++ Builder

Клавиши	*Описание*	*Назначение*
F4	Run to cursor	Выполнить до текущего места
F5	Set breakpoint	Установить точку прерывания программы

Клавиши	*Описание*	*Назначение*
F7	Trace into	Выполнить строку программы с заходом в код вызываемых функций
F8	Step over	Выполнить строку программы без захода в код вызываемых функций
F9	Run	Выполнить
Shift+F7	Trace to next source line	Выполнить до следующей строки с исполняемым кодом
Ctrl+Alt+L	Show local variables	Показать окно значений локальных переменных
Ctrl+Alt+W	Show watches	Показать окно слежения за переменными
Ctrl+Alt+B	Show breakpoints	Показать окно точек прерывания
Ctlr+F2	Stop	Остановка отладки

«Горячие» клавиши для отладки приложений в ИСР Visual C++

Клавиши	*Описание*	*Назначение*
Ctrl+F10	Run to cursor	Выполнить до текущего места
F9	Set breakpoint	Установить точку прерывания программы
F11	Trace into	Выполнить строку программы с заходом в код вызываемых функций
F10	Step over	Выполнить строку программы без захода в код вызываемых функций
F5	Run	Выполнить
Ctrl+Shift+F5	Restart	Выполнить с учётом изменений
Shift+F5	Stop	Остановить отладку

При отладке необходимо соблюдать правила:

- Вводить в критические места программы точки прерываний.
- На критическом участке выполнять программу по шагам.
- При вызове вашей функции используйте отладку с заходом в функции. Если же в текущей строке программы вызываются только библиотечные функции – используйте отладку без захода в функции.
- Для слежения за значениями переменных внутри функции используйте окно значений локальных переменных.
- Для слежения за выбранными переменными используйте окно слежения за переменными.
- В окне слежения за переменными используйте Ins для добавления переменной и Del для удаления.

Отладка приложений позволяет выявить ошибки алгоритма и ошибки программиста, которые компилятор самостоятельно выявить не может.

Лабораторная работа 4. Использование циклов для решения задач численными методами

Вычисление значения $\int f(x)dx$ с заданной точностью методом прямоугольников

Для вычисления первого приближения интеграла разделим отрезок [*a*,*b*], отвечающий пределам интегрирования (рис. 4.1), на *n* равных частей (*n* = 4), определим значения $x_i = a + h*i - h/2$; $h = (b - a)/n$.

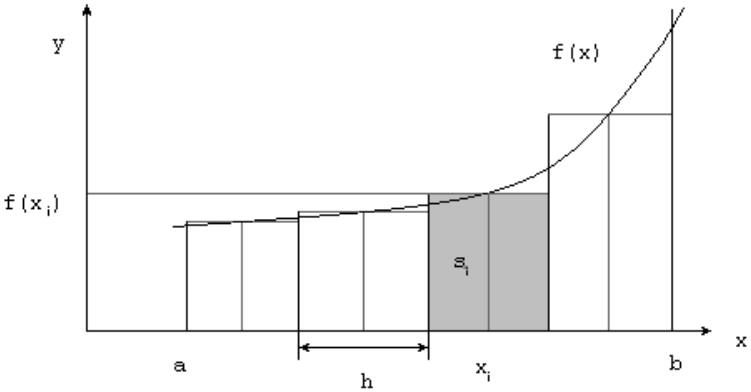


Рис. 4.1. Вычисление интеграла методом прямоугольников

Вычислим площадь одного прямоугольника $s_i = h*f(x_i)$. Сумма s_i площадей полученных прямоугольников является приближенным значением интеграла: $S1 = \sum_{i=1}^{n} s_i = h\sum_{i=1}^{n} f(x_i)$.

Однако одно приближение не позволяет оценить точность, с которой вычислено значение интеграла, необходимо найти следующее приближение. Для этого увеличим n в два раза, т. е. $n = 2n$. Аналогичнонайдем $S2 = h\sum_{i=1}^{n} f(x_i)$.

Требуется вычислить значение интеграла с точностью ε, поэтому проверим условие $|S1 - S2| < \varepsilon$. Если условие выполняется, то *S*2 принимается за искомое значение интеграла; если не выполняется, то последнее выполненное значение *S*2 считается предыдущим, т. е. *S*1 = *S*2. После этого удвоим число точек деления отрезка и вычислим новое значение *S*2. Процесс удвоения *n* и вычисления *S*2 будем продолжать до тех пор, пока модуль разности S1 и *S*2 не станет меньше ε.

Пример 4.1. Вычисление интеграла $\int_0^{3/2\pi} (1/(5-3\cos x))dx$ методом прямоугольников.

```
#include "stdafx.h"
#include <iostream>
#include <math.h>
#include <conio.h>
using namespace std;

const double Pi=3.1415926;
int _tmain (int argc, _TCHAR* argv[])
{
 unsigned long i, n = 4;
 double a,b,x,h,S1,S2,eps,exact;
 a = 0;    b = 3/ (2*Pi);    eps = 0.001;
 S1 = 0;
 h = (b - a)/n;
//Вычисляем сумму в первом приближении
 for (i = 1; i<=n; i++)
{ x = a + i*h - h/2;
  S1 = S1+ (1/ (5 - 3*cos (x)))*h;
}
// Вычисляем текущее приближение
// и сравниваем его с предыдущим
 do
{ n = 2*n;
  h = (b - a)/n;
  S2 = 0;
  for (i = 1; i<=n; i++)
{ x = a + i*h - h/2;
  S2 = S2 + (1/ (5 - 3*cos (x)))*h;
}
  exact =  fabs (S1 - S2);
  S1 = S2;
} while (exact>eps);
 cout << "S = " << S2;
_getch();
 return 0;
}
```

Вычисление по формуле Симпсона путем деления отрезка [a,b] на множество более мелких отрезков

Для нахождения интеграла $\int_a^b f(x)dx$ вычислим площадь под графиком функции, являющейся подынтегральным выражением (рис. 4.2). Здесь a и b – пределы интегрирования; $x_i = a + i\,(b-a)/n$.

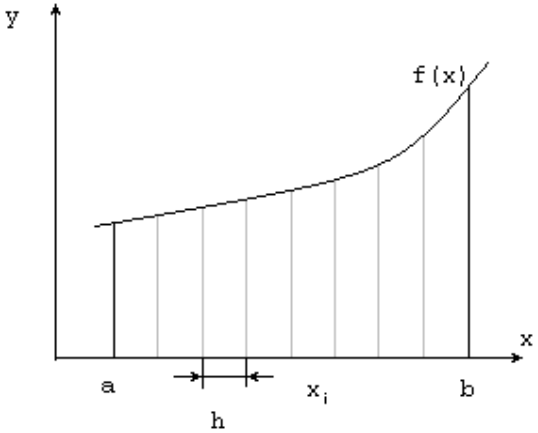


Рис. 4.2. Вычисление интеграла по формуле Симпсона

Для использования формулы Симпсона разбиваем отрезок [*a*, *b*] на *n* (четное) более мелких отрезков.

Формула Симпсона имеет вид:

$$\int_a^b f(x)dx = \frac{b-a}{3n}\big(f(a)+4f(x_1)+2f(x_2)+4f(x_3)+\ldots$$
$$+2f(x_n-2)+4f(x_n-1)+f(b)\big)$$

Здесь *n* – четное число делений интервала интегрирования; $x_i = a + i\,(b - a)/n$.

Алгоритм состоит в циклическом выполнении расчетов $f\,(x_i)$. При этом следует отдельно рассмотреть случаи для границ интегрирования $f(a)$ и $f\,(b)$ и учесть, что при нечетном номере вычисляемого элемента значение функции умножается на 4, при четном – на 2. При конечных значениях отрезка умножение не производится.

Пример 4.2. Вычисление интеграла $\int_0^{1.8} 1/(1+\sqrt{x})dx$ по формуле Симпсона.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <math.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
 setlocale (LC_ALL, "Russian"); // подключение русификатора
 unsigned long i, n;
```

```
 double a,b,x,h,y,s;
 cout << "Четное количество делений -> ";
 cin >> n;
 a = 0;      b = 1.8;
 s = 0;      x = a;
 h = (b – a)/n;
 for (i = 0; i <= n; i++)
  { y = 1/ (1+sqrt (x));
    x = x + h;
    if (i % 2 != 0)
        s +=4*y;
    else if (i == 0 || i == n)
            s += y;
        else s += 2*y;
}
 s*=h/3;
 cout << "S = " << s;
_getch();
 return 0;
}
```

Вычисление с заданной точностью ε корня уравнения F (x)=0 методом простых итераций

Пусть корень уравнения находится на отрезке [a, b] (рис. 4.3). Для использования метода итераций исходное уравнение $F(x) = 0$ нужно привести к виду $x = f(x)$. Если известно начальное приближение к корню $x = x_1$, то подставив его в правую часть уравнения $x = f(x)$, получим новое приближение $x_2 = f(x_1)$. Затем аналогичным образом получим $x_3 = f(x_2)$,..., $x_{k+1} = f(x_k)$.

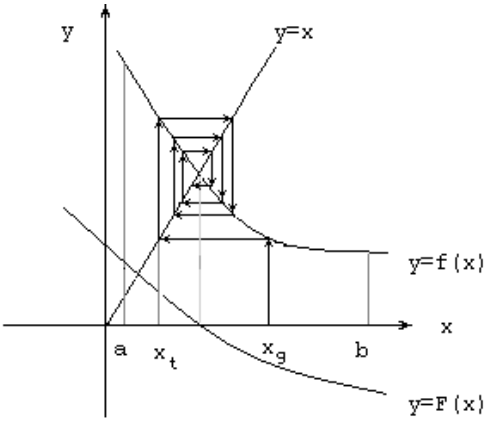


Рис. 4.3. Метод простых итераций

Итерационный процесс сходится к корню уравнения, если $|f'(x)| < 1$ на отрезке, содержащем корень уравнения. Если выполняется неравенство $-1 < f'(x) < 0$, то корень уравнения всегда находится на отрезке $[x_k, x_{k+1}]$ или $[x_{k+1}, x_k]$ и условие окончания итерационного процесса имеет вид неравенства $|x_{k+1} - x_k| < \varepsilon$.

Переход от уравнения $F(x) = 0$ к уравнению $f(x)$ можно осуществить таким образом. Умножим левую и правую части уравнения $F(x) = 0$ на некоторую константу h и добавим к обеим частям уравнения неизвестное x. Эти действия не изменяют корней уравнения:

$$hF(x) + x = 0*h + x; hF(x) + x = x.$$

Обозначив $f(x) = h\,F(x) + x$, перейдем к уравнению $x = f(x)$. Величину h необходимо выбрать такой, чтобы выполнялись неравенства $|f'(x)| < 1, f'(x) < 0$ на отрезке, содержащем корень уравнения.

Исходными данными для программы, соответствующей приведенному алгоритму, являются начальное приближение к корню и точность его вычисления. Условием выхода из итерационного процесса служит неравенство $|x_g - x_t| < \varepsilon$, при этом искомым значением является x_t:

Пример 4.3. Решение уравнения $x^3 - 2x^2 - 3 = 0$ методом итераций.

Преобразуем уравнение к виду $x = -0,2(x^3 - 2x^2 - 3) + x$.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
#include <math.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
 setlocale (LC_ALL, "Russian"); // подключение русификатора
 double xt, xg = 2.3, eps = 0.001, x = 5*eps;
 int step = 0;
 while (fabs (x)>=eps)
{ xt = -0.2* (xg*xg*xg – 2*xg*xg -3)+xg;
  x = xt-xg;
  step++;
  xg = xt;
}
 cout << "Корень=" << xt
      << " и получен на шаге " << step;
 _getch();
 return 0;
}
```

Решение уравнения f (x) = 0 с заданной точностью ε методом деления отрезка пополам

Метод деления отрезка пополам заключается в следующем. Проверяется наличие корня на отрезке [*a*, *b*] (рис. 4.4).

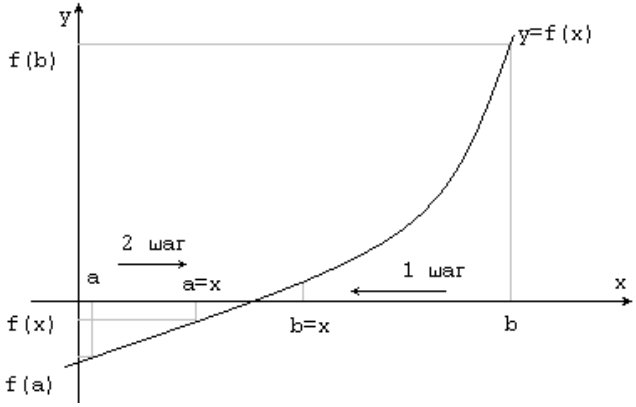


Рис. 4.4. Метод деления отрезка пополам.

Для этого вычисляются значения функций *f* (*a*) и *f* (*b*). Если *f* (*a*)**f* (*b*) > 0, то не гарантируется, что уравнение имеет корни на заданном отрезке. Предлагаемый метод не применим. Если *f* (*a*)**f* (*b*) < 0, т. е. на концах отрезка [*a*, *b*] функция *f* (*x*) имеет противоположные знаки, то искомый корень лежит на этом отрезке. Для поиска корня находим в точке *a* значение функции $y_1 = f(a)$. Затем определяем значение *x* как среднюю точку между *a* и *b*, вычисляем значения $y_2 = f(x)$. Теперь, если *f* (*a*)**f* (*x*)>0, то корень находится на отрезке [*x*, *b*], иначе – на отрезке [*a*, *x*]. В соответствии с этим перемещаем точку *a* вправо или точку *в* влево, выполняя, соответственно присваивание *a* = *x* или *b* = *x*. Таким образом, получаем второй отрезок [*a*, *b*], но вдвое меньший предыдущего. Процесс деления отрезка пополам продолжаем до тех пор, пока длина отрезка [*a*, *b*] не станет меньше заданной точности. После этого вычисляем значение *x* = (*a*+*b*)/2.

Пример 4.4. Решение уравнения $x^3 - 2x^2 - 3 = 0$ с заданной точностью eps = 0.01 методом деления отрезка пополам, если корень находится на отрезке [1, 3].

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
#include <math.h>
#include <stdlib.h>
```

```
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
  setlocale (LC_ALL, "Russian"); // подключение русификатора
  int n = 0;
  double x, a = 1, b = 3, y1, y2, eps = 0.001, r, l;
  l = a; r = b;
  y1 = a*a*a - 2*a*a - 3;
  y2 = b*b*b - 2*b*b - 3;
  if (y1*y2>0)
{ cout << "Корней нет" <<endl;
 _getch();
  exit (1) ;
}
  do
{  ++n;
  x = (a+b)/2;
  y1 = a*a*a - 2*a*a - 3;
  y2 = x*x*x - 2*x*x - 3;
  if (y1*y2>0)
a = x;
  else b = x;
} while ( (b - a)>eps);
 x = (a + b)/2;
 cout << "Корень уравнения на отрезке "
      << l << ", " << r << "  равен  " << x
      << " и получен за " << n <<" шагов";
 cout << endl;
_getch();
 return 0;
}
```

Контрольные вопросы

1. Каким образом цикл **while** может имитировать цикл **for**?
2. Каким образом цикл **while** может имитировать цикл **do-while?**
3. В каких случаях используются операторы **break**, **continue**, **exit?**
4. Почему в языке С++ нет необходимости использовать оператор **goto?**

Задания

Ввести с клавиатуры *x* и точность вычисления *Eps*. Вычислить с заданной точностью сумму.

Таблица 4.1. Простые варианты

Вариант	*Задание*
1	$S(x) = \sum_{k=1}^{\infty} (-1)^k x^k / k, \ \lvert x \rvert < 1$

Вариант	*Задание*
2	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+3}/(x^2+k^3)$
3	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+1}/(x^3k^2)$
4	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+1}\frac{k^2}{x*(k+1)^3}$
5	$S(x)=\sum_{k=1}^{\infty}(-1)^{k}\frac{x^k}{k!},\quad \lvert x\rvert<1$
6	$S(x)=\sum_{k=1}^{\infty}\frac{x^{k+1}}{(k+1)!},\quad \lvert x\rvert<1$
7	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^{2k}}{(2k)!},\quad \lvert x\rvert<1$
8	$S(x)=\sum_{k=2}^{\infty}(-1)^{k-1}\frac{x^k}{(k-1)},\quad \lvert x\rvert<1$
9	$S(x)=\sum_{k=1}^{\infty}(-1)^{k}\frac{x^{2k+1}}{(2k+1)},\quad \lvert x\rvert<1$
10	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^k}{(k+3)!},\quad \lvert x\rvert<1$
11	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+2}\frac{x^2}{(2k)^3}$
12	$S(x)=\sum_{k=1}^{\infty}(-1)^{k-1}\frac{2x}{k(k+1)},\quad \lvert x\rvert<1$
13	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+2}\frac{kx+2}{kx^2},\quad \lvert x\rvert>1$
14	$S(x)=\sum_{k=1}^{\infty}(-1)^{k}\frac{x}{k!}$
15	$S(x)=\sum_{k=1}^{\infty}(-1)^{k+2}\frac{x+2}{(k+2)!}$

Таблица 4.2. Сложные варианты

Вариант	*Задание*
1	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x + \ln(x + 0.5) - 0.5 = 0$ на интервале [0, 2]. Абсолютная погрешность не превышает 10^{-3}. Сравнить методы вычисления по количеству итераций
2	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $2x^3 + 4x - 1 = 0$ на интервале [0, 0.5]. Абсолютная погрешность не превышает 10^{-4}. Сравнить методы вычисления по количеству итераций
3	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $1/x = \sin x$ на интервале [0.1, 2]. Абсолютная погрешность не превышает 10^{-4}. Сравнить методы вычисления по количеству итераций
4	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_0^{\pi/3} \sin x\,dx$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций
5	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x^4 + 2x^3 - x - 1 = 0$ на интервале [0, 1]. Абсолютная погрешность не превышает 0.0015. Сравнить методы вычисления по количеству итераций
6	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x^3 + 12x - 2 = 0$ на интервале [0.1, 1]. Абсолютная погрешность не превышает 0.0015. Сравнить методы вычисления по количеству итераций
7	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_0^1 dx/(1+x^2)$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций
8	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x^5 - x - 0.2 = 0$ на интервале [0.9, 1.1]. Абсолютная погрешность не превышает 0.0001. Сравнить методы вычисления по количеству итераций
9	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $5x + 8\ln x - 1 = 0$ на интервале [0.5, 1]. Абсолютная погрешность не превышает 0.0015. Сравнить методы вычисления по количеству итераций

Вариант	*Задание*
10	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_0^{\pi} \cos x/(x+1)dx$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций
11	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x^3 - 2x^2 + x - 3 = 0$ на интервале [2.1, 2.2]. Абсолютная погрешность не превышает 0.001. Сравнить методы вычисления по количеству итераций
12	Методом деления отрезка пополам и методом итераций найти приближенное значение корня уравнения $x^3 + x^2 - 3 = 0$ на интервале [0.5, 3]. Абсолютная погрешность не превышает 0.001. Сравнить методы вычисления по количеству итераций
13	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_{1.0}^{8.0} (x\sqrt[3]{1+x})dx$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций
14	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_0^{3/2\pi} \frac{dx}{5-3\cos x}$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций
15	По формуле Симпсона и методом прямоугольников вычислить приближенное значение интеграла $\int_2^{2.5} \frac{dx}{x\ln^2 x}$. Точность не превышает 0.001. Сравнить методы вычисления по количеству итераций

Лабораторная работа 5. Составление программ с использованием массивов

Массив – это множество однотипных переменных, занимающее смежные ячейки памяти и обозначенное одним именем.

Одномерные массивы

Одномерный массив, или вектор – это переменные, совместно использующие одно и то же имя (имя массива). В одномерном массиве доступ к отдельной переменной осуществляется по индексу (порядковому номеру).

Синтаксис для объявления одномерного массива:

```
тип_элемента_массива имя_массива [число_элементов];
```

Пример 5.1. Объявление одномерных массивов

```
int MyArr [10];
char Literal [31];
double x_arr [100];
```

При объявлении одномерных массивов в языке C++ необходимо соблюдение правил:

1) в объявлении массива указывается количество элементов;

2) индекс первого элемента массива равен 0 (это значение нельзя изменить или переопределить);

3) индекс последнего элемента определяется как количество элементов минус 1.

Доступ к элементам массива осуществляется через имя массива и индекс элемента, указываемый в квадратных скобках: a[1], b[25].

При работе с массивами рекомендуется проверять допустимость значения индекса массива. Допустимыми являются значения индексов в диапазоне от 0 до **число_элементов** – 1.

Ввод массива осуществляется с использованием цикла.

Пример 5.2. Ввод массива с клавиатуры.

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <math.h>
#include <conio.h>
using namespace std;

const int M = 10;
int _tmain (int argc, _TCHAR* argv[])
```

```
{ //объявление действительного массива из 10 элементов:
  double x[M];
  for (int i= 0;  i<M;  i++)
   { cout << "x[" << i << "] : ";
     cin >> x[i]; // ввод элемента массива с клавиатуры
}
   // вывод массива на экран:
 for (int i= 0;  i<M;  i++)  cout << setw (5) << x[i];
 cout << endl;
 _getch();
 return 0;
}
```

Для заполнения массива значениями можно использовать генератор случайных чисел.

Пример 5.3. Ввод массива с помощью генератора случайных чисел.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

const int N = 100;

int _tmain (int argc, _TCHAR* argv[])
{
 srand ( (unsigned)time (NULL));
 int arr[N];
 for (int  i=0; i<N; i++)
     { arr[i]=rand()%100;
        cout << arr[i] << ' ';
     }
 _getch();
 return 0;
}
```

Здесь формируется массив arr, состоящий из N элементов, пронумерованных от 0 до N-1 и содержащий целые значения в диапазоне от 0 до 99. При необходимости получения с равной вероятностью положительных и отрицательных значений элементов матрицы из сгенерированного значения вычитают число, равное половине аргумента функции random:

```
arr[i] = rand()%100 – 50;
```

Инициализация элементов массива осуществляется указанием его начальных значений в фигурных скобках.

Пример 5.4. Инициализация одномерного массива

```
.  .  .
 const int M=10;
double x[M]= {12.2, 45.4, 67.2, 12.2, 34.6, 87.4, 83.6,
12.3, 14.8, 55.5};
.  .  .
```

Если начальных значений задано меньше, чем элементов в массиве, компилятор присвоит оставшимся элементам массива значение 0. Если количество начальных значений больше, чем число в квадратных скобках, компилятор выдаст сообщение об ошибке.

Точное число начальных значений можно не указывать, так как язык C++ позволяет задавать размер массива автоматически, используя количество элементов в соответствующем списке начальных значений. Следовательно, число в квадратных скобках при описании массива может отсутствовать. В этом случае размер массива определит компилятор, например, double x[ ] = {1, 2, 3};

Контрольные вопросы

1. Позволяет ли язык C++ изменять размер массива?
2. Необходимо ли использовать для массивов только предопределенные типы?
3. Укажите, допустимы ли объявления:

```
int hats [3] = {10, 20, 30};
int caps [ ] = {5, 7, 9}
```

Задания

Таблица 5.1. Простые варианты

Вариант	*Задание*
1	Дан массив int mas[15]. Заменить минимальный элемент этого массива на сумму предыдущих элементов, максимальный – на сумму последующих элементов этого массива
2	Дан массив, состоящий из 15 целочисленных значений. Определить, симметричен ли он
3	Дан массив int mas[15]. Определить, что стоит правее – максимальный или минимальный элемент
4	Сдвинуть все элементы 15-элементного целочисленного массива на две позиции вправо
5	В массиве из 20 целых чисел найти сумму и количество чисел, больших введенного с клавиатуры числа и меньших этого же числа
6	Дан целочисленный массив, состоящий из 15 элементов. Сформировать другой массив, в который поместить сначала числа меньшие среднего арифметического значения этого массива, затем – большие
7	В массиве из 15 вещественных чисел поменять местами его максимальное и минимальное значения

Вариант	*Задание*
8	В массиве из 20 целых чисел найти что больше: среднее арифметическое значение положительных элементов или модуль среднего арифметического значения отрицательных элементов
9	Дан массив int mas[15]. «Перевернуть» его
10	Даны два целочисленных массива. Определить, в каком из них больше сумма максимального и минимального элементов
11	Определить, упорядочен ли массив из 15 вещественных чисел по возрастанию
12	Создать массив из 20 символьных значений. Сформировать из его значений другой массив, расположив в нем сначала цифры, затем – буквы, затем – символы, не являющиеся ни буквами, ни цифрами
13	Сдвинуть все элементы 10-элементного целочисленного массива на одну позицию влево
14	Даны два целочисленных массива a[n] и b[n]. Создать из их элементов третий массив c[n], руководствуясь принципом c[i]=max (a[i], b[i])

Таблица 5.2. Варианты повышенной сложности

Вариант	*Задание*
1	Определить, является ли данный целочисленный массив упорядоченным по убыванию. Если да, то заменить его минимальный элемент на сумму предшествующих элементов, если нет, то заменить его максимальный элемент на сумму последующих элементов
2	Создать массив из номеров нулевых элементов исходного массива и сжать его, выбросив все нулевые элементы
3	В массиве вещественных чисел определить, сколько раз меняется знак и вывести номера позиций, в которых происходит смена знака
4	В массиве целых чисел найти наиболее часто встречающееся число. Если таких чисел несколько, определить наименьшее из них
5	В массиве целых чисел найти сумму чисел, находящихся между минимальным и максимальным элементами, включая и сами эти числа
6	Не используя других массивов переставить элементы массива в обратном порядке
7	В массиве вещественных чисел определить наибольшее количество последовательно расположенных положительных чисел

Вариант	*Задание*
8	В массиве вещественных чисел определить наибольшее количество последовательно расположенных чисел, образующих «пилу» (например, 3, 7, 5, 9, 2, 4, 1, 6)
9	Заменить повторяющиеся элементы исходного массива нулевыми значениями
10	Объединить два упорядоченных массива в один, не нарушая упорядоченности
11	Написать программу поиска элемента в упорядоченном массиве, путем деления массива пополам (бинарный поиск)
12	В массиве из 20 символов найти длины максимальной буквенной последовательности и максимальной цифровой последовательности
13	Определить, является ли символьный массив Short[5] подмассивом символьного массива Long[20]
14	Написать программу сортировки символьного 10-элементного массива (любым из методов)

Лабораторная работа 6. Составление программ с использованием двумерных массивов

Многомерные массивы

В многомерном массиве каждый дополнительный индекс обеспечивает дополнительное средство доступа к конкретному элементу, или дополнительное измерение. Наиболее распространенным видом многомерного массива являются двумерные массивы, или матрицы.

Общий синтаксис для объявления двумерных и трехмерных массивов:

```
тип_элемента  имя_массива [N_1] [N_2];
тип_элемента  имя_массива [N_1] [N_2] [N_3];
```

Как и в одномерных массивах, каждый индекс имеет нижнюю границу, равную 0, а число элементов для каждого уровня индекса определяется при объявлении многомерного массива.

Пример 6.1. Объявление многомерных массивов

```
double matrix A [100][10];
char table [41][22][3];
int index [7][12];
```

Большинство компиляторов хранит элементы многомерного массива друг за другом, т. е. как длинный одномерный массив. Исполняемый модуль вычисляет, где расположен искомый элемент в этом массиве.

Однако заполнение и обработку многомерных массивов чаще всего производят пользуясь вложенными циклами.

Язык C++ дает возможность инициализировать многомерный массив способом, аналогичным инициализации одномерных массивов. Для этого нужно использовать список значений, расположенных в той же последовательности, в которой элементы многомерного массива хранятся в памяти ЭВМ.

Пример 6.2. Инициализация матрицы

```
.  .  .
int main ( )
    {  const int COL=4;
       const int ROW=3;
       double x[ROW][COL]= {{ 1,  2,  3,  4}, // строка 1
                            { 5,  6,  7,  8}, // строка 2
                            { 9,10,11,12}};   // строка 3
       .  .  .
       return 0   };
```

Пример 6.3. Ввод многомерных массивов с клавиатуры.

```
        .  .  .
```

```
for (i=0; i<MAX_ROW; i++)
           { for (j=0; j<MAX_COL; j++)
                   { cout << "x[" << i <<"][" << j<<"]=";
                 cin >> x[i][ j];   }
           }
            .  .  .
```

Пример 6.4. Ввод многомерных массивов с помощью генератора случайных чисел.

```
            .  .  .
#include <time.h>
time_t t;
srand (time (&t));     //<stdlib.h>
            .  .  .
    for (i=0; i<MAX_ROW; i++)
      {   for (j=0; j<MAX_COL; j++)   x[i][ j]=rand()%10;
            }
            .  .  .
```

Вывод массивов также осуществляется с помощью циклов, одиночных для одномерных массивов или вложенных для многомерных массивов.

При выводе удобно использовать манипулятор setw, позволяющий задать ширину поля вывода данных. Ширина задается в скобках после названия манипулятора и определяет количество знакомест (символов), которые отводятся под выводимые данные. Для его подключения необходим заголовочный файл <iomanip>.

Пример 6.5. Вывод матриц

```
    .  .  .
for (i=0; i<MAX_ROW; i++)
            {  for (j=0; j<MAX_COL; j++)
                     { cout << setw (5) << x[i][ j];
                     }
       cout << endl; }
```

При обработке матриц чаще всего встречаются такие типы задач:

1) работа с матрицей в целом;

2) работа со строкой (столбцом) матрицы;

3) работа с диагональными элементами.

При работе с матрицей в целом, в качестве объекта для сравнения выбирают элемент с индексами (0,0), затем, последовательно перебирая все элементы матрицы, выполняют необходимые действия. Перебор элементов происходит с использованием вложенных циклов.

Пример 6.6. Определить среднее арифметическое значение элементов матрицы x[6][7] целого типа.

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
```

```
#include <time.h>
#include <iomanip>
using namespace std;

int main()
{   const int MAX_COL=6;
    const int MAX_ROW=7;
    int x[MAX_ROW][MAX_COL];
int i,j,sum=0;
float sred;
time_t t;
setlocale (LC_ALL, "Russian"); // подключение русификатора
srand (time (&t)); // инициализация генератора
                   // случайных чисел
for (i=0; i<MAX_ROW; i++) // Заполнение массива случайными
for (j=0; j<MAX_COL; j++) // числами
x[i][j]=rand()%10;
for (i=0; i<MAX_ROW; i++) // Вывод матрицы на экран
{  cout << endl;
    for (j=0; j<MAX_COL; j++)
{   cout << setw (5) << x[i][j]; // и вычисление суммы
sum+=x[i][j];                    // ее элементов
}
   }
sred= float (sum)/ (MAX_ROW*MAX_COL);
cout << "\nСумма = " << sum
     << "\nСреднее арифметическое = " << sred;
 _getch();
return 0;
}
```

Для поиска элемента строки (столбца) объектом сравнения является первый элемент строки с индексами i,0 или столбца с индексами 0,j и обработка идет до конца строки (столбца). Обычно результатом такой обработки массива является вектор-столбец или вектор-строка.

Пример 6.7. Определить минимальный элемент каждой строки действительной матрицы matr [8][4].

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#include <iomanip>
using namespace std;

int main()
{ const int COL=4;
   const int ROW=8;
   float matr[ROW][COL]; //объявление матрицы
                         // вещественного типа
   int i,j;
   float min[ROW];
   time_t t;
   srand (time (&t));
```

```
   for (i=0; i<ROW; i++) // заполнение и вывод матрицы
        { for (j=0; j<COL; j++)            // на экран
            { matr[i][j]=rand()%100;
               cout << setw (6) << matr[i][j];
            }
      cout << endl;
  }
   for (i=0; i<ROW; i++)   // формирование вектора-столбца
        { min[i]=matr[i][0]; // из минимальных элементов
                             // строк
     for (j=0; j< COL; j++)
                if (matr[i][j]<min[i]) min[i]=matr[i][j];
     cout << "min= " << min[i];
     }
   cout <<endl ;
   for (i=0; i< ROW; i++) cout << setw (6) << min[i];
    _getch();
}
```

При работе с диагональными элементами учитывают, что индексы элементов, стоящих на главной диагонали, удовлетворяют условию $i = j$.

Элементы, стоящие на побочной диагонали, имеют индексы, удовлетворяющие условию $i=n-j-1$, если n – порядок матрицы (предполагается, что матрица – квадратная).

Пример 6.8. Найти количество четных элементов целочисленной квадратной матрицы порядка 9, стоящих ниже побочной диагонали.

```
#include <iostream>
#include <stdlib.h>
#include <conio.h>
#include <math.h>
#include <time.h>
#include <iomanip>
using namespace std;

void main()
{   const int RANG=9;
    int matr[RANG][RANG];
    int i,j,kol=0;
    time_t t;
    srand (time (&t));
    setlocale (LC_ALL, "Russian");
    for (i=0;i<RANG;i++)
      { for (j=0;j<RANG;j++)
         { matr[i][j]=rand()%10;
           cout << setw (4) << matr[i][j];
         }
         cout <<endl;
      }
    for (i=0; i<RANG; i++)
      { for (j=0; j<RANG; j++)
         if ( (i>RANG-j-1) && (matr[i][j]%2==0))
            kol++; }
```

```
        }
    cout << "Количество четных элементов
         << под главной диагональю равно "
         << kol;
   _getch();
}
```

Контрольные вопросы

1. Позволяет ли язык C++ изменять размер массива?
2. Как расположены в памяти элементы многомерных массивов?
3. Необходимо ли использовать для массивов только предопределенные типы?
4. Как объявить двумерный массивы?

Задания

Таблица 6.1. Простые варианты

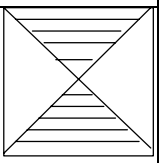
Вариант	*Задание*
1	Дана целочисленная матрица размером 6×9, содержащая как положительные, так и отрицательные элементы. Найти минимальный элемент матрицы и среднее арифметическое значение положительных элементов матрицы
2	Дана целочисленная матрица 7×10. Поменять местами максимальное и минимальное значения матрицы
3	Дана целочисленная матрица 5×8. Определить, что больше: сумма четных элементов правой половины матрицы или сумма нечетных элементов левой половины матрицы
4	Дана целочисленная матрица 7×7. Найти сумму положительных элементов, стоящих над главной диагональю и количество нечетных элементов, стоящих под побочной диагональю
5	Дана целочисленная матрица *mxk*. Найти минимальное значение для верхней половины матрицы и максимальное значение для нижней половины
6	Дана целочисленная матрица *mxm*. Найти максимальное значение для ее первой четверти и минимальное – для третьей четверти
7	Дана матрица 6×7, элементами которой являются значения символьного типа. Подсчитать количество буквенных символов в матрице и количество цифровых символов в ее верхней половине
8	Дана целочисленная матрица 9×9. Определить количество нулевых элементов над побочной диагональю и сумму четных элементов под главной диагональю

Вариант	*Задание*
9	Дана целочисленная матрица 5×7. Найти минимальное значение среди четных элементов матрицы и сумму положительных элементов матрицы
10	Дана матрица 5×7, элементами которой являются значения символьного типа. Найти максимальное значение среди буквенных символов и минимальное среди цифровых значений
11	Дана действительная матрица 7×7. Найти сумму элементов, попадающих в заданный с клавиатуры интервал. Найти количество элементов, не попадающий в этот интервал
12	Дана матрица 4×5, элементами которой являются значения символьного типа. Заменить на символ, введенный с клавиатуры, значения символов, больших, чем это значение, и подсчитать количество таких замен
13	Дана целочисленная матрица 4×7. Определить минимальное значение матрицы. Найти среднее арифметическое значение модулей отрицательных элементов в матрице
14	Дана целочисленная матрица размером 4×6. Получить новую матрицу путем умножения всех элементов на наименьший по модулю элемент

Таблица 6.2. Варианты повышенной сложности

Вариант	*Задание*
1	Дана целочисленная матрица размера 6×9, содержащая как положительные, так и отрицательные элементы. Сформировать одномерные массивы, состоящие из средних арифметических значений положительных элементов каждого столбца матрицы и средних арифметических значений нечетных элементов каждой строки матрицы
2	Дана целочисленная матрица 7×10. Поменять местами максимальное и минимальное значения матрицы. Найти суммы элементов тех столбцов, где находятся эти значения
3	Дана целочисленная матрица 5×8. Найти сумму четных элементов каждого столбца правой половины матрицы и среднее арифметическое значение нечетных элементов каждого столбца левой половины матрицы. Сформировать соответствующие одномерные массивы

Вариант	*Задание*
4	Дана целочисленная матрица 4×7. Определить минимальное и максимальное значения матрицы и их местоположение. Найти среднее арифметическое значение положительных элементов и модулей отрицательных элементов в каждом столбце матрицы. Результаты вычислений записать в одномерные массивы
5	Дана целочисленная матрица *m*x*k*. Найти минимальное значение для правой половины матрицы и максимальное значение для левой половины. Сформировать новую матрицу путем прибавления к каждому элементу верхней половины исходной матрицы определенного минимального значения и к каждому элементу ее нижней половины определенного максимального значения
6	Дана целочисленная матрица *mxm*. Найти максимальное значение для каждого столбца матрицы и минимальное – для каждой строки матрицы. Заменить каждый элемент матрицы на значение его квадрата
7	Дана матрица 5×7, элементами которой являются значения символьного типа. Составить одномерный массив, содержащий количество буквенных символов в каждом из ее столбцов. Подсчитать количество цифровых символов над третьей строкой матрицы и общее количество символов, не являющихся ни цифрой, ни буквой
8	Дана целочисленная матрица 9×9. Определить количество нулевых элементов под главной диагональю матрицы и под ее побочной диагональю. Сформировать одномерный массив, содержащий количество отрицательных элементов каждого столбца матрицы
9	Дана действительная матрица 7×7. Найти минимальное значение среди элементов, стоящих над главной диагональю, и максимальное среди элементов, находящихся ниже главной диагонали, а также их местоположение. Сформировать одномерный массив, содержащий сумму элементов четных строк и произведение элементов нечетных строк
10	Дана целочисленная матрица размером 5×5. Получить новую матрицу путем умножения всех элементов на наименьший по модулю элемент. Сформировать одномерный массив из максимальных элементов каждой строки полученной матрицы

Вариант	*Задание*
11	Дана действительная матрица 7×7. Найти сумму элементов в заштрихованной области. Сформировать новую матрицу путем прибавления полученного значения к каждому элементу из незаштрихованной области исходной матрицы. столбце полученной матрицы. Сформировать одномермерный массив, содержащий количество четных элементов в каждом 
12	Дана целочисленная матрица 7×7. Найти сумму положительных элементов, стоящих над главной диагональю и количество нечетных элементов, стоящих под побочной диагональю. Заменить полученными значениями соответственно элементы четных и нечетных столбцов
13	Дана действительная матрица 6×8. Найти сумму элементов каждой строки верхней половины матрицы и произведение элементов каждой строки ее нижней половины. Определить значение и местоположение максимального элемента верхней половины матрицы и минимального элемента ее нижней половины
14	Дана вещественная квадратная матрица 5×5. Получить новую матрицу путем прибавления к элементам каждой строки матрицы наименьшего значения элементов этой строки. Сформировать одномерный массив, содержащий количество четных элементов в четных столбцах полученной матрицы и одномерный массив, содержащий количество нечетных элементов в нечетных столбцах

Лабораторная работа 7. Программирование задач с использованием строк

Символьная строка (далее просто строка) является массивом типа *char*, который заканчивается нуль-символом (‘\0’).

Описание переменных строкового типа

Переменную строкового типа можно описать несколькими способами.

1. Описать как массив символов:

а) char **имя_массива**[n];

где n – количество символов в строке, включая завершающий нуль-символ. Например, char stroka[10];

б) char **имя_массива**[ ];

Например, char s[ ];

В описании *а* для переменной stroka будет выделено 10 байт памяти ЭВМ. В описании *б* память выделена не будет. Это описание означает, что переменная s – это переменная строкового типа, длина которой не определена.

2. Описать с использованием указателя на тип char, например, char *stroka;

В данном описании память для переменной stroka не выделяется. Такое описание означает, что переменная stroka может содержать адрес ячейки памяти первого символа строки. Выделить память для переменной stroka можно с помощью функции new:

```
char *stroka=new char[20];
```

где 20 – объем памяти, выделенной для строки.

Инициализация переменных строкового типа

Задать значение переменным строкового типа можно с помощью строковых констант:

а) char stroka [10]="строка";

б) char stroka [ ]="строка".

В первом случае под переменную stroka отводится 10 байт, но используются только первые 7 байт (включая завершающий нуль-символ ‘\0’). Во втором случае переменная stroka занимает 7 байт, т. е. для этой переменной память выделяется автоматически и ее размер определяется количеством символов строковой константы плюс завершающий нуль-символ.

Инициализация массива строк

Общий вид:

```
char имя_массива[n][m]={ строковая константа1,
                         строковая константа2,
       . . .
                         строковая константаn};
```

где n – количество элементов массива, т. е. количество строк; m – длина строки.

Пример 7.1

```
char mas_words [10][80];
char mas_words [3][10] ={ "первое",
                          "второе",
                          "третье"};
```

Ввод строки

Операция >> вводит строку из (MAX_LEN-1)-символов, не содержащую пробелов. Значащих символов в строке может быть не более, чем (MAX_LEN-1), так как строка обязательно содержит завершающий нулевой байт '\0', который служит признаком завершения строки. Особенность операции >> состоит в том, что пробел она воспринимает как нулевой символ, т. е. как признак конца строки.

Если строка содержит пробелы, то для ее считывания используется метод класса istream cin.get(). Мы пока не будем останавливаться на подробностях использования методов классов – речь об этом пойдет в лабораторной работе 13, связанной с объектно-ориентированным программированием. Для работы со строками достаточно знать, что оператор

```
cin.get (str,MAX_LEN);
```

позволяет ввести с клавиатуры строку str, не превышающую (MAX_LEN-1) символов.

Пример 7.2. Ввод строки, содержащей пробелы.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ const int MAX_LEN = 80; // максимальная длина строки
  char str[MAX_LEN];
  cout << "Input string  : ";
  cin.get (str,MAX_LEN);
  cout << "Inputed string  :" << str << endl;
  _getch();
  return 0;
}
```

Для ввода строки можно также воспользоваться методом getline.

Пример 7.3. Ввод строки, содержащей пробелы.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ const int MAX_LEN = 80; // максимальная длина строки
  char str[MAX_LEN];
  cout << "Input string  : ";
  cin.getline (str,MAX_LEN);
  cout << "Inputed string  : " << str << endl;
  _getch();
  return 0;
}
```

Методы работают почти одинаково. Различие состоит в том, что метод getline считывает из входного потока не более (MAX_LEN-1)-символов и записывает их в указанную строковую переменную, заменяя символ перевода строки \n нулевым байтом. Символ перевода строки при этом из потока удаляется.

Метод get оставляет символ перевода строки в потоке. Поэтому при попытке ввода нескольких строк, как показано в примере 7.4, будет введена лишь первая строка. Вместо всех остальных окажутся пустые строки, так как символ \n, оставленный в потоке методом get при вводе первой строки, будет обнаружен следующим методом get и интерпретируется как ввод пустой строки. При этом сам символ \n так и остается в потоке.

Пример 7.4. Попытка ввода нескольких строк.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ const int MAX_LEN = 80; // максимальная длина строки
  char str[MAX_LEN];
  cout << "Input string  : ";
  cin.get (str,MAX_LEN);  // или так:
  cin.get(); // cin.getline (str,MAX_LEN);
  cout << "Inputed string  : " << str << endl;
  cout << "Input string  : ";
  cin.get (str,MAX_LEN);
  cin.get();
  cout << "Inputed string  : " << str << endl;
  cout << "Input string  : ";
  cin.get (str,MAX_LEN);
  cin.get();
```

```
  cout << "Inputed string  : " << str << endl;
  _getch();
  return 0;
}
```

Удаление символа \n из потока может быть осуществлено вызовом метода get() без параметров.

В примере 7.4 можно воспользоваться методом getline вместо метода get. Тогда эффект потери строк не возникает, и нет необходимости специально удалять символ \n из потока.

Современные версии языка C++ позволяют вводить несколько строк, пользуясь методом get с тремя аргументами: имя строковой переменной, длина строки и символ завершения ввода. В этом случае символ перевода строки не будет служить признаком завершения ввода строки. Теперь строки можно вводить до тех пор, пока не будет введен символ, объявленный признаком завершения ввода, или пока не будет превышен максимальный размер строки.

Пример 7.5. Ввод нескольких строк.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{ // подключение русификатора:
  setlocale(LC_ALL, "Russian");
  // максимальная длина строки:
  const int MAX_LEN = 180;
  char str[MAX_LEN];
  cout << "Введите строку: ";
  cin.get (str,MAX_LEN,'*');
  cout << "Введенная строка:\n"
       << str << endl;
  _getch();
  return 0;
}
```

В результате получится:

Введите строку: Якорь выбран чугунный И в открытых лагунах Наши старые шхуны Снова лижет прибой. *	Введенная строка: Якорь выбран чугунный И в открытых лагунах Наши старые шхуны Снова лижет прибой.

Ввод строки можно осуществить также с помощью функции gets (char *). Эта функция читает строку со стандартного устройства ввода (клавиатуры) до тех пор, пока не встретится символ новой строки ('\n').

Функция считывает все символы до символа новой строки и присоединяет нуль-символ в конец строки.

Пример 7.6

```
char str [80];
gets (str);
char * stroka = new char [20];
gets (stroka);
```

Ввести строку можно и с помощью функции scanf с использованием формата «%s», но эта функция считывает символы до символа разделителя (пробел, табуляция, переход на новую строку) или по количеству символов в спецификации формата (%10s). Это очень неудобно, так как практически всегда требуется вводить строку с символами-разделителями. Поэтому функция scanf менее предпочтительна.

Вывод строки

Для вывода строк мы уже использовали операцию <<. Никаких особенностей при выводе этого типа данных она не имеет.

Вывод строки может осуществляться также с помощью функции puts (char *). Символы записываются в стандартный выводной поток, в конец строки добавляется символ конца строки ('\n'). Нуль-символ не выводится.

Пример 7.7

```
puts (str);
puts (stroka);
```

Вывести строку можно с помощью функции printf. В этом случае нуль-символ не выводится, но и символ '\n' в конец строки не добавляется.

Пример 7.8

```
printf ("%s", str);
printf ("%s %s\n", str, stroka);
printf ("%s\n%s\n", str, stroka);
```

Доступ к компонентам строки

При работе со строками можно обращаться и к отдельным символам в строке: str [3], str[0]. При этом следует помнить, что нумерация символов в строке начинается с 0, как и в массивах любого другого типа (int, float).

Библиотека языка C++ содержит ряд функций, работающих со строками. Прототипы этих функций находятся в файле string.h, поэтому в начале программы необходимо подключить данный файл:

```
#include <string.h>
```

Описание наиболее используемых функций работы со строками приведены в приложении к лабораторной работе.

Выделение слов из строки

Очень часто при работе со строками необходимо выделять слова, т. е. группу символов, находящуюся между символами-разделителями (пробелами, запятыми, точками и т. п.). Сформировать слово можно несколькими способами.

Формирование слова с помощью анализа компонентов строки

Пример 7.9. Пусть введена строка символов str, разделенных пробелами. Анализируя каждый символ, сформируем слово в переменной word и выведем его на печать.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <string.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
  char word[20];
  char str[80];
  int j=0;            // Количество символов в слове word
  cout << "Input string  : ";
  cin.get (str,80);
  for (int i=0;i<strlen (str);i++) // Пока не конец строки
    {
      if (str[i]==' ') continue; // Пропуск пробелов в str
      while (str[i]!=' '  &&  str[i]!='\0')
// Пока текущий символ не пробел и не нуль-символ
           word[j++]=str[i++]; // Формирование слова word
    }
   word[j]='\0';   // Нуль-символ в конец слова
   cout<<word;     // Вывод слова
   _getch();
   return 0;
}
```

Формирование слова с помощью функции strtok

Функция strtok позволяет выделять из строки слова, разделенные символами-разделителями. При первом обращении к функции необходимо указать строку, в которой происходит поиск слов, и строку из символов-разделителей. При последующем поиске вместо строки указывается NULL. К сожалению, данная функция портит содержимое исходной строки.

Пример 7.10

```
char *word;
word=strtok (str," ,.?"); // word – первое слово в строке
while (word)
{
    cout<<word;
    word=strtok (NULL," ,.?");// Следующее слово
}
```

Формирование слова с помощью функции strpbrk

Функция strpbrk находит в анализируемой строке str первое местоположение любого из заданных символов-разделителей. При этом содержимое исходной строки не изменяется.

Пример 7.11

```
char* w, *word;
w=strpbrk (str,” ,.?”);      //w указывает на
                             // символ-разделитель в str
while (w!=NULL)              // Цикл по всем словам
{
     n=w–str;                // n – длина слова
     strncpy (word,str,n);   // word – текущее слово
     word[n++]='\0';         // Нуль-символ в конец строки
     cout<<word;
     str+=n;              // Анализ следующей части строки
     w=strpbrk (str,” ,.?”); // Поиск символа-разделителя
                             // в stroka
}
```

Примеры программирования

Пример 7.12. Дана строка символов. Вывести на экран четыре самых коротких слова строки.

```
#include "stdafx.h"
#include <string.h>
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
  setlocale (LC_ALL, "Russian");
  char str[80];      // str – исходная строка
  cout << "Введите строку  : ";
  cin.get (str,80);  // Ввести строку
  char word[10][20]; // word – массив слов
  int k=0;           //k – количество слов в строке str
// Формирование массива слов word из строки str
  for (int i=0;i<strlen (str);i++) // Пока не конец строки
   {
     while (str[i]==' ') continue; // Пропуск пробелов в str
```

```
     int j=0;       // Количество символов в слове word[k]
     // Пока текущий символ не пробел и не нуль-символ
     while (str[i]!=' ' && str[i]!='\0')
           word[k][j++]=str[i++];
    // Формирование k-го слова в массиве word
     word[k++][j]='\0';    // Нуль-символ в конец слова
   }
// Сортировка слов в порядке увеличения их длины
  char* sh_word=new char[20];
    // sh_word – короткое слово из неотсортированных слов
  for (int i=0;i<k;i++) // Цикл по всем словам массива word
    {
      strcpy (sh_word,word[i]);
    // Пусть i-е слово будет самым коротким
      int num=i;    // Номер короткого слова в массиве = i
// Поиск короткого слова среди неотсортированных
// в массиве слов
      for (int j=i;j<k;j++)
// Если длина текущего слова меньше, чем длина короткого
// слова
        if (strlen (word[j])<strlen (sh_word))
          {         //Короткое слово – это текущее слово
            strcpy (sh_word,word[j]);
            num=j;  // Номер короткого слова в массиве = j
          }
      strcpy (word[num],word[i]);
    // Обмен местами i-го слова и короткого слова
      strcpy (word[i],sh_word);
    }
  for (int i=0;i<4;i++)       // Вывод на экран четырех
    cout<<word[i]<< "  ";     // самых коротких слова
  delete[ ] sh_word;          // Освобождение динамической
                              // памяти
  _getch();

return 0;
}
```

Результат выполнения программы.

```
Введите строку >один два три четыре пять шесть
два три один пять
```

Пример 7.13. Дана строка символов, разделенных пробелами, точками, запятыми. Сформировать новую строку, содержащую слова исходной строки, поставив перед самыми короткими словами в строке символ * (звездочка).

```
#include "stdafx.h"
#include <string.h>
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
```

```
{
  char* str = new char[80];       // str – исходная строка
  char* word = new char[20];      // word – слово в строке
  char* newstr = new char[80];    // newstr – новая строка
  setlocale (LC_ALL, "Russian");
  cout << "Введите строку  : ";
  cin.get (str,80);               // Ввод строки
    // Пусть длина самого короткого слова = длине строки
  int minlen = strlen (str);
    // stroka = str + 'пробел'
  char* stroka = new char[strlen (str)+2];
  strcpy (stroka,str);
  strcat (stroka," ");
  word=strtok (str," ,.?");
  while (word)            // word – первое слово в строке
    {
// Если длина короткого слова меньше длины текущего слова,
//то длина короткого слова = длине текущего слова
      if (minlen>strlen (word)) minlen=strlen (word);
      word=strtok (NULL," ,.?");        // Следующее слово
// Выделение динамической памяти
      word=new char[20];
// Формирование новой строки
      int n;
      newstr[0]='\0';   // newstr – пустая строка
      str[0]='\0';      // str – пустая строка
     // word указывает на символ-разделитель в stroka
      word=strpbrk (stroka," ,.?");
      while (word!=NULL)     // Цикл по всем словам
         {
           n=word-stroka;    // n – длина слова
           strncpy (str,stroka,n);  // str – текущее слово
           str[n++]='\0';
// Если длина текущего слова равна длине самого короткого
// слова то в newstr  заносим символ '*'
           if (strlen (str)==minlen)
              strcat (newstr,"*");
           strcat (newstr,str);
// В newstr записывается текущее слово
           strcat (newstr," ");// В newstr записывается
                                // пробел
           stroka+=n;    // Анализ следующей части строки
           word=strpbrk (stroka," ,.?");
 // Поиск символа разделителя в stroka
         }
  cout<<newstr;            // Вывод новой строки на экран
    }
  _getch();

return 0;
}
```

Результат выполнения программы.

```
Введите строку >один, два три, еще три, еще один
один *два *три *еще *три *еще один
```

Пример 7.14. Дана строка. В строке имеются символы '*'. Подсчитать количество подстрок, заключенных между символами '*'. Определить самую длинную подстроку.

```
#include "stdafx.h"
#include <string.h>
#include <iostream>
#include <conio.h>
using namespace std;
int _tmain (int argc, _TCHAR* argv[])
{
  char* str = new char[80];     // str - исходная строка
  setlocale (LC_ALL, "Russian");
  cout<<"Введите строку >";
  cin.get (str,80);       // Ввод строки
  int count = 0,          // Количество подстрок
       maxlen = 0,        // Длина наибольшей подстроки
       pos = 0,           // Номер текущей позиции в str
       j = 0;             // Счетчик символов в подстроке
  char* substr = new char[80]; // substr - текущая
                               // подстрока
  char maxstr[80];       // maxstr - наибольшая подстрока
  maxstr[0]='\0';        // maxstr - пустая строка
  for (int i=0;i<strlen (str);i++) // Цикл - по всей
                                   //исходной строке
    if (str[i]=='*')          // Если текущий символ = '*'
    {                         // Формирование подстроки
      substr[j]='\0';
      count++;      // Увеличение количества подстрок на 1
      if (count>1)  // Если это не первый символ '*'
        {
          if (maxlen <= i - pos)
// Если длина наибольшей подстроки меньше или равна
// длине текущей подстроки,то длина наибольшей подстроки
// равна длине текущей подстроки, а наибольшая подстрока
// равна текущей подстроке
                        {
                      maxlen = i - pos;
                      strcpy (maxstr,substr);
                        }
        }
      substr[0]='\0'; // substr - пустая строка
      pos=i+1;   // анализ следующей части исходной строки
      j=0;       // Установка начальных значений
    }
// Если текущий символ не '*', то переписываются текущие
// символы из исходной строки в подстроку
    else substr[j++]=str[i];
  cout<<"Количество подстрок  >"<<--count<<endl;
  cout<<"Наибольшая подстрока >"<<maxstr
      <<", длиной "<<maxlen;
  _getch();
  return 0;
}
```

Результат выполнения программы.

```
Введите строку > один *два и три *еще три *еще один
Количество подстрок >2
Наибольшая подстрока >два и три, длиной 10
```

Контрольные вопросы

1. Что представляют собой строки?
2. Каким образом строки описываются и определяются?
3. Какие функции используются для ввода строки и чем они отличаются друг от друга?
4. Какие функции используются для вывода строки?
5. Каким образом можно выделить слово из строки?
6. Где находится описание прототипов функции обработки строк ?

Задания

Таблица 7.1. Простые варианты

Вариант	*Задание*
1	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «and» запятую
2	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «no» запятую
3	Дана строка слов. Сформируйте новую строку, удалив пробелы, с которых может начинаться строка, а каждую внутреннюю группу пробелов замените одним пробелом
4	Дана строка слов, разделенных пробелами. Определите количество слов, которые встречаются более одного раза
5	Дана строка слов, разделенных пробелами. Сформируйте строку из неповторяющихся слов
6	Дана строка слов, разделенных пробелами, запятыми, точками. Сформируйте новую строку из пяти самых длинных слов
7	Дана строка символов и некоторый символ sym. Сформируйте новую строку, вставив после каждого вхождения символа sym пробел
8	Дана строка символов и некоторый символ sym. Сформируйте новую строку, вставив после каждого вхождения символа sym запятую
9	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество подстрок (заключенных между запятыми) в строке
10	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество слов в строке

Вариант	*Задание*
11	Дана строка символов, представляющих собой арифметическое выражение. Порядок операций определен слева направо. Подсчитайте результат данного выражения. Каждый операнд выражения представляет собой цифру
12	Дана строка слов, разделенных пробелами. Определите самое короткое слово
13	Дана строка слов, разделенных пробелами, запятыми, точками. Определите количество слов, заканчивающихся последней буквой алфавита (русского или латинского)
14	Дана строка слов. Сформируйте новую строку, вставив перед каждым из слов «а» и «но» запятую

Таблица 7.2. Варианты повышенной сложности

Вариант	*Задание*
1	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «and» запятую. Определите, сколько в строке симметричных слов
2	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «no» запятую. Подсчитайте количество подстрок между запятыми. Определите, сколько в этой строке слов, у которых первая буква содержится в слове более одного раза
3	Дана строка слов, разделенных пробелами. Сформируйте новую строку, удалив пробелы, с которых может начинаться строка, а каждую внутреннюю группу пробелов замените одним пробелом. Подсчитайте количество слов в данной строке и количество слов, у которых первая и последняя буквы совпадают
4	Дана строка слов, разделенных пробелами. Определите количество слов, которые встречаются более одного раза. Сформируйте строку из неповторяющихся слов
5	Дана строка слов, разделенных пробелами. Сформируйте строку из неповторяющихся слов, расположив их в алфавитном порядке
6	Дана строка слов, разделенных пробелами, запятыми, точками. Сформируйте новую строку из пяти самых длинных слов. Определите количество слов, начинающихся первой буквой алфавита (русского или латинского)

Вариант	*Задание*
7	Дана строка символов и некоторый символ sym. Сформируйте новую строку, вставив после каждого вхождения символа sym пробел. Подсчитайте количество различных слов в образовавшейся строке
8	Дана строка символов и некоторый символ sym. Сформируйте новую строку, вставив после каждого вхождения символа sym запятую. Определите самое длинное слово в строке
9	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество подстрок (заключенных между запятыми) в строке. Определите длину самого короткого слова
10	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество слов в строке и сформируйте новую строку из самых длинных слов подстрок (заключенных между запятыми)
11	Дана строка символов, представляющих собой арифметическое выражение. Порядок операций определен слева направо. Подсчитайте результат данного выражения. Каждый операнд выражения представляет собой число (не цифру)
12	Дана строка слов, разделенных пробелами. Определите самое короткое слово. Оставьте в строке только первые вхождения слов
13	Дана строка слов, разделенных пробелами, запятыми, точками. Определите количество слов, заканчивающихся последней буквой алфавита (русского или латинского). Сформируйте новую строку из трех самых коротких слов
14	Дана строка слов. Сформируйте новую строку, вставив перед каждым из слов «а» и «но» запятую. Подсчитайте количество подстрок, разделенных запятыми. Сформируйте строку из слов, с которых начинаются подстроки

Приложение. Функции работы со строками

Заголовочный файл: string.h

```
char* strcat (char * string1, char * string2);
```

добавляет строку string2 в конец строки string1, записывая в конец строки результата нуль-символ, и возвращает указатель на сцепленную строку (string1).

```
char* strchr (char* string, int sim);
```

возвращает указатель на первое местонахождение символа, имеющего код sim, в строке string. Символ sim может быть нулевым символом (‘\0’), тогда поиск ведется для нулевого символа. Функция возвращает NULL, если символ не найден.

```
int strcmp (char* string1, char* string2);
```

сравнивает строки string1 и string2 лексикографически и возвращает значение:

```
    меньше 0, если string1< string2,
    равное 0, если string1= string2,
    больше 0, если string1> string2.
```

```
char* strcpy (char* string1, char* string2);
```

копирует строку string2, включая нуль-символ, в строку string1 и возвращает значение аргумента string1.

```
int strcspn (char* string1, char* string2);
```

возвращает индекс первого символа в строке string1, который принадлежит набору символов string2. Завершающий нуль-символ не учитывается при поиске. Если string1 начинается с символа из string2, то возвращается значение 0.

```
int strlen (char* string);
```

возвращает длину в байтах строки string. Нуль-символ не учитывается.

```
char* strncat (char* string1, char* string2, int n);
```

добавляет первые n символов из строки string2 в строку string1, завершая результирующуюстроку нуль-символом. Если n больше длины строки string2, то длина строки string2 используется вместо n.

```
int strncmp (char* string1, char* string2, int n);
```

сравнивает первые n символов в строках string1 и string2 лексикографически и возвращает результат:

```
значение    < 0,  если string1< string2,
            = 0,  если string1= string2,
            > 0,  если string1> string2.
char* strncpy (char* string1, char* string2, int n);
```

копирует n символов строки string2 в строку string1. Если значение меньше, чем длина строки string2, то нуль-символ не добавляется в новую строку. Если значение n больше, чем длина строки string2, то нуль-символ добавляется в конец строки string1.

```
char* strpbrk (char* string1, char* string2);
```

находит первое вхождение в строке string1 любого символа из набора символов, содержащихся в строке string2. Завершающий нуль-символ не включается в поиск. Возвращаемое значение – указатель на первое местоположение любого символа из string2 в string1или значение NULL,если нет общих символов.

```
int strspn (char* string1, char* string2);
```

возвращает индекс первого символа строки string1, который не принадлежит набору символов, содержащихся в строке string2. Нуль-символ не рассматривается. Если строка string1 начинается с символа не из набора string2 , функция возвращает значение 0.

```
char* strstr (char* string1, char* string2);
```

возвращает указатель на первое вхождение подстроки, которая содержится в символьном массиве string2 в строке string1. Возвращает NULL, если вхождение не найдено.

```
char* strtok (char* string1, char* string2);
```

символы из string1 группируются в слова string2 – набор символов-разделителей для строки string1. При первом вызове strtok производит возврат адреса первого символа string1. Чтобы найти начало следующего слова в string1, необходимо вызвать strtok с NULL-значением аргумента string1. Набор разделителей может различаться от вызова к вызову. Возвращаемое значение является указателем на слово в строке. Все слова завершаются нуль-символом.

Функции проверки символов

Заголовочный файл: ctype.h

int isalnum (int c); проверяет символ «c» на латинскую букву или цифру. Возвращаемое значение равно 0, если это буква или цифра.

int isalpha (int c); проверяет символ «c» на латинскую букву.

int isdigit (int c); проверяет символ «c» на десятичную цифру.

Лабораторная работа 8. Структурный тип данных на языке C++

Иногда удобно иметь набор значений различного типа, с которым можно обращаться как с одним элементом.

Рассмотрим, например, коллекцию фильмов на компакт-дисках (CD). Для удобства пользования дисками сделаем их описание, в которое включим информацию:

- о названии CD;
- фамилии режиссера;
- числе фильмов;
- стоимости CD;
- дате покупки..

Эта структура имеет пять полей. Определим тип для поля структуры.

Таблица 8.1. Типы полей структуры, описывающей CD

Имя элемента	*Тип данных*
Название CD	Символьный массив из 25 символов
Фамилия режиссера	Символьный массив из 20 символов
Число фильмов	Целый
Стоимость CD	С плавающей точкой
Дата покупки	Символьный массив из 8 символов

Определение структурного типа

Для определения структурного типа используется ключевое слово **struct**:

```
struct имя_структурного_типа
{ описание поля1;
        . . .
   описание поляn;
} [одна или более переменных];
```

Поле описывается как переменная стандартного типа или типа, определенного пользователем.

Пример 8.1. Структура с информацией о CD:

```
struct cd_info
{
    char titlel[25];  // название CD
    char regiss [20]; // режиссер
    int num_films;    // число фильмов
    float price;      // стоимость CD
    char date_bought[8]; // дата покупки
```

```
} col1,col2,col3;
```

Переменные структурного типа можно объявлять так же, как и переменные стандартных типов.

Пример 8.2. Структура с информацией о CD:

```
struct cd_info
{   char titlel[25];  // название CD
    char regiss [20]; // режиссер
    int num_films;    // число фильмов
    float price;      // стоимость CD
    char date_bought[8]; // дата покупки
};
cd_info cd1, cd2, cd3; //определение переменных cd1,
                       // cd2, cd3
```

При определении структурной переменной язык C++ резервирует для нее место в памяти. Если же был описан только структурный тип, а ни одной переменной данного типа определено не было, то место в памяти не выделяется.

Инициализация переменной структурного типа

Переменную структурного типа можно инициализировать одновременно с объявлением. Инициализация переменной структурного типа производится, если после имени переменной структурного типа следуют знак равенства (=) и список значений полей структурного типа в фигурных скобках. Значения разделяются запятой.

Пример 8.3. Инициализация структуры, содержащей информацию о коллекции фильмов на CD:

```
struct cd_info
    {       char titlel[25];     // название CD
            char regiss [20];    // режиссер
            int num_films;       // число фильмов
            float price;         // стоимость CD
            char date_bought[8]; // дата покупки
    };
cd_info cd={    "Payback",
                "Brain Helgeland",
                2,
                11.95,
                "02.13.97"
           };
```

Значения переменным-полям присваиваются в порядке их объявления при определении структурного типа.

Доступ к значениям полей структурного типа

Доступ к значениям полей структурного типа осуществляется с помощью операции точка.

Общий вид:

Имя_переменной_структурного_типа . имя_поля

Пример 8.4

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;
struct cd_info
    { char titlel[25];   // название
    char regiss [20];    // режиссер
    int num_films;       // число фильмов
    float price;         // стоимость CD
    char date_bought[9]; // дата покупки
    };
cd_info cd={" Payback ",
       " Brain Helgeland ",
                     2,
                     11.95,
                     "02.13.97"};
int _tmain (int argc, _TCHAR* argv[])
// вывод содержимого переменной cd
   setlocale (LC_ALL, "Russian");
   cout<<"Название: "<<cd.titlel<<endl;
   cout<<"Режиссер: "<<cd. regiss<<endl;
   cout<<"Число фильмов: "<<cd. num_films<<endl;
   cout<<"Стоимость: "<<cd.price<<endl;
   cout<<"Дата покупки "<<cd.date_bought<<endl;
   _getch();
         }
```

Вложенные структурные типы

Язык С++ позволяет определять один структурный тип в рамках другого. Например, структурный тип PersonInfo для записи роста, веса и даты рождения человека может быть определен, как показано в примере 8.5.

Пример 8.5

```
struct Date
{
   int month;     //месяц
    int day;      //число
   int year;      //год
};
struct PersonInfo
{
```

```
   float height;  // рост
   float weight;  // вес
   Date birthday; // дата рождения
};
```

Пример объявления переменной person типа PersonInfo:

```
PersonInfo  person;
```

Чтобы получить доступ к значению поля birthday – переменной структурного типа PersonInfo, нужно использовать операцию точка: person.birthday. Но так как birthday в свою очередь является полем структурного типа Date, то доступ, например, к году рождения осуществляется добавлением операции точка с именем year: person.birthday.year.

Массивы элементов структурного типа

Массивы элементов структурного типа определяются так же, как и массивы стандартных типов. Общий вид:

```
Имя_структурного_типа имя_переменной [N];
```

где N – количество элементов массива. Например, PersonInfo personal[20];.

Нумерация элементов массива начинается с нулевой компоненты и заканчивается индексом, равным количеству элементов массива минус 1.

Доступ к элементам массива структурного типа также осуществляется с использованием операции точка, например: personal[5].weight, personal[3].birthday.day и т. п.

Пример 8.6. Сформировать массив, содержащий сведения о вкладах. Структурный тип содержит поля: ФИО вкладчика, номер счета, тип вклада (срочный, депозит, обычный), сумма вклада, дата последнего обращения к вкладу. Написать программу, которая выводит информацию о всех вкладчиках банка в алфавитном порядке имен и о вкладчиках банка, имеющих наибольший размер вклада.

```
#include "stdafx.h"
#include <string.h>
#include <conio.h>
#include <iostream>
#include <iomanip>
using namespace std;
struct Banc
    {
      char name[20];  // фамилия вкладчика
      int sch;        // номер счета
      char tip[10];   // тип вклада
      float sum;      // сумма вклада
      char data[10];  // дата последнего обращения
    };
int _tmain (int argc, _TCHAR* argv[])
{
  setlocale (LC_ALL, "Russian");
```

```
  Banc b[10],b_new[10];
  int kol;
  cout<<"Введите количество вкладчиков >";
  cin>>kol;
  for (int i=0; i<kol; i++)
  {
    cout<<"Имя вкладчика :";
    cin>>setw (15)>>b[i].name;
    cout<<"Номер счета :";
    cin>>b[i].sch;
    cout<<"Тип вклада :";
    cin>>b[i].tip;
    cout<<"Сумма вклада :";
    cin>>setw (10)>>b[i].sum;
    cout<<"Дата последнего обращения к вкладу :";
    cin>>setw (10)>>b[i].data;
    cout<<endl;
  }

/* Вывод содержимого массива вкладчиков
   в алфавитном порядке*/
  for(int i=0; i<kol; i++)
   {
     Banc min=b[i];
     int num=i;
     for(int j=i; j<kol; j++)
        if(strcmp(b[j].name,min.name)<0)
         {
           min=b[j];
           num=j;
         }
     b[num]=b[i];
     b[i]=min;
   }

  cout<<"Имя вкладчика"<<"Номер счета"
      <<"Тип вклада";
  cout<<"Сумма вклада"
      <<"Дата последнего обращения к вкладу\n";
  for(int i=0; i<kol; i++)
    {
cout<<setw(15)<<b[i].name<<setw(13)<<b[i].sch
    <<setw(11)<<b[i].tip;
cout<<setw(13)<<b[i].sum
    <<setw(16)<<b[i].data<<endl;
    }
// Формирование массива вкладчиков,
// имеющих наибольший размер вклада
  float max=0.0;
  int kol_new=0;
// Определение значения наибольшего вклада
  for(int i=0; i<kol; i++)
    if(b[i].sum>max)
         max=b[i].sum;
```

```
  cout<<"Наибольшая сумма вклада"<<max<<endl;
// Массив вкладчиков с наибольшей суммой вклада
  for(int i=0; i<kol; i++)
    if(b[i].sum==max)
       b_new[kol_new++]=b[i];
  cout<<"Имя вкладчика"<<"Номер счета"
      <<"Тип вклада"<<"Сумма вклада"
      <<"Дата последнего обращения к вкладу\n";
  for( int i=0; i<kol_new; i++)
    {
      cout<<setw(15)<<b_new[i].name
          <<setw(13)<<b_new[i].sch
          <<setw(11)<<b_new[i].tip;
         <<setw(13)<<b_new[i].sum
         <<setw(16)<<b_new[i].data<<endl;
    }
_getch();
return 0;
}
```

Контрольные вопросы

1. Какова область применения структур?
2. Каким образом определяется структура?
3. Как определяются переменные типа структура?
4. Как осуществляется доступ к структурным членам?

Задания

Таблица 8.2. Простые варианты

Варианта	*Задание*
1	Сформировать массив, содержащий сведения о количестве изделий, собранных рабочими цеха за неделю. Структурный тип содержит поля: фамилия сборщика; количество изделий, собираемых им ежедневно в течение шестидневной недели, т. е. раздельно в понедельник, вторник и т. д. Написать программу, выдающую на печать фамилию рабочего и общее количество деталей, собранных им за неделю
2	Сформировать массив, содержащий сведения о количестве изделий категорий *A*, *B*, *C*, собранных рабочим за месяц. Структурный тип содержит поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Считая заданными значения расценок *SA*, *SB*, *SC* за выполненную работу по сборке единицы изделия категорий *A*, *B*, *C*, вывести на экран общее количество изделий категорий *A*, *B*, *C*, собранных рабочим цеха

Варианта	*Задание*
3	Сформировать массив, содержащий сведения о телефонах абонентов. Структурный тип содержит поля: фамилия абонента, место жительства (название улицы, номер дома), год установки телефона. Написать программу, выдающую информацию о количестве установленных телефонов с *XXXX* (ввести с клавиатуры) года
4	Сформировать массив, содержащий сведения об ассортименте игрушек в магазине. Структурный тип содержит поля: название игрушки, цена, количество, возрастные границы (например, от 3 до 5 лет). Написать программу, выдающую стоимость самой дорогой игрушки и ее название
5	Сформировать массив, содержащий сведения о сдаче студентами сессии. Структурный тип содержит поля: индекс группы, фамилия студента, оценки по пяти экзаменам. Написать программу, выдающую фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей
6	Сформировать массив, содержащий сведения о личной коллекции книголюба. Структурный тип содержит поля: шифр книги, автор, название, год издания, местоположение (номер стеллажа). Написать программу, выдающую список книг и их местоположение автора Z, находящихся в коллекции
7	Сформировать массив, содержащий ежедневные метеосводки. Структурный тип содержит поля: число месяца, температура ночная, температура дневная, осадки (дождь, снег, без осадков), ветер, видимость на дорогах. Написать программу, которая выдает информацию о самом холодном дне с видимостью на дорогах менее 500 м
8	Сформировать массив, содержащий сведения об ассортименте обуви в магазине фирмы. Структурный тип содержит поля: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской. Написать программу, выдающую о наличии и стоимости обуви артикула *X* (ввести с клавиатуры)
9	Сформировать массив, содержащий сведения о нападающих команды «Спартак». Структурный тип содержит поля: имена нападающих, число заброшенных ими шайб, число сделанных голевых передач, заработанное штрафное время. Написать программу, которая определяет по сумме очков (голы + передачи) четырех лучших игроков
10	Сформировать массив, содержащий сведения о реках, пригодных для путешествий. Структурный тип содержит поля: название реки, категория сложности, протяженность маршрута, начало нави-

Варианта	*Задание*
	гации (месяц), конец навигации (месяц). Написать программу, которая выводит информацию о реках указанной протяженности
11	Сформировать массив, содержащий сведения об отправлении поездов дальнего следования с Казанского вокзала. Структурный тип содержит поля: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Написать программу, выдающую информацию о наличии билетов на поезд с номером *XXX* (ввести с клавиатуры)
12	Сформировать массив, содержащий сведения о сотрудниках института. Структурный тип содержит поля: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Написать программу, которая позволяет получить информацию о среднем стаже работающих в отделе *X* (ввести с клавиатуры)
13	Сформировать массив, содержащий сведения о пациентах глазной клиники. Структурный тип содержит поля: фамилия пациента, пол, возраст, место проживания (город), диагноз. Написать программу, которая выдает информацию о количестве иногородних, прибывших в клинику
14	Сформировать массив, содержащий сведения о наличии билетов на рейсы Аэрофлота. Структурный тип содержит поля: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Написать программу, выдающую время вылета самолетов в город *X* (ввести в клавиатуры)

Таблица 8.3. Варианты повышенной сложности

Вариант	*Задание*
1	Сформировать массив, содержащий ежедневные метеосводки. Структурный тип содержит поля: число месяца, температура ночная, температура дневная, скорость ветра, видимость на дорогах. Написать программу, которая выдаст информацию: · о самом холодном дне с видимостью на дорогах не менее 500 м; · дне с минимальной среднесуточной температурой и о дне с максимальной среднесуточной температурой; · о всех днях, когда было теплее, чем введено с клавиатуры со скоростью ветра не более, чем введено с клавиатуры. Вывести всю информацию об этих днях
2	Сформировать массив, содержащий сведения о цветочном магазине. Структурный тип содержит поля: фамилия поставщика, название цветов, количество проданных цветов, цена поставщика, цена магазина. Вывести на экран информацию:

Вариант	*Задание*
	· об общем количестве цветов, полученных магазином от данного поставщика; · выручке магазина от продажи цветов; · выручке данного поставщика
3	Сформировать массив, содержащий сведения о сотрудниках института. Структурный тип содержит поля: фамилия преподавателя, название кафедры, год рождения, стаж работы, ученая степень, ученое звание, оклад. Написать программу, которая позволит: · составить список профессоров с указанием стажа работы; · вычислить средний оклад сотрудников института; · составить список профессоров и доцентов, работающих на кафедре, название которой введено с клавиатуры
4	Сформировать массив, содержащий информацию о туристских поездках. Структурный тип содержит поля: название тура, необходимость визы, цена, длительность пребывания, категория отеля, есть ли море, предусмотрены ли экскурсии. Написать программу, которая выводит информацию о турах в соответствии с требованиями: · безвизовый тур с возможностью купаться в море и посещать экскурсии; · тур по Европе, не дороже введенной с клавиатуры стоимости, с категорией отеля, не ниже введенной с клавиатуры; · наличие моря, ограничение по цене и по времени пребывания, которые введены с клавиатуры
5	Сформировать массив, содержащий сведения о наличии автобусных маршрутов. Структурный тип содержит поля: номер автобуса, пункт отправления, пункт назначения, интервал движения, время в пути. Написать программу, выдающую следующую информацию: · возможность добраться от пункта А до пункта Б без пересадок; · будет ли автобус от пункта А до пункта Б в течение времени, введенного с клавиатуры; · возможность добраться от пункта А до пункта Б за указанное время (с учетом пересадок, не более одной)
6	Сформировать массив, содержащий сведения о детских садах города. Структурный тип содержит поля: номер, государственный или частный, профиль (обычный, спортивный, художественный, с изучением иностранных языков и т. п.), наличие свободных мест, количество человек в группе, возможность ночного пребывания. Написать программу, способную выдать информацию:

Вариант	*Задание*
	• о наличии мест для двух детей в государственном детском саду спортивного профиля с возможностью ночного пребывания; • количестве частных детских садов в городе с дополнительными возможностями и наличие свободных мест в них; • возможности размещения ребенка в детском саду с количеством детей в группе не большим введенного с клавиатуры, имеющим профиль, введенный с клавиатуры
7	Сформировать массив, содержащий сведения о булочной. Структурный тип содержит поля: номер, название товара, розничная цена товара, фирма-поставщик, количество полученных единиц данного товара, количество проданных единиц данного товара, цена поставщика на данный товар. Выдать на экран информацию: • о выручке магазина от продажи хлебо-булочных изделий; • наличии в данный момент булочки, название которой введено с клавиатуры; • выручке фирмы-поставщика, название которой введено с клавиатуры, от продажи его хлебо-булочных изделий
8	Сформировать массив, содержащий информацию об озерах. Структурный тип содержит поля: название, соленое или пресное, глубина, площадь, судоходное или нет, есть ли места для отдыха, возможна ли рыбалка, замерзает ли зимой. Написать программу, выдающую информацию об озерах: • где возможна зимняя рыбалка; • на которых до места отдыха можно добраться на теплоходе; • по введенному с клавиатуры названию
9	Сформировать массив, содержащий базу данных риэлтера, продающего дачные участки. Структурный тип содержит поля: район, поселок, расстояние от Москвы, шоссе, площадь участка, есть ли дом, наличие электричества, наличие воды, цена. Написать программу, которая покажет: • самый большой участок Истринского района; • участок по Волоколамскому шоссе, не дальше от Москвы, чем введено с клавиатуры, где есть дом и электричество; • все участки района (название введено с клавиатуры), площадью не меньше, чем введено с клавиатуры и не дороже указанной цены

Вариант	*Задание*
10	Сформировать массив, содержащий данные о горных маршрутах и горнолыжных курортах. Структурный тип содержит поля: название горы, высота, наличие подъемника, категория сложности, наличие горнолыжных трасс, наличие альпинистских маршрутов. Написать программу, выдающую сведения: · о самой высокой горе из тех, по которым проходят альпинистские маршруты; · горах, имеющих горнолыжные трассы с подъемниками, категории сложности не выше, чем введено с клавиатуры; · горах, высотой не менее, чем введено с клавиатуры, категории сложности не менее указанной, где можно проводить соревнования по горнолыжному спорту
11	Сформировать массив, содержащий данные о самолетах, участвующих в авиасалоне. Структурный тип содержит поля: место проведения авиасалона, тип самолета, бортовой номер, пилот, воинское звание пилота, опыт пилота (общее количество летных часов), название пилотажной группы. Написать программу, выдающую: · фамилии пилотов пилотажной группы «Стрижи»; · названия пилотажных групп, участвующих в авиасалоне, название которого введено с клавиатуры; · фамилии всех участников авиасалона (название введено с клавиатуры), имеющих воинское звание (введенно с клавиатуры), с количеством летных часов не менее чем введено с клавиатуры
12	Сформировать массив, содержащий сведения о фильмах на DVD-дисках из частной коллекции. Структурный тип содержит поля: название фильма, фамилия режиссера, год выпуска, время трансляции, цветной или черно-белый. Написать программу, выдающую: · названия всех короткометражных фильмов (короче 30 мин); · названия всех черно-белых фильмов, выпущенных с года, введенного с клавиатуры; · названия всех фильмов режиссера, фамилия которого введена с клавиатуры, выпущенные не раньше введенного с клавиатуры года

Лабораторная работа 9. Файловый ввод/вывод. Текстовые файлы. Организация ввода и вывода. Файловая система

Операции ввода/вывода в языке С осуществляются через потоки. *Поток* – это логическое устройство, выдающее и принимающее информацию.

С потоком связано понятие внутреннего указателя, определяющего позицию, с которой начинается следующая операция чтения или записи. При каждой операции чтения или записи происходит автоматическое перемещение указателя.

В языке С (С++) формат стандартных файлов ввода/вывода описан в заголовочном файле stdio.h. Имена стандартных файлов ввода/вывода для языка С (С++) представлены в табл. 9.1. В момент начала выполнения программы на языке С (С++) автоматически открываются три потока: stdin, stdout, stderr.

Таблица 9.1. Потоки, определяемые в языке С и С++

Имя стандартного файла	*Описание*
stdaux	Последовательный ввод/вывод
stderr	Выходной поток ошибок
stdin	Стандартный ввод
stdout	Стандартный вывод
stdprn	Вывод на принтер

С++ поддерживает всю систему ввода/вывода С и добавляет к ней дополнительные возможности, связанные в основном с вводом/выводом объектов. Описание средств для создания потоков в языке С++ представлено в заголовочном файле iostream.h. Когда начинает работать программа на С++, открываются потоки, приведенные в табл. 9.2.

Таблица 9.2. Потоки, определяемые в языке С++

Имя стандартного файла	*Описание*
cin	Стандартный ввод – клавиатура
cout	Стандартный вывод – экран
cerr	Стандартная ошибка – экран
clog	Буферизованная версия cerr – экран

Файловая система языков С и С++ состоит как бы из двух уровней: логических файлов и физических файлов, с которыми логические файлы всегда связаны.

Логический файл описывается как указатель на открываемый поток FILE * и служит средством взаимодействия с физическим файлом. Имя физического файла появляется в программе всего один раз, в тот момент, когда происходит открытие файла, осуществляемое функцией fopen() и одновременно его связывание с логическим файлом.

Основными действиями, производимыми над файлами, являются их открытие, обработка и закрытие. Обработка файлов может заключаться в считывании блока данных из потока в оперативную память, запись блока данных из оперативной памяти в поток, считывание определенной записи данных из потока, занесение определенной записи данных в поток. При этом необходимо помнить, что понятие файла в памяти ЭВМ не определено, и приобретает смысл только после его связи с внешним физическим файлом.

Текстовые файлы

Тип FILE определяется в заголовочном файле stdio.h и обычно представляет собой структуру, содержащую параметры реализации потока, такие как адреса буферов, указатели позиций потока, маркеры ошибок потока и т. п.

При работе с дисковыми файлами в момент их открытия следует задать режим доступа, чтобы определить, к какому файлу осуществляется доступ: к текстовому или двоичному, а также способ доступа – чтение или запись. Все это выполняется функцией fopen(), имеющей синтаксис:

```
fopen ("имя_файла", "режим_доступа")
```

Режимы доступа к файлам для функции fopen() приведены в табл. 9.3.

Таблица 9.3. Режимы доступа к файлам

Режим	*Описание*
r	Открыть файл только для чтения, модификация файла запрещена
w	Создать новый файл только для записи. При попытке открыть таким способом существующий файл происходит перезапись файла. Чтение данных из файла запрещено
a	Открыть файл для дозаписи. Если файла с указанным именем не существует, он будет создан
r+	Открыть существующий файл для чтения и записи
w+	Создать новый файл для чтения и записи
a+	Открыть существующий файл для дозаписи и чтения

Чтобы открыть текстовый файл, например, для чтения, нужно произвести следующие действия:

```
FILE *ft;          // объявили указатель на файловый поток
ft = fopen ("inp_f.txt","r");   // открыли файл inp_f.txt
```

При попытке открыть существующий файл можно допустить ошибку в его имени или в указании пути к нужному файлу, что вызывит ошибку исполнения программы. Следует предвидеть подобные ситуации и проводить проверку возможности открытия файла. Такую проверку осуществить довольно легко, так как функция fopen() возвращает значение указателя в случае успешного его открытия и значение NULL, если доступ к файлу невозможен. Поэтому достаточно написать:

```
if (ft = fopen ("inp_f.txt","r") != NULL)
    {  // обработка файла
    }
```

Текстовой файл состоит из последовательности символов, разбитой на строки путем использования управляющего символа \n. На диске текстовые файлы хранятся в виде сплошной последовательности символов, и их деление на строки становится заметным лишь в момент вывода на экран или печать, так как именно при выводе управляющие символы начинают выполнять свои функции. Текстовые файлы легко переносятся с одного типа компьютера на другой лишь в случаях, когда они содержат только символы, принадлежащие стандартному набору символов.

При работе с текстовыми файлами возможна их посимвольная или построчная обработка.

Основные методы обработки текстовых файлов

Файловые функции ввода/вывода fprintf() и fscanf() работают аналогично функциям printf() и scanf(), но имеют дополнительный аргумент, являющийся указателем на файловый поток.

Пример 9.1. Написать программу, которая записывает в выходной файл каждый пятый символ из входного файла.

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{  // объявим две файловые переменные:
   FILE *f_in, *f_out;
   char ch;
   int count = 0;       // счетчик элементов
   // открыть файл для чтения:
   f_in=fopen("d:\\input.txt", "r");
   if ((f_in) != NULL) // если файл открылся
```

```
     {  // открыть файл для записи:
        fopen_s(&f_out, "d:\\outp.txt", "w");
        while((ch = fgetc(f_in)) != EOF)
            if (count++ % 5 == 0) // выводит каждый
              fputc(ch, f_out);   // пятый символ
        fclose(f_in);  // закрыть входной файл
        fclose(f_out); // закрыть выходной файл
     }
    else printf("File Not Opened\n ");
   _getch();
   return 0;
}
```

При работе с текстовыми файлами возможна не только поэлементная обработка файлов, но и построчная.

Пример 9.2. Построчное чтение информации из входного файла и вывод ее на экран.

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  FILE *f_in;
   char buffer[256]; // максимальная длина строки -
                     // 255 символов
   char name[80];    // имя входного файла
   cout<<"Input File Name";
   cin>>name;
   if ( (f_in = fopen (name, "r")) != NULL)
         { while (fgets (buffer,255,f_in) != NULL)
//Пока файл не закончится = while not EOF
              { fputs (buffer, stdout);
                fputc ('\n', stdout);
              }
           fclose (f_in);
         }
      else cout<<"File Not Opened\n ";
  _getch();
  return 0;
}
```

В цикле while присутствуют две файловые функции работы со строками: fgets() для чтения строки символов в буфер и fputs() – для записи содержимого буфера в файл.

Закрыть файл не менее важно, чем открыть его, так как в этот момент происходит заполнение ячейки таблицы размещения файлов значением, которое является признаком завершения файла и установка атрибутов файла.

Кроме того, вывод текстового файла буферизован. Это значит, что в тот момент, когда работает оператор записи в файл, фактическая запись может и не происходить, поскольку реально сначала происходит заполнение данными текстового буфера, а потом его содержимое записывается на диск. Запись буфера происходит как только он окажется полностью заполненным или при выполнении специальных команд принудительной записи на диск. Процесс записи недозаполненного буфера на диск называется *флэшированием* и обычно выполняется с помощью функции fflush (f_out). При необходимости завершить работу сразу со всеми открытыми файлами пользуются функцией flushall().

Закрытие файла посредством функции fclose (f_out) также включает процесс флэширования, т. е. перенос информации из буфера на диск.

Доступ к элементам текстовых файлов возможен только в последовательном режиме как при записи файла, так и при его чтении.

Контрольные вопросы

1. Что включает в себя понятие файла? Как оно связано со стандартными потоками ввода/вывода?
2. Как связаны между собой понятия логического и физического файлов?
3. Что такое режим доступа? Перечислить возможные режимы доступа при работе с текстовыми файлами.

Задания

Таблица 9.4. Простые варианты

Вариант	*Задание*
1	Создать текстовый файл, содержащий произвольные целочисленные значения. Определить, являются ли значения, находящиеся в файле, упорядоченными по возрастанию. Результат записать в другой файл
2	Создать текстовый файл, содержащий произвольные целочисленные значения. Определить сумму четных элементов файла и количество нечетных. Вычисленные значения записать в другой файл
3	Создать текстовый файл, содержащий произвольные символьные значения. Подсчитать количество буквенных, цифровых символов и символов, не являющихся ни буквами, ни цифрами. Результат записать в другой текстовый файл

Вариант	*Задание*
4	Создать текстовый файл, содержащий произвольные целочисленные значения. Определить максимальное значение среди элементов файла, имеющих нечетные порядковые номера и записать его в другой файл, сопроводив комментариями
5	Создать текстовый файл, содержащий произвольные символьные значения. Определить максимальное среди буквенных символов. Результат записать в другой текстовый файл
6	Создать текстовый файл, содержащий произвольные целочисленные значения. Определить, сколько из значений, находящихся в файле, кратны трем
7	Создать текстовый файл, содержащий произвольные вещественные значения. Считать из файла записанные данные и определить максимальное значение. Если оно находится в первой половине файла, записать в другой файл сумму следующих за ним элементов
8	Создать текстовый файл, содержащий произвольные целочисленные значения. Считать из файла записанные данные и определить максимальное значение. Если оно является степенью тройки, записать в другой файл каждое третье значение данного файла
9	Создать текстовый файл, содержащий произвольные целочисленные значения. Считать из файла записанные данные и определить сумму нечетных значений. Если оно больше введенного с клавиатуры числа, записать в другой файл четные элементы данного файла
10	Создать текстовый файл, содержащий произвольные символьные значения. Определить максимальное среди цифровых символов. Результат записать в другой текстовый файл
11	Создать текстовый файл, содержащий произвольные целочисленные значения. Считать из файла записанные данные и определить максимальное и минимальное значения и их порядковые номера. Записать в другой файл полученные значения
12	Создать текстовый файл, содержащий произвольные символьные значения. Определить количество нецифровых символов. Результат записать в другой текстовый файл
13	Создать текстовый файл, содержащий произвольные вещественные значения. Считать из файла записанные данные и определить максимальное значение. Если оно находится во второй половине файла, записать в другой файл сумму предыдущих элементов
14	Создать текстовый файл, содержащий произвольные целочисленные значения. Считать из файла записанные данные и записать в другой файл четные элементы исходного файла

Таблица 9.5. Варианты для работы с табличными данными

Вариант	*Задание*
1	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 5 1 2 8 12 3 – – – – Считать из файла пары значений и в тех из них, где X>Y , поменять значения X и Y местами. Результат записать в другой текстовый файл такого же формата
2	Ввести с клавиатуры попарно значения вещественного типа и записать их в текстовый файл в виде таблицы формата: X : Y 2.1 : 3.7 6.2 : 5.4 --- – --- Считать из файла полученные пары значений и создать из них другой файл вида: sin (x) : cos (y) значение sin (2.1) : значение cos (3.7) ---------- – ----------
3	Случайным образом создать таблицу пар значений и записать её в текстовый файл в виде: n * c 5 * m 7 * a 3 * q ---------- Считать из файла пары значений и создать из них другой текстовый файл вида mmmmm aaaaaaa qqq
4	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 1 8 3 –

Вариант	*Задание*
	– Считать из файла пары значений и в тех из них, где X кратен Y , пометить в том же файле строку таблицы: X Y 5 1 *** 8 12 3 *** –
5	Случайным образом создать таблицу пар значений и записать её в текстовый файл в виде: a b c 5.2 4.6 2.5 можно 1.2 8.9 2.3 ---------- Считать из файла записанные данные и определить, можно ли построить треугольник с такими сторонами. Пометить соответствующие строки таблицы (в том же файле)
6	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 5 1 2 8 12 3 – – – – Считать из файла пары значений и в тех из них, где X+Y является четным числом, записать в другой текстовый файл такого же формата
7	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 5 1 2 8 12 3 – – – – Считать из файла пары значений и в тех из них, где X является степенью 2, а Y – нечетное, пометить в том же файле строку таблицы: X Y . 5 8 16 3 ###

Вари-ант	*Задание*
	32 6 – –
8	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 25 3 7 – Считать из файла пары значений и в тех из них, где X является точным квадратом Y или наоборот, найти сумму значений X и Y. Результат записать в другой текстовый файл в виде X Y sum 5 25 30 3 49 7 56
9	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 5 1 2 8 12 3 – – – – Считать из файла пары значений и в тех из них, где Y кратен X, а X – четное, пометить в том же файле строку таблицы: X Y . 5 10 2 8 *** 12 3 – –
10	Случайным образом создать таблицу пар значений и записать её в текстовый файл в виде: a b c 5.2 4.6 2.5 можно 1.2 8.9 2.3 ---------- Считать из файла записанные данные и определить, можно ли построить треугольник с такими сторонами. В соответствующих строках (где можно), указать площадь полученного треугольника (в другом файле)

Вариант	*Задание*
11	Создать текстовый файл, содержащий целые значения, следующего формата 5 3 21 4 37 52 9 2 . . . Считать из файла записанные данные и определить минимальное значение. Если оно кратно трем, заменить каждое третье значение файла нулем, если кратно пяти – заменить его суммой первого и последнего элементов
12	Случайным образом создать таблицу пар значений и записать её в текстовый файл в виде: n * c 5 * m 7 * a 3 * q ---------- Преобразовать эту таблицу по следующему образцу (преобразования производить в исходном файле) n * c # 5 * m mmmmm 7 * a aaaaaaa 3 * q qqq ---------- Если первое значение не является числом, то в третьем столбце стоит один символ #
13	Ввести с клавиатуры значения вещественного типа и записать их в текстовый файл в виде таблицы следующего формата: X : Y : Z 2.1 : 3.7 : 0.9 6.2 : 5.4 : 4.2 --- – --- : --- Считать из файла полученные значения и создать из них другой файл вида: sin (max{X,Y,Z}) : cos (min{X,Y,Z}) значение sin (3.7) : значение cos (0.9) ---------- – ----------
14	Случайным образом создать таблицу пар целочисленных значений и записать её в текстовый файл в виде: X Y 5 1 2 8 12 3 – – – –

Вари-ант	*Задание*
	Считать из файла пары значений и те из них, где X*Y является четным числом, записать в другой файл в виде: X Y RES 2 8 16 12 3 36 – –

Лабораторная работа 10. Файловый ввод/вывод. Двоичные файлы

Двоичные файлы

Текстовые файлы не являются единственно возможным способом хранения информации на диске. Можно запоминать информацию и в двоичном виде.

Принципиально обработка двоичных файлов не сильно отличается от обработки текстовых файлов. В любом случае, прежде чем работать с файлом, следует связать его физическое имя с логическим именем (с файловым потоком) и открыть файл, указав режим доступа. После работы с файлом его необходимо закрыть. Тем не менее, отличия обработки двоичных файлов и текстовых все же существуют. Рассмотрим их.

Открыть файл для двоичной обработки можно посредством вызова функции fopen(), но ко всем режимам доступа добавляют строчную латинскую букву b. Режимы доступа одинаковы для текстовых и для двоичных файлов.

```
FILE *fb = fopen ( "Bin_fil.dat","wb");
```

откроет файл для записи в двоичном режиме. А чтобы открыть его для чтения и записи нужно написать:

```
FILE *fb = fopen ( "Bin_fil.dat","r+b");
```

Если после вызова функции fopen() указатель на файловый поток fb не равен 0, его можно использовать в последующих обращениях к функциям работы с двоичными файлами, таким как fread() и fwrite().

Закрывают двоичные файлы, как и текстовые, функцией fclose().

Не следует использовать функции обработки текстовых файлов применительно к двоичным файлам и наоборот.

Существует два способа доступа к элементам двоичных файлов: последовательный и произвольный.

Последовательный доступ к элементам двоичных файлов

Последовательный доступ к элементам файла особенно эффективен, если нужно перебрать все данные, хранящиеся в нем. Кроме того, если файл открыт для записи, но еще не содержит данных (пустой), то заполнение его возможно лишь в последовательном режиме.

Пример 10.1. Записать 100 целых, случайно выбранных чисел, в двоичный файл.

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  setlocale (LC_ALL, "Russian");
   FILE *f_out;
   int number;
   fopen_s (&f_out, "Int.dat","wb");
   if (!f_out) { puts ("Нельзя создать файл !\n");
                 exit (1);
               }
   for (int i = 0; i < 100; i++)   // цикл формирования
                                   // данных и записи их в файл
         {number = rand()%100 - 50;
          fwrite (&number, sizeof (int), 1, f_out);
         }
   fclose (f_out);

   fopen_s (&f_out, "Int.dat","rb");
   if (!f_out) {  puts ("Нельзя открыть файл !\n");
                  exit (1);
               }
   for (int i = 0; i < 100; i++)// цикл чтения данных из
                                // файла и вывода их на экран
    { fread (&number, sizeof (int), 1, f_out);
      cout<<number<<' ';
    }
   fclose (f_out);
   _getch();
   return 0;
}
```

Все использованные в программе функции встречались ранее, кроме функции fwrite(), производящей запись данных в поток. Прототип ее находится в заголовочном файле stdio.h. Синтаксическое описание функции имеет вид:

```
size_t fwrite (const void *ptr, size_t site,
                                size_t n, FILE *stream);
```

Параметры функции:

const void *ptr – указатель на исходные данные, записываемые в файл;

size_t size – размер в байтах одного элемента данных;

size_t n – число записываемых в файл элементов данных, размером size байт каждый;

FILE *stream – указатель на файловый поток, открытый в двоичном режиме.

Поскольку функция fwrite() имеет в качестве одного из своих аргументов количество записываемых элементов, можно применять ее и для записи в файл сразу целого массива, который к моменту записи в файл должен быть сформирован и заполнен данными.

Функция чтения значений с диска fread() имеет тот же синтаксис, что и функция записи fwrite(), только первым аргументом здесь является адрес приемника, в который эта функция должна скопировать байты с диска. При использовании этой функции нужно убедиться в том, что размер приемника достаточен для копирования количества байтов. Прототип функции fread() находится в заголовочном файле stdio.h.

Функция fread() может загрузить больше одного значения с диска в один прием. Тогда в качестве приемника должен выступать массив элементов соответствующего типа и достаточного размера. Этот массив может быть статическим, если заранее известна длина файла, или динамическим, если размер файла определяется в процессе выполнения программы.

```
float arr_f[100];
   . . .
fread (&arr_f, sizeof (float), 100, f_inp);
```

Это самый быстрый способ загрузки из файла большого количества чисел.

Организация произвольного доступа к элементам двоичных файлов

Организация произвольного доступа к компонентам файла позволяет считывать значения из любой позиции в файле, а также записывать новую информацию в любое место в файле. Но к файлам с произвольным доступом предъявляется одно жесткое требование: их компоненты должны иметь одинаковую длину. Двоичные файлы позволяют обеспечить удовлетворение этого требования. О том, чтобы данные, которые будут находиться в файле произвольного доступа, имели одинаковый размер, следует позаботиться в момент создания файла.

Еще раз напомним о том, что первичная запись в файл возможна только в режиме последовательного доступа.

Для организации произвольного доступа к элементам файла можно осуществить с помощью функции fseek(), прототип которой описан в заголовочном файле stdio.h. Синтаксическое описание функции:

```
int fseek (FILE * stream, long offset, int whence);
```

Функция fseek() перемещает внутренний указатель файлового потока, изменяя место в файле, с которого начинается следующая операция чтения или записи. В случае успешного завершения функция возвращает 0, в случае ошибки – ненулевое значение.

Параметры функции:

FILE *stream – указатель на открытый файловый поток, аналогичной возвращаемому функцией fopen().

long offset – число байтов, на которое нужно переместить файловый указатель в направлении, указанном параметром whence. Для перемещения файлового указателя в обратном направлении (в сторону начала файла), следует устанавливать offset равным отрицательному значению.

int whence – указывает положение точки отсчета файлового указателя, от которой будет происходить его перемещение. Значения аргумента whence представлены в табл. 10.1.

Таблица 10.1. Значения аргумента whence

Значение	*Описание*
SEEK_SET	Перемещение файлового указателя происходит относительно начала файла
SEEK_END	Перемещение файлового указателя происходит относительно конца файла
SEEK_CUR	Перемещение файлового указателя происходит относительно текущей позиции файлового указателя

При использовании функции fseek() следует соблюдать осторожность, так как попадание за пределы файла чаще всего не приводит к генерации ошибки, поэтому программисту самому следует принимать меры для предотвращения обращения к диску за пределами известных границ файла.

При организации произвольного доступа используется функция ftell(), осуществляющая навигацию внутри файла. Прототип функции описан в stdio.h. Данная функция возвращает внутренний указатель файлового потока, равный смещению в байтах от начала двоичного файла до байта, с которого начинается следующая операция ввода/вывода. Это значение можно передать функции fseek(), или использовать каким-либо другим образом.

Синтаксическое описание функции:

```
long int ftell (FILE *stream).
```

Единственным параметром функции является указатель на открытый файловый поток.

Следующие операторы демонстрируют возможности функции fseek():

fseek (f, sizeof (t), SEEK_CUR); – перемещает файловый указатель с текущей позиции на следующую;

fseek (f, -sizeof (t), SEEK_CUR); – на предыдущую позицию;

fseek (f, 0,SEEK_END); – на конец файла.

При работе с двоичными файлами не следует забывать добавлять букву «b» при указании режима доступа функции fopen(), но нужно быть особенно внимательным, если предполагается, что будет осуществляться произвольный доступ к компонентам данного файла.

Пример 10.2. Записать ноль на место максимального значения в файле Test.dat.

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <io.h>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{  FILE *f_inp;
   int i_max;  // номер максимального элемента
   int handle; // дескриптор файла
   int value, max, zero = 0;
   int number[20];

   fopen_s (&f_inp, "I:\\Int.dat","wb");
   if (!f_inp)
     {  puts("Нельзя создать файл !\n");
        exit(1);
     }
   for (int i = 0; i < 20; i++)
     {number[i] = rand()%100 - 50;
      cout<<number[i]<<' ';
     }
   fwrite(number, sizeof(int), 20, f_inp);
   fclose(f_inp);
   cout<<endl;
   cout<<endl;

   fopen_s(&f_inp, "I:\\Int.dat","r+b");
   if (! f_inp)
     {  puts( "File not opened !\n" );
        exit(1);
     }
// преобразовать открытый файловый поток
// в дескриптор файла
   handle = _fileno(f_inp);
// считать 0 элемент файла:
   fread(&max, sizeof(int), 1, f_inp);
   i_max = 0;
```

```
   for (int i = 1; i<_filelength(handle); i++)
    { fread(&value, sizeof(int), 1, f_inp);
       if (value>max) {max = value; i_max = i;}
    }

     cout<< max<< "_____" << i_max<<endl;
     fseek(f_inp, i_max*sizeof(int), SEEK_SET);
     fwrite(&zero, sizeof(int), 1, f_inp);
     fclose(f_inp);

     fopen_s(&f_inp, "I:\\Int.dat","r+b");
     if (! f_inp)
       {puts( "File not opened !\n" );
        exit(1);
       }
     fread(number, sizeof(int), 20, f_inp);
     for (int i = 0; i < 20; i++)
             cout<<number[i]<<' ';
    cout<<endl;

     fclose(f_inp);
     _getch();
    return 0;
}
```

Функция filelength (f_inp) в качестве аргумента принимает дескриптор файла, открытого функцией fopen(), и возвращает размер этого файла. Для получения *дескриптора*, используемого для идентификации файла, служит функция преобразования файлового потока в дескриптор fileno(). Аргументом функции fileno (FILE *stream) является файловый поток. Отрицательное значение дескриптора служит признаком ошибки.

Заметим, что, как и в массивах, первый элемент, находящийся в файле, имеет номер 0, второй – 1 и т. д.

Для определения конца файлового потока используется функция feof(), прототип которой описан в заголовочном файле stdio.h. Функция возвращает истину (1), если внутренний указатель заданного файлового потока находится за последним байтом файла и ложь (0), если внутренний указатель файла находится не в конце файла.

Контрольные вопросы

1. Какие функции осуществляют чтение и запись при работе с двоичными файлами?
2. В чем состоят особенности чтения массивов из двоичных файлов и их записи двоичные файлы?
3. Что такое последовательный и произвольный доступ к компонентам файла?
4. Как «подойти» к предпоследнему элементу файла? Привести все возможные варианты.

Задания

Таблица 10.2. Простые варианты

Вариант	*Задание*
1	Создать двоичный файл из случайно заданных значений целого типа. Записать в другой файл четные значения исходного файла. Порядок следования чисел сохранить
2	Создать двоичный файл из случайно заданных значений целого типа. Записать в другой файл ноль на место каждого четного значения исходного файла, единицу – на место каждого нечетного значения исходного файла
3	Создать двоичный файл из случайно заданных значений целого типа. Записать в другой файл первые пять и последние пять элементов исходного файла. Порядок следования чисел сохранить
4	Из двоичного файла, элементами которого являются целые числа, переписать вторую половину в другой файл. Порядок следования чисел сохранить
5	Создать двоичный файл из случайно заданных значений символьного типа. Записать в другой файл буквенные и числовые значения исходного файла. Порядок следования элементов файла сохранить
6	Создать двоичный файл из случайно заданных значений целого типа. Записать в другой файл элементы исходного файла, которые больше числа, введенного с клавиатуры. Порядок следования чисел сохранить
7	В двоичном файле типа double найти максимальный элемент. Записать его в другой файл
8	Из файла, содержащего как положительные, так и отрицательные вещественные числа, сформировать два других, в один из которых поместить положительные значения исходного файла, в другой – отрицательные
9	Создать двоичный файл из случайно заданных значений символьного типа. Записать в другой файл заглавные буквы исходного файла
10	В двоичном файле с элементами типа float вычислить среднее арифметическое значение элементов этого файла. Полученное значение записать в другой файл
11	Создать двоичный файл из случайно заданных значений символьного типа. Записать в другой файл символы исходного файла, обозначающие арифметические операции
12	Создать двоичный файл с элементами символьного типа. Записать в другой файл элементы исходного до первого символа '!'.

Вариант	*Задание*
13	Создать двоичный файл из случайно заданных значений целого типа. Записать в другой файл значения исходного файла, кратные трем. Порядок следования чисел сохранить
14	Из двоичного файла, элементами которого являются целые числа, переписать в другой файл элементы исходного файла, стоящие на четных местах

Таблица 10.3. Варианты повышенной сложности

Вариант	*Задание*
1	Создать двоичный файл из случайно заданных значений целого типа. Записать их в другой файл, исключив повторные вхождения элементов. Порядок следования чисел сохранить
2	Создать двоичный файл из случайно заданных значений целого типа. Заменить каждое третье число нулем, если оно отрицательное и единицей, если положительное
3	В двоичном файле, элементами которого являются числа, заменить заданным числом: · первый элемент · k-й элемент · пятый элемент · последний элемент Новые значения вводятся с клавиатуры
4	В двоичном файле, элементами которого являются целые числа, удалить число, записанное после первого нуля. Результат записать в другой файл
5	Упорядочить значения двоичного файла, содержащего данные типа char по алфавиту
6	В двоичном файле с элементами типа int заменить четные элементы на 0, а нечетные на 1
7	В файле типа double поменять местами максимальный и минимальный элементы
8	Из файла, содержащего как положительные, так и отрицательные вещественные числа, сформировать два других, в один из которых поместить значения, большие, чем число введенное с клавиатуры, в другой – меньшие. Определить количество элементов в обоих файлах
9	В конец целочисленного двоичного файла дописать нечетные значения, содержащиеся в этом файле
10	В двоичном файле с элементами типа float заменить максимальный и минимальный элементы средним арифметическим значением элементов этого файла

Вариант	*Задание*
11	Создать двоичный файл из случайно заданных значений целого типа. Создать другой файл, в который поместить все элементы исходного файла, порядковый номер которых кратен 3, упорядочив их по убыванию
12	Создать два двоичных файла с элементами символьного типа. Выяснить, совпадают ли их элементы. Если нет, то получить номер первого компонента, в котором эти файлы отличаются друг от друга
13	Создать два двоичных файла с упорядоченными наборами элементов целого типа. Получить новый файл в отсортированном виде путем слияния двух исходных файлов
14	Создать два двоичных файла с наборами элементов целого типа. Создать двоичный файл, в который поместить элементы, встречающиеся только в одном из файлов и не встречающиеся в другом
15	Создать два двоичных файла с наборами элементов целого типа. Создать двоичный файл, в который поместить элементы общие для обоих файлов

Таблица 10.4. Варианты для работы с типами данных, создаваемых пользователями

Вариант	*Задание*
1	Создать двоичный файл, содержащий таблицу с информацией об автомобилях: фамилия владельца, марка автомобиля, дата выпуска, расход горючего. Вывести в один из текстовых файлов список владельцев автомобилей, определенной марки. В другой – информацию об автомобилях, имеющих дату выпуска не ранее XXXX года и расход горючего в заданном диапазоне. Год выпуска, диапазон значений расхода горючего вводится с клавиатуры
2	Создать в двоичном файле таблицу с информацией о клиентах банка: фамилия, размер вклада, процент по вкладу, дата внесения вклада. Увеличить размер вклада на соответствующее количество процентов тем вкладчикам, чей вклад пролежал год. Вывести в текстовый файл список из 3 вкладчиков с максимальным размером вклада

Вариант	*Задание*
3	Создать в двоичном файле таблицу с информацией о вершинах горного массива: название вершины, высота над уровнем моря, категория сложности (от 1 до 5), лучший месяц для восхождения. Упорядочить список по категориям сложности, а затем – по высоте (в исходном файле). Вывести в текстовый файл список вершин заданной категории сложности, которые можно штурмовать в определенном месяце. Категория сложности и месяц вводятся с клавиатуры
4	Создать двоичный файл, содержащий таблицу со списком продуктов магазина: название, стоимость единицы товара, вес единицы товара, дата реализации. Посчитать и вывести в текстовый файл стоимость и вес корзины покупателя. Набор и количество товара ввести с клавиатуры. Вывести в другой текстовый файл список товаров с истекшим сроком хранения
5	Создать двоичный файл, содержащий информацию о продукции цветочного салона: название цветка, стоимость, цвет. Составить варианты трехцветного и двухцветного букетов стоимостью не больше стоимости, заданной с клавиатуры. Сохранить информацию в текстовом файле
6	Создать двоичный файл, содержащий таблицу со списком абитуриентов: фамилия, средний балл аттестата, результат экзамена по русскому языку (зачет/незачет), балл по математике, балл по физике (по 10-балльной системе). Вывести в один из текстовых файлов список абитуриентов, имеющих зачет по русскому языку и суммарный балл по математике и физике ≥ 17. В другой – список абитуриентов со средним баллом аттестата ≥ 4.95
7	Создать двоичный файл, содержащий таблицу сотрудников фирмы: фамилия, пол, количество членов семьи, зарплата. Вывести в один из текстовых файлов список сотрудников с наибольшим количеством членов семьи и минимальной зарплатой. Вывести в другой текстовый файл список сотрудников женского пола с зарплатой ниже средней по фирме
8	Создать двоичный файл, содержащий таблицу туров некоторой туристической фирмы: страна, продолжительность путешествия, дата отправления, стоимость. Вывести в текстовый файл список «горящих» путевок (до отправления осталось 5 дней). Вывести список путешествий в заданном временном интервале и не превышающий определенной стоимости

Вариант	*Задание*
9	Создать двоичный файл, содержащий таблицу с информацией об учениках класса: фамилия, пол, возраст, рост, вес. Упорядочить список в исходном файле по убыванию роста. В класс поступил новый ученик. Вставить его фамилию в список, не нарушая порядка. Вывести в текстовый файл список учеников, рост которых больше вновь поступившего в класс
10	Создать двоичный файл, содержащий таблицу с данными о численности населения (миллионов жителей) и площади (тысяч квадратных километров) нескольких государств. Добавить в исходный файл информацию о плотности населения каждой страны. Вывести в текстовый файл названия трех государств с максимальной плотностью населения
11	Создать двоичный файл, содержащий таблицу с метеорологическими данными некоторого месяца: дата, среднесуточная температура воздуха, скорость ветра, направление ветра, облачность, количество осадков. Вывести в текстовый файл список тех дней, когда выпадали осадки и посчитать их общее количество. Вывести в другой текстовый файл список тех дней, когда среднесуточная температура была больше среднемесячной
12	Создать двоичный файл, содержащий таблицу с телефонами абонентов: фамилия абонента, место жительства (название улицы, номер дома), год установки телефона. Вывести в текстовый файл список тех абонентов, которым установили телефоны в прошлом десятилетии. Вывести в другой текстовый файл список номеров телефонов, которые принадлежат жильцам определенного дома

Лабораторная работа 11. Программирование с использованием функций

Основным модулем программ в языке C++ является функция. *Функция* – это логически завершенный, определенным образом оформленный фрагмент программы, имеющий имя. Функции позволяют разделить большие вычислительные задачи на более мелкие.

Каждая программа на языке C++ обязательно содержит функцию с именем main (главная), с которой начинается выполнение программы. Для всех остальных функций, если они присутствуют в программе, следует объявлять прототипы – описания функций, которые сообщают компилятору имя, а также информацию о возвращаемом значении и аргументах функции.

Синтаксис для прототипа функций с параметрами:

```
тип_возвращаемого_значения имя_функции
                    (список_параметров_с_указанием_типов);
```

Функции в языке C++ бывают стандартные (библиотечные) и программируемые пользователем.

Стандартные функции

Описания стандартных функций находятся в файлах, включаемых в программу с помощью директивы #include. Такие файлы называют заголовочными; они имеют расширение h.

Обращение к имени функции в основной программе называется вызовом функции.

Вызов функций приводит к выполнению некоторых действий или вычислению некоторой величины, используемой затем в программе.

```
y = sin (x);    //функция вычисления синуса
```

В общем виде функции определяются таким образом:

```
тип_возвращаемого_значения имя_функции
                    (тип имя_параметра,...,тип имя_параметра)
{
   тело_функции
}
```

Программируемые функции

Функции, которые программист создает сам, упрощают процесс, поскольку они помогают избежать повторного программирования, так как одну и ту же функцию можно применять в различных программах. Также функции повышают уровень модульности программы, следовательно, облегчают ее чтение, внесение изменений и коррекцию ошибок.

Пример 11.1. Создадим функцию, которая печатает 65 символов «*» в ряд. Чтобы эта функция выполнялась в некотором контексте, она включена в программу печати фирменного бланка. Программа состоит из функций: main() ustars().

Фирменный бланк

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

const int Limit=65;

 void  stars (void);    // прототип функции stars()

 int _tmain(int argc, _TCHAR* argv[])
{stars();
 cout<<"MIET" <<endl;
 stars();
 _getch();
 return 0;
}

// Определение функции stars()
void stars()
 {  int count;
    for (count=1; count<=Limit; count++)
      putchar ('*');
    putchar ('\n');
  }
```

Мы рассмотрели пример простейшей функции, не имеющей аргументов и не возвращающей никаких значений.

Параметры функций

Рассмотрим на примере использование параметров функции.

Пример 11.2. Напишем функцию space(). Ее параметр – количество пробелов, которое должна напечатать эта функция.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

#define address " Zelenograd"
#define name " Moscow Institute Electronic Engineering "
#define department " Informatics and Programming"
const int LIMIT=65;

void stars();
void space (int number);

int _tmain (int argc, _TCHAR* argv[])
```

```
  {  int spaces;
     stars();
     space (25);
     cout<<address<<endl;
     spaces= (LIMIT – strlen (name))/2;
     // Вычислить, сколько нужно пробелов
     space (spaces);
     cout<<name<<endl;
     space ( (LIMIT – strlen (department))/2);
     cout<<department<<endl;
     stars();
     _getch();
     return 0;
}

//Определение функции stars()
void stars()
{  int count;
   for (count=1; count<=LIMIT; count++)
   putchar ('*');
   putchar ('\n');
}
//Определение функции space()
void space (int number)
{  int count;
   for (count=1; count <=number; count++)
putchar ('  ');
}
```

Переменная number называется формальным параметром. Эта переменная приобретает значения фактического аргумента при вызове функции. Другими словами, *формальный аргумент* – переменная в определении вызываемой подпрограммы, а *фактический аргумент* – конкретное значение, присвоенное этой переменной вызывающей программой.

Если для связи с некоторой функцией требуется более одного аргумента, то наряду с именем функции можно задать список аргументов, разделенных запятыми:

```
void  printnum (int i, int j)
{  cout<<"Координаты точек"<< i << j <<endl; }
```

Входные данные передаются функции с помощью *аргументов*, выходная величина возвращается при помощи ключевого слова return.

Возвращение значений с помощью оператора return

Если функция должна возвращать в вызывающую программу некоторое значение, то перед именем функции вместо слова void следует указать тип возвращаемого значения.

Пример 11.3

```
#include "stdafx.h"
#include <conio.h>
```

```
#include <iostream>
using namespace std;

int abs(int x); // прототип функции abs();

int _tmain(int argc, _TCHAR* argv[])
{      int a=10, b=0, c= -22;
     int d, e, f;
     setlocale(LC_ALL, "Russian");
     d = abs(a);
     e = abs(b);
     f = abs(c);
     cout<<d<<' '<<e<<' '<<f<<endl;

     _getch();
     return 0;
}

// функция, вычисляющая абсолютное
// значение числа
int  abs(int x)
{  int y;
    y = (x<0) ? -x : x;
    return y;   // возвращает значение y
                // вызывающей программе
}

//Другая версия функции abs(x):
 int abs(int x)
 {  if (x<0) return -x;
     return   x;
 }
//Третья версия функции abs(x):
int abs(int x)
{  if (x<0) return -x;
   return   x;
    cout<<"А вот эти строки не напечатаются,";
    cout <<endl;
    cout<<"так как этому препятствует "
        <<"наличие оператора return.";
    cout <<endl;
}
```

Применение оператора return без указания значения приводит к тому, что функция, в которой он содержится, завершает свое выполнение и управление возвращается в вызывающую функцию, а поскольку у данного оператора отсутствует возвращаемое выражение, никакое значение в вызывающую функцию не передается.

Тип функции определяется типом возвращаемого ею значения, а не типом ее аргументов.

Если функция не возвращает никакого результата посредством выполнения оператора return **выражение**, то тип возвращаемого ею значе-

ния определяется как void [в том числе и для функции main()]. Иными словами, тип функции не определен.

В языке C++ запрещается определять функции внутри других функций.

Связь между функциями определяется через аргументы, возвращаемые значения и внешние переменные. В исходном файле функции разрешается располагать в любом порядке. Исходную программу можно разбивать на любое число файлов, лишь бы ни одна из функций не оказалась «разрезанной».

Если текст функции находится в том же файле, где и вызывающая функция, причем если текст вызываемой функции расположен в файле раньше, чем текст вызывающей функции, то описание прототипа необязательно. Но описание прототипа является хорошим стилем программирования.

Рассмотренные выше функции содержали объявления переменных внутри определения функции. Такие переменные называются локальными и доступны только в пределах функции, в которой они определены. В следующем примере переменная var1 доступна только внутри функции func(), а var2 активна только в main().

Пример 11.4

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

void func (void)
  { int var1=11;        //локальная переменная var1
     cout<<"var1="<<var1<<endl;
  }
 int _tmain (int argc, _TCHAR* argv[])
     {  int var2=22;   //локальная переменная var2
         cout<<"var2="<<var2<<endl;
         func();
         cout<<"var1="<<var1<<endl;//приведет к
      // возникновению ошибки 'var1':undeclared identifier
         _getch();
         return 0;
      }
```

Переменная, объявленная вне любой из функций, называется глобальной переменной и доступна от точки ее объявления до конца файла. Чтобы сделать var1 доступной как в func(), так и в main(), следует описать эту переменную в начале файла.

Пример 11.5

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;
```

```
int var1=11;        // глобальная переменная var1
void func (void)
  {   cout<<"var1="<<var1<<endl;
   }

 int _tmain (int argc, _TCHAR* argv[])
   {       int var2=22;
    cout<<"var2="<<var2<<endl;
    func();                        //на экран будут выданы
    cout<<"var1="<<var1<<endl; //два одинаковых значения.
     _getch();
     return 0;
}
```

Передача массивов в качестве аргументов функции

Рассмотрим передачу массивов, точнее их адресов, в качестве аргументов функции. Для передачи одномерного массива достаточно объявить функцию, которая примет в качестве параметра массив значений типа double и вычислит сумму элементов этого массива. Прототип этой функции можно записать таким образом:

```
double SumOfData (double data[max]); // при константном
                                     // значении max
```

Еще лучше объявить

```
double SumOfData (double data[ ], int n);
```

При объявлении функции SumOfData() ей можно передать массив аргументов вместе с целым числом n, сообщающим, сколько элементов содержится в передаваемом массиве.

Сама функция SumOfData() может содержать, например, обычный цикл while, суммирующий элементы массива, а функция return возвращает результат:

```
double SumOfData (double data[ ], int n)
{  double sum=0;
    while (n>0)
        sum += data [--n];
    return sum;
 }
```

При передаче двумерных массивов необходимо сообщить компилятору количество столбцов для вычисления адреса элементов в начале каждой строки.

Пусть ROWS – количество строк, COLS – количество столбцов. Объявим функцию с массивом в качестве параметра:

```
void AnyFn (int data[ROWS][COLS]);
```

или

```
void AnyFn (int data[ ][COLS]);
```

Можно передать параметр, определяющий количество строк:

```
void AnyFn (int data[ ][COLS], int numRows);
```

где numRows сообщает функции число строк в массиве data.

Пример 11.6. Передача функции многомерных массивов

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
using namespace std;

const int COLS=8;

void FillArray (int data[ ][COLS], int numRows);
void DisplayTable (int data[ ][COLS], int numRows);
int data1[7][COLS];
int data2[4][COLS];
 int _tmain (int argc, _TCHAR* argv[])
{ time_t t;
// инициализация генератора случайных чисел
     srand (time (&t));
     FillArray (data1, 7);
     DisplayTable (data1, 7);
     FillArray (data2,4);
     DisplayTable (data2,4);
     _getch();
     return 0;

  }
  void FillArray (int data[ ][COLS], int numRows)
   { int r, c;
     for (r=0; r<numRows; r++)
       for (c=0; c<COLS; c++)
            data[r][c]= rand()%100;
// rand() -  генератор случайных чисел
   }
void DisplayTable (int data[ ][COLS], int numRows)
{ int r, c;
    for (r=0; r<numRows; r++)
       { cout<<endl;
          for (c=0; c<COLS; c++)
                 cout<<setw (5)<<data[r][c];
       }
    cout<<endl;
}
```

Понятие об указателях

Указатель – это переменная, содержащая адрес другой переменной. Указателями в языке C++ являются, в частности, имена массивов. *Имя*

массива – это константный указатель. Он указывает на первый элемент массива, индекс которого в языке С++ равен нулю.

При работе с указателями используются два оператора:

& унарный оператор определения адреса переменной,

* унарный оператор раскрытия ссылки.

Записывая функцию языка С++, которая передает параметр в виде массива, можно объявить этот параметр как указатель на базовый тип массива.

Общий синтаксис для прототипа функции с параметром в виде указателя на массив:

```
возвращаемый_тип имя_функции (базовый_тип*,
                              другие_типы_параметров);
```

Общий синтаксис для определения этой функции:

```
возвращаемый_тип имя_функции (базовый_тип *массив,
                                 другие_параметры)
{
   // тело_функции
}
```

Напишем функцию, использующую массивы, а затем перепишем ее, применяя указатели.

Чтобы вызванная функция могла изменять некоторую переменную в вызывающей функции, а также возвращать несколько значений, необходимо передать функции не переменные, а их адреса. При этом формальные параметры, соответствующие передаваемым адресам, должны быть описаны как указатели на переменные данного типа.

В следующем примере приведена программа, в которой вводятся значения целых переменных a и b, и с помощью функции swp() происходит их обмен между собой.

Пример 11.7

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

void swp (int *x, int *y);

int _tmain(int argc, _TCHAR* argv[])
{ int a, b;
  setlocale(LC_ALL, "Russian");
  cout<<"Введите значения a и b ";
  cin>>a>>b;
  swp(&a, &b);
  cout<<"Измененные значения: a= "<<a
      <<"b= "<<b<<endl;
  _getch();
  return 0;
```

```
}

void swp(int *x, int *y)
// x и y - указатели,  *x и *y -
// значения, на которые  они указывают
{ int c;
   c=*x;
  *x=*y;
  *y=c;
}
```

Пример 11.8. Рассмотрим функцию, которая находит среднее значение массива целых чисел. Для входа в функцию задано имя массива и количество его элементов. Результатом работы функции является среднее значение, которое передается при помощи оператора return.

Оператор вызова функции может выглядеть таким образом:

```
cout<<"Среднее из заданных значений"
    <<mean (numbs, size)<<endl;
/* Находим среднее значение массива из n целых чисел.*/
int mean (int array [ ], int n)
{  int index;
long sum;   /*
```

Если массив содержит много элементов, сумма может оказаться довольно большой, поэтому используем тип long int */.

```
if (n>0)
    {  for (index=0, sum=0; index<n; index++)
              sum+=array [index];
         return sum/n;                // Возвращает int.
    }
    else
        {  cout<<"Нет массива."<<endl;
              return 0;
           }
}
```

Перепишем эту же программу с использованием указателей. Объявим iPtr указателем на тип int и заменим элемент массива array[index] на соответствующее значение: * (iPtr+index).

Пример 11.9. Использование указателей для нахождения среднего значения массива n целых чисел.

```
int mean (int *iPtr, int  n)
{  int index;
    long sum;
    if (n>0)
       {  for (index=0, sum=0; index<n; index++)
                 sum+=* (iPtr+index);
          return sum/n; // возвращает целое sum / n
       }
      else
          { cout<<"Нет массива"<<endl;
```

```
            return 0;
        }
}
```

При такой замене в вызове функции ничего менять не придется, так как имя массива и указатель в данном случае взаимозаменяемы. Поэтому можно присваивать указателю адрес первого элемента, используя просто имя массива.

```
Int iarray [10];
int *iPtr = iarray;   // То же самое, что iPtr=&iarray[0].
```

В языке C++ указатели можно увеличивать или уменьшать. Указатель, увеличенный на 3, будет указывать на четвертый элемент массива (эквивалентно &array[3]). Другими словами, увеличивая указатель на единицу, мы в действительности увеличиваем адрес, который он представляет, на размер объекта связанного с ним типа.

Так как указателям типа void не соответствует никакой тип данных, к ним нельзя применять арифметические операции.

Указатель можно индексировать точно так же, как массив. Компилятор в действительности преобразует индексацию в арифметику указателей: iarray[3]=10; представляется как * (iarray+3)=10;

Понятие ссылки

В языке C++ имеется несколько видоизмененная форма указателя, называемая ссылкой. Ссылка на некоторую переменную может рассматриваться как указатель, который при употреблении всегда разыменовывается. Однако для ссылки не требуется дополнительного пространства памяти: она является просто другим именем (псевдонимом) переменной.

Для определения ссылки применяется унарный оператор &.

Пример 11.10

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int value=10;
int &refval=value;   // Ссылка на value


 int _tmain (int argc, _TCHAR* argv[])
  {
    setlocale (LC_ALL, "Russian");
    cout<<"value="<<value<<endl;
refval+=5;           // Модификация через ссылку
cout<<"value="<<value<<endl;
cout<<"Адрес value равен "<<&value<<endl;
cout<<"Адрес refval равен "<<&refval<<endl;
```

```
    _getch();
    return 0;
  }
```

Хотя ссылка может быть определена как псевдоним переменной, прежде всего ссылки применяются для параметров функций и возвращаемых функциями значений. Вызов функции, ожидающей в качестве параметра ссылочную переменную, синтаксически не отличается от вызова, в котором параметр передается значением. Чтобы указать, что функция ожидает передачу параметра ссылкой, в прототипе перед именем переменной ставится модификатор &. Рассмотрим пример использования параметров-ссылок.

Пример 11.11

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

void Inc_val (int i) // Получает параметр-значение
{ i++;               // Модификация не влияет на оригинал
}
void Inc_ptr (int *i) // Получает адрес оригинала
{ (*i)++;      // Модифицирует оригинал путем косвенной
               // адресации
}
void Inc_ref (int & i)  // Получает параметр-ссылку
{i++;                   // Модифицирует оригинал!
}
 int _tmain (int argc, _TCHAR* argv[])
   {
   setlocale (LC_ALL, "Russian");
   int j=10;
   cout<<"j="<<j<<endl;
   Inc_val (j);
   cout<<"После Inc_val (j) j = "<<j<<endl;
   Inc_ptr (&j);
   cout<<"После Inc_ptr (j) j="<<j<<endl;
   Inc_ref (j);
   cout<<"После Inc_ref (j) j="<<j<<endl;
     _getch();
     return 0;
   }
```

При вызове функции, ожидающей передачи ссылки, компьютер неявно передает адрес специфицированного параметра. Внутри самой функции компилятор транслирует все обращения к параметру, разыменовывая переданный адрес. При этом:

- вызывающей функции не требуется применять операцию взятия адреса (&);

- вызываемая функция избавляет от использования косвенных операций (*).

Ссылка может быть использована как тип, возвращаемый функцией. Это позволяет присваивать функции значение.

Пример 11.12.Ссылка как возвращаемый функцией тип.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

const int arraySize = 0xF;
static int valArray[arraySize];
int &valueAt (int indx)
{   return valArray [indx];
}

 int _tmain (int argc, _TCHAR* argv[])
  {
    setlocale (LC_ALL, "Russian");
    for (int i=0; i<arraySize; i++)
         valueAt (i)=1<<i;
for (int i=0; i<arraySize; i++)
cout<<"Значение "<<i<<"= "<<valueAt (i)<<endl;

     _getch();
     return 0;
     }
```

Функцию valueAt (int) можно применять как для чтения элемента с определенным индексом, так и для присваивания нового значения этому элементу.

Контрольные вопросы

1. Каков допустимый уровень вложенности функции в языке C?
2. Как определить функцию? Что такое прототип функции? Всегда ли обязательно объявление прототипов?
3. Как передать информацию функции?
4. В чем разница между формальными и фактическими аргументами? Где описываются аргументы?
5. Где описываются локальные переменные функции?
6. Для чего служит оператор return? Обязательно ли его использование?
7. Проверьте, все ли правильно в следующем определении функции:

```
hallo (num)
{ int num, count;
  for (count =1; count <= num; num++)
       cout<<"Hello, my friend!"<<endl;
}
```

Задания

Каждый логически завершенный фрагмент программы оформить в виде функции. Не использовать глобальные массивы переменных.

Таблица 11.1. Простые варианты

Вариант	*Задание*
1	Заменить минимальный элемент данного целочисленного массива из 15 элементов на сумму предшествующих элементов, а максимальный элемент на сумму последующих элементов
2	Создать массив, состоящий из 20 целочисленных значений. Создать новый массив из номеров элементов исходного массива, имеющих нулевое значение, и сжать исходный массив, удалив все нулевые элементы
3	В массиве из 15 вещественных чисел определить, сколько раз изменяется знак и вывести номера позиций, в которых происходит смена знака
4	В массиве из 20 целых чисел найти среднее арифметическое значение элементов, попадающих во введенный с клавиатуры интервал
5	В массиве из 20 целых чисел найти сумму и количество чисел, находящихся между минимальным и максимальным элементами, включая и сами эти числа
6	Не используя других массивов переставить элементы 10-элементного символьного массива в обратном порядке
7	В массиве из 15 вещественных чисел (положительных и отрицательных) определить наибольшее количество последовательно расположенных положительных чисел
8	Создать массив из 20 символьных значений. Сформировать из его значений три других массива, состоящих: из цифр, из букв и из символов, не являющихся ни буквами, ни цифрами
9	Заменить повторяющиеся элементы исходного 15-элементного массива, состоящего из целых чисел, нулевыми значениями
10	Объединить два упорядоченных по возрастанию целочисленных массива (M1[10] и M2[5]) в один, не нарушая упорядоченности
11	Дан массив int mas[15]. Создать другой массив, поместив в него сначала элементы массива, стоящие на четных местах, затем – стоящие на нечетных местах
12	Заменить нечетные элементы исходного 20-элементного массива, состоящего из целых чисел, нулевыми значениями
13	Объединить два упорядоченных по алфавиту символьных массива (M1[10] и M2[5]) в один, не нарушая упорядоченности

Вариант	*Задание*
14	Дан массив char mas[10]. Создать другой массив, поместив в него сначала цифровые элементы массива, затем – буквенные (считаем, что в массиве содержатся только алфавитно-цифровые символы)

Таблица 11.2. Варианты повышенной сложности

Вариант	*Задание*
1	Задать значения вещественным элементам массивов $A = \{a_i \mid i = 0,1,\ldots,7\}$, $B = \{b_j \mid j = 0,1,\ldots,5\}$, $C = \{c_k \mid k = 0,1,\ldots,9\}$, $D = \{d_n \mid n = 0,1,\ldots,9\}$ и вычислить $P = \dfrac{\prod_{i=0}^{7} a_i + \ln \prod_{j=0}^{5} b_j}{\sin \prod_{k=0}^{9} c_k * \sqrt{\prod_{n=0}^{9} d_n}}$
2	Задать значения целочисленным элементам массивов $A = \{a_i \mid i = 0, 1,\ldots, 8\}$, $B = \{b_j \mid j = 0, 1,\ldots,5\}$, $C = \{c_k \mid k = 0,1,\ldots,6\}$ и вычислить $S = \sum_{i=0}^{8} a_i + tg \dfrac{\sum_{j=0}^{5} b_j}{\sum_{k=0}^{6} c_k}$
3	Задать значения целочисленным элементам массивов $M = \{m_i \mid i = 0, 1,\ldots,7\}$, $L = \{l_j \mid j = 0, 1,\ldots,6\}$, $C = \{c_k \mid k = 0, 1,\ldots,8\}$ и вычислить $Z = \dfrac{\prod_{i=0}^{7}(m_i - 1) + \prod_{k=0}^{8}(c_k - 5)}{\prod_{j=0}^{6} l_j - \prod_{i=0}^{7} m_i}$
4	Задать значения вещественным элементам массивов $A = \{a_i \mid i = 0, 1,\ldots, 6\}$, $B = \{b_j \mid j = 0, 1, 2, 3\}$, $C = \{c_k \mid k = 0, 1, 2,\ldots, 10\}$ и вычислить $Z = \min(\max_{0 \le i \le 6}\{a_i\}, \max_{0 \le j \le 4}\{b_j\}, \max_{0 \le j \le 10}\{c_k\})$
5	Задать значения целочисленным элементам матриц $A=\{a_{ij}\}$, $B = \{b_{ij}\}$, где $i = 0, 1, 2, 3$; $j = 0, 1, 2,\ldots, 6$ и сформировать массивы C и D, состоящие из максимальных элементов столбцов матриц соответственно A и B

Вариант	*Задание*
6	Задать значения вещественным элементам массивов $A = \{a_i\}$ и $B = \{b_i\}$, где $i = 0, 1, 2,\ldots,9$ и вычислить элементы массивов $X = \{x_i\}$ и $Y = \{y_i\}$ по формулам: $x_i = \dfrac{\sin(a_i)}{\sin(a_i)^2 - \sin^2(a_i)}$; $y_i = \dfrac{tg(b_i)}{tg(b_i{}^2) - tg^2(b_i)}$.
7	Задать значения целочисленным элементам матриц $M = \{m_{ij}\}$ и $N = \{n_{ij}\}$, где $i = 0, 1, 2,\ldots, 7$; $j = 0, 1, 2,\ldots, 5$ и сформировать массивы C и D, состоящие из максимальных элементов строк матриц соответственно M и N
8	Задать значения вещественным элементам матриц $A = \{a_{ij}\}$ и $Q = \{q_{ij}\}$, где $i = 0, 1, 2,\ldots, 6$; $j = 0, 1, 2,\ldots, 4$ и сформировать массивы B и R, состоящие из минимальных элементов столбцов матриц соответственно A и Q
9	Задать значения целочисленным элементам матриц $P = \{p_{ij}\}$ и $Q = \{q_{ij}\}$, где $i = 0, 1, 2,\ldots, 4$; $j = 0, 1, 2,\ldots, 7$ и сформировать массивы R и T из сумм отрицательных элементов строк матриц соответственно P и Q
10	Задать значения вещественным элементам матриц $C = \{c_{ij}\}$ и $D = \{d_{ij}\}$, где $i = 0, 1, 2,\ldots, 5$; $j = 0, 1, 2,\ldots, 5$ и сформировать массивы X и Y из произведений положительных элементов столбцов матриц соответственно C и D
11	Задать значения целочисленным элементам матриц $W = \{w_{ij}\}$ и $Z = \{z_{ij}\}$, где $i = 0, 1, 2$; $j = 0, 1, 2,\ldots, 7$ и сформировать массивы T и S соответственно из элементов матриц W и Z, больших заданного числа P
12	Задать значения вещественным элементам матриц $B = \{b_{ij}\}$ и $D = \{d_{ij}\}$, где $i = 0, 1, 2,...,7$; $j = 0, 1, 2, 3$, и сформировать массивы Y и Z, состоящие соответственно из элементов матриц B и D, меньших заданного числа R
13	Задать значения вещественным элементам матриц $P=\{p_{ij}\}$, $Q = \{q_{ij}\}$, где $i = 0, 1, 2,..., 5$; $j = 0, 1, 2,..., 8$ и сформировать массивы R и T, состоящие соответственно из минимальных элементов столбцов матриц P и Q
14	Задать значения целочисленным элементам матриц $A = \{a_{ij}\}$, $B = \{b_{ij}\}$, где $i = 0, 1, 2,..., 5$; $j = 0, 1, 2, 3$ и вычислитьэлементы массивов $X = \{x_{ij}\}$,$Y = \{y_{ij}\}$ по формулам: $x_{ij} = \ln(a_{ij})$; $y_{ij} = \exp(b_{ij})$
15	Задать значения вещественным элементам массивов $A = \{a_i\}$, $B = \{b_i\}$, $C = \{c_i\}$,где $i = 0, 1, 2,...,6$ и вычислить $X = \max\limits_{0\le i\le 6}\{a_i - b_i\} + \max\limits_{0\le i\le 6}\{b_i - c_i\} / \max\limits_{0\le i\le 6}\{a_i - c_i\}$

Вариант	*Задание*
16	Задать значения целочисленным элементам матриц $A = \{a_{ij}\}$, $B = \{b_{ij}\}$, где $i = 0, 1$; $j = 0, 1, 2$ и вычислить элементы матриц $Y = \{y_{ij}\}$, $Z = \{z_{ij}\}$ по формулам $y_{ij} = \sin(a_{ij})$ $z_{ij} = \cos(b_{ij})$
17	Задать значения целочисленным элементам массивов $C = \{c_i\}$ и $D = \{d_i\}$, где $i = 0, 1, 2,..., 6$ и вычислить элементы массивов $S = \{s_i\}$ и $T = \{t_i\}$, где $i = 0, 1, 2,..., 6$, по формуле: $s_i = \dfrac{\ln(c_i) + \ln(d_i)}{\ln(c_i + d_i) * \ln(c_i - d_i)}$; $t_i = \dfrac{\sin(c_i) + \sin(d_i)}{\sin(c_i + d_i) - \sin(c_i - d_i)}$.
18	Задать значения вещественным элементам массивов $X = \{x_i \mid i = 0, 1, 2,..., 5\}$, $Y = \{y_j \mid j = 0, 1, 2,..., 7\}$, $Z = \{z_k \mid k = 0, 1, 2,..., 9\}$, и вычислить $W = \begin{cases} \prod_{i=0}^{5}(\sin(x_i) + 2), & \text{если } \prod_{j=1}^{7}(1 - y_j^2) > 0,5; \\ \prod_{k=0}^{9}(1 - z_k^{\,3}), & \prod_{j=1}^{7}(1 - y_j^2) \le 0,5. \end{cases}$
19	Задать значения вещественным элементам четырех квадратных матриц 5×5 и вычислить сумму квадратов значений той из матриц, у которой наибольшая сумма диагональных элементов
20	Задать значения целочисленным элементам матриц $A = \{a_{ij}\}$, $B = \{b_{ij}\}$, где $i = 0, 1, 2, 3, 4j = 0, 1, 2, 3$ и вычислить величины $S = x_0/x_4 + x_1/x_3 + ... + x_4/x_0$, $T = y_0/y_4 + y_1/y_3 + ... + y_4/y_0$, где x_i, y_i – максимальные элементы i-х строк соответственно матриц A и B

Таблица 11.3. Простые варианты заданий с использованием динамических массивов

Вариант	*Задание*
1	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий индексы элементов матрицы, попадающие в интервал, введенный с клавиатуры
2	Дана матрица 6×9, целого типа. Создать одномерный массив, содержащий индексы четных элементов матрицы
3	Дана матрица 7×8 вещественного типа. Создать другой массив, содержащий элементы матрицы в порядке их следования, пока их количество не превысит число, введенное с клавиатуры
4	Дана матрица 6×8 символьного типа. Создать одномерный массив, содержащий элементы исходной матрицы, являющиеся цифрами или латинскими буквами
5	Дана матрица 7×9 вещественного типа. Создать одномерный массив, содержащий элементы матрицы, большие среднего арифметического значения элементов матрицы
6	Дана матрица 9×9 целого типа. Создать одномерный массив, содержащий элементы матрицы, попадающие в интервал, введенный с клавиатуры
7	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий элементы матрицы, кратные трем
8	Дана матрица 6×6 целого типа. Создать одномерный массив, содержащий элементы матрицы, меньшие ее элемента, стоящего в правом верхнем углу
9	Дана матрица 6×4 вещественного типа. Создать одномерный массив, содержащий элементы матрицы, меньшие среднего арифметического значения нечетных элементов матрицы
10	Дана матрица 6×8 символьного типа. Создать одномерный массив, содержащий символы исходной матрицы, не являющиеся цифрами
11	В данной матрице 7×7 определить координаты элементов, кратных 7. Сформировать массив из полученных значений
12	Дана матрица 5×8 целого типа. Сформировать одномерный массив из элементов матрицы, кратных минимальному значению этой матрицы
13	Дана матрица 6×6 целого типа. Создать одномерный массив, содержащий элементы матрицы, меньшие по модулю, чем разница первого и последнего элементов этой матрицы
14	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий элементы матрицы, которым кратен максимальный элемент этой матрицы

Таблица 11.4. Варианты заданий с использованием динамических массивов (повышенной сложности)

Вариант	*Задание*
1	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий элементы матрицы, кратные числу, введенному с клавиатуры. Для созданного массива определить минимальный элемент и его индекс
2	Дана матрица 6×9, состоящая из единиц и нулей. Создать одномерный массив, содержащий индексы нулевых элементов матрицы. Для созданного массива определить сумму и произведение элементов
3	Дана матрица 7×8 вещественного типа. Создать другой массив, содержащий элементы матрицы в порядке их следования, пока их сумма не превысит число, введенное с клавиатуры. Для созданного массива определить сумму и разность максимального и минимального элементов
4	Дана матрица 6×8 символьного типа. Создать одномерный массив, содержащий элементы исходной матрицы, не являющиеся цифрами или латинскими буквами. Для созданного массива определить количество символов «точка» и количество символов «звездочка»
5	Дана матрица 7×9 вещественного типа. Создать одномерный массив, содержащий элементы матрицы, меньшие среднего арифметического значения элементов матрицы. Для созданного массива определить индекс минимального элемента и индекс максимального элемента
6	Дана матрица 9×9 целого типа. Создать одномерный массив, содержащий элементы матрицы, стоящие до максимального значения матрицы. Для созданного массива определить сумму и количество четных элементов
7	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий индексы элементов матрицы, меньших числа, введенного с клавиатуры. Для созданного массива определить сумму и количество элементов, являющихся точными квадратами
8	Дана матрица 6×6 целого типа. Создать одномерный массив, содержащий элементы матрицы, стоящие до минимального значения матрицы, считая от ее последнего элемента. Для созданного массива определить сумму и произведение нечетных элементов
9	Дана матрица 6×4 вещественного типа. Создать одномерный массив, содержащий элементы матрицы, большие среднего арифметического значения положительных элементов матрицы. Для созданного массива определить сумму и разность его первого и последнего элементов

Вариант	*Задание*
10	Дана матрица 6×8 символьного типа. Создать одномерный массив, содержащий латинские буквы исходной матрицы, стоящие в алфавите раньше символа, введенного с клавиатуры. Для созданного массива сумму элементов, являющихся точными степенями двойки и количество элементов, являющихся точными степенями тройки
11	В данной матрице 7×7 определить координаты точек, которые будучи минимальными в своей строке, являются минимальными и в своем столбце. Сформировать массив из полученных значений. Для созданного массива определить сумму и произведение элементов, кратных 3
12	Дана матрица 5×8. Сформировать одномерный массив из элементов матрицы, являющихся степенями 2. Для созданного массива определить сумму и количество простых элементов
13	Дана матрица 6×6 целого типа. Создать одномерный массив, содержащий элементы матрицы, стоящие до максимального значения матрицы, считая от ее первого элемента. Для созданного массива определить сумму и произведение положительных элементов
14	Дана матрица 6×8 целого типа. Создать одномерный массив, содержащий элементы матрицы, являющиеся степенями 2. Для созданного массива определить минимальный и максимальный элементы

Лабораторная работа 12. Программирование задач с использованием динамических структур данных

Понятие о самоссылочных структурах

Как и обычные структуры, динамические структуры состоят из полей, или членов структур. Но кроме информационных полей они обязательно содержат поля-указатели на свой собственный тип структуры. Поэтому динамические структуры часто называются *самоссылочными* структурами. Среди динамических структур наиболее часто встречаются очереди (списки), стеки, деревья, двунаправленные списки.

Самоссылочную структуру можно объявить таким образом:

```
struct node { int info;    // здесь может находиться любое
                           // информационное поле, и их
                           // может быть больше, чем одно
              struct node *next; // указатель на структуру
                                 // типа node
           }
```

Структура объявленного типа имеет информационное поле целого типа и указатель на такую же структуру, с помощью которого структуры объединяются в списки. Списки очень удобны для хранения различной информации.

Важно, что указатель содержит адрес структуры (иногда называемую узлом) целиком, а не указывает на отдельные компоненты, находящиеся внутри нее.

Формирование очереди

Для создания списка нужно объявить структуру, аналогично структуре node:

```
struct node { int info;
    struct node *next;
    };
```

и указатель на объявленный структурный шаблон:

```
typedef node *NodePtr;  //указатель на тип node
```

После этого объявляют указатель на начало списка, который иногда называют заголовком списка. Назовем указатель head и объявим его:

```
NodePtr head = NULL;
```

Нулевой указатель на начало списка является признаком того, что список пустой. Для начала формирования списка, т. е. для размещения

первого элемента в списке, следует выделить память под этот элемент, предварительно убедившись, что список пустой. Это достигается путем выполнения операторов:

```
if (head == NULL)
   {head = new node;
    head->info = 1;  // или какому-то другому значению
    head->next = NULL;
   }
```

Оператор p->info = 1; эквивалентен оператору (*p).info = 1;. Первый из них называется операцией косвенного выбора и объединяет в себе действия, связанные с разыменованием и выбором поля структуры. Второй оператор осуществляет прямой выбор. Не следует забывать круглые скобки вокруг *p, так как приоритет операции разыменования ниже, чем приоритет операции обращения к полю структуры.

Для выделения памяти можно пользоваться любым способом резервирования памяти.

Список из одного такого элемента имеет вид, представленный на рис. 12.1.

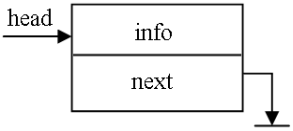


Рис. 12.1. Список из одного элемента

Но список редко состоит из одного элемента. Поэтому следующим шагом является добавление нового элемента в список. Для этого необходимо объявить указатель на текущий элемент p и выполнить следующие действия.

```
NodePtr p;          //указатель на текущий элемент
NodePtr tail;       //указатель на "хвост" очереди
if (head == NULL)
    { head = new node;
      head->info = 1;   // или какому-то другому значению
      head->next = NULL;
    }
p = new node;
p->info = 2;
p->next = head->next;   // в данном случае – NULL
head->next = p;
tail = p;
```

После вставки элемента список приобретает вид, показанный на рис. 12.2.

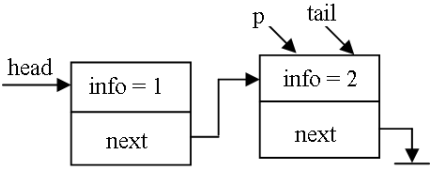


Рис. 12.2. Список из двух элементов

После вставки аналогичным образом ещё нескольких элементов, список приобретает вид, показанный на рис. 12.3. Если при этом известно количество элементов в очереди, используют оператор цикла с параметром:

```
NodePtr p;         // указатель на текущий элемент
NodePtr tail;      // указатель на "хвост" очереди
int N = 10;        // количество элементов в очереди
int cnt = 1;       // счетчик элементов в очереди
head = NULL;
.  .  .
if (head == NULL)
   { head = new node;
     head->info = cnt++; // или какому-то другому значению
     head->next = NULL;
     tail = head;
     }

for (int i = 2; i<=N; i++)
    { p = new node;
     p->info = cnt++;
     tail->next = p;   // в данном случае – NULL
     p->next = NULL;
     tail = p;
    }
```

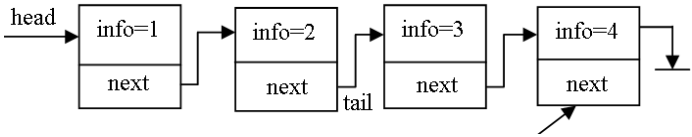


Рис. 12.3. Список из нескольких элементов

Мы создали список или *очередь* (queue, FIFO, First Input-First Output, Первым вошел – Первым вышел), поэтому заголовочный элемент (head) всегда указывает на первый элемент списка.

Пример 12.1. Программа формирования очереди из 10 элементов и вывода ее на экран.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ struct node { int info;
                node *next;
              };
  typedef node *NodePtr; // указатель на тип node
  NodePtr head = NULL;
  NodePtr p;       // указатель на текущий элемент
  NodePtr tail;    // указатель на "хвост" очереди
```

```
  int N = 10;       // количество элементов в очереди
  int cnt = 1;      // счетчик элементов в очереди

  if (head == NULL)
   { head = new node;
      head->info = cnt++;  // или какому-то
                           // другому значению
      head->next = NULL;
      tail = head;
    }
  for (int i = 2; i <= N; i++)
       { p = new node;
         p->info = cnt++;
         tail->next = p;   // в данном случае - NULL
         p->next = NULL;
         tail = p;
       }

  // Вывод очереди на экран
  p = head;
  for (int i = 1; i <= N; i++)
    { cout << p->info << ' ';
      p = p->next;
    }
  cout <<endl;
  _getch();
  return 0;
}
```

Формирование стека

Стеком называется структура данных, организованная по принципу: «Последним вошел – первым вышел» (LIFO, Last Input – First Output). Аналог стека – бусинки, которые нанизаны на нитку. Последнюю из нанизанных бусинок снять легко, а чтобы добраться до самой первой, нужно снять все. Как и очередь, стек организован на основе самоссылочных структур (рис. 12.4). Поэтому для формирования стека объявим такую же структуру, как и для формирования очереди:

```
struct node {  int info;
       struct node *next;
            };
typedef node *NodePtr; //указатель на тип Node
```

Первым в стек помещен элемент с информационным полем, равным 1, вторым – 2 и т. д. Как видно из рис. 12.4, следуя по указателям next от элемента head, мы доберемся до элемента 1 в последнюю очередь.

Формирование стека можно осуществить с помощью следующих операторов:

```
NodePtr head = NULL;
if (head == NULL)
    {     head = new node;
    head->info = random (100) - 50;
```

```
    head->next = NULL;
    }
    else     {    p = new node;
    p->info = random (100) – 50;
    p->next = head;
    head = p;
    }
```

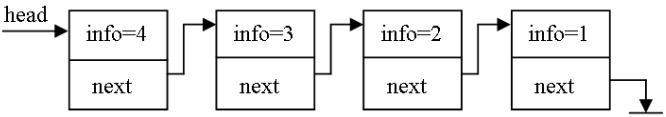


Рис. 12.4. Стек

Выбор оператора цикла при создании стека происходит в зависимости от того, известно ли количество элементов в стеке или известен признак завершения его формирования.

Пример 12.2. Программа формирования стека из 10 элементов и вывода его на экран.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ struct node { int info;
                node *next;
              };
  typedef node *NodePtr;// указатель на тип node
  NodePtr head = NULL;
  NodePtr p;      // указатель на текущий элемент
  int N = 10;     // количество элементов в стеке
  int cnt = 1;    // счетчик элементов в стеке

  if (head == NULL)
   { head = new node;
     head->info = cnt++; //или так: rand()%100-50;
     head->next = NULL;
   }
  for (int i = 2; i <= N; i++)
          { p = new node;
            p->info = cnt++; //или rand()%100-50;
            p->next = head;
            head = p;
          }

  // Вывод стека на экран
  p = head;
  for (int i = 1; i <= N; i++)
    { cout << p->info << ' ';
      p = p->next;
```

```
    }
  cout <<endl;
  _getch();
  return 0;
}
```

Добавление и удаление элементов в односвязных списках

При формировании очереди было показано добавление элемента в ее конец, при формировании стека – в его начало. Но может возникнуть необходимость добавить элемент в любое место формируемого списка. И, если список уже сформирован, то не важно, что он собой представляет – стек или очередь (рис. 12.5, 12.6).

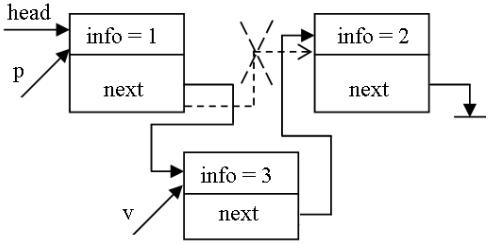


Рис. 12.5. Добавление элемента в середину списка

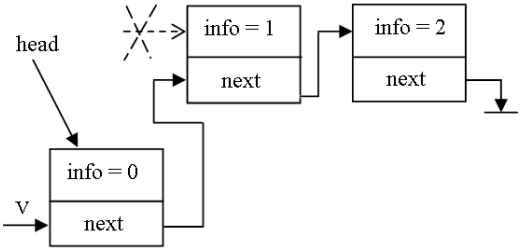


Рис. 12.6. Добавление элемента в начало списка

Действия, связанные с добавлением и удалением элементов стека и очереди аналогичны, поскольку, когда список уже сформирован, доступ к его элементам осуществляется одинаково: путем прохождения от элемента, на который указывает заголовочный указатель до последнего элемента, поле-указатель которого равно NULL.

Для добавления элемента в середину списка нужно перенаправить указатели таким образом, чтобы поле next элемента p, за которым будет располагаться добавляемый элемент v, указывало на него, а поле next элемента v указывало на следующий за ним элемент.

```
NodePtr  v, p;
```

```
   . . .
v = new node;
v->info = 3;
v->next = p->next;
p->next = v;
```

Добавить элемент в начало списка еще проще. Нужно направить указатель head на добавляемый элемент v, а v->next – на элемент, который был первым в списке прежде.

```
NodePtr v,p;
   . . .
v = new node;
v->info = 0;
v->next = head->next;
head = v;
```

При работе со списками весьма распространено не только добавление элементов, но и их удаление, которое также выполняется с помощью перенаправления указателей. Только при этом не следует забывать освобождать память и возвращать ее обратно в «кучу». Удаление элементов из начала, середины и конца списка показано соответственно на рис. 12.7, 12.8 и 12.9 .

```
NodePtr d;    // указатель на удаляемый элемент
   . . .
d = head;
head = head->next;
delete d;
```

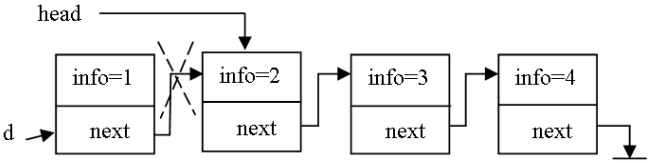


Рис. 12.7. Удаление элемента из начала списка

```
NodePtr d;   // указатель на удаляемый элемент
   . . .
p->next = d->next;
delete d;
```

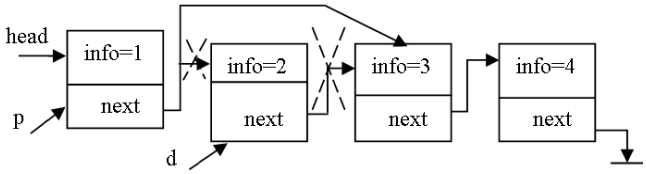


Рис. 12.8. Удаление элемента из середины списка

```
NodePtr d;   // указатель на удаляемый элемент
   . . .
p->next = d->next;  // фактически это  p->next = NULL
delete d;
```

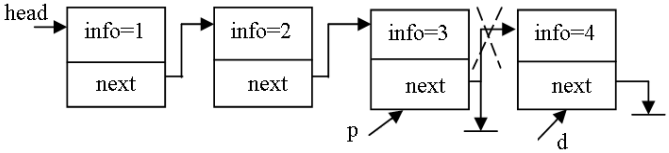


Рис. 12.9. Удаление элемента из конца списка

Не меньшее значение, чем добавление и удаление элементов в связном списке, имеет поиск нужного элемента.

Пример 12.3. Поиск элемента в связном списке.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{ int N;          // количество элементов в списке
  int target;     // искомое значение
  int count;      // местоположение искомого элемента
  struct node { int dat;     // информационное поле
                node *link;  // поле-указатель
              };
   typedef node *NodePtr; //указатель на тип Node
   NodePtr head = NULL;
   NodePtr here;          // указатель на текущий элемент
   cout << "Input a quantity elements:  ";
   cin >> N;
   for (int i = 0; i<N; i++)   // формируем стек
   { if (head == NULL)
     { head = new node;
        if (head == NULL)
{ cout << "Not enough memory !" << endl;
       exit (1);
        }
       head->dat = rand()%100 – 50;
       cout << head->dat << ' ';
       head->link = NULL;
     }
       else
       { here = new node;
          if (here == NULL)
            { cout << "Not enough memory !"
                      << endl;
              exit (2);
            }
          here->dat = rand()%100 – 50;
          cout << here->dat << ' ';
          here->link = head;
          head = here;
       }
   }
```

```
   cout <<endl;

   cout << "Input target ";
   cin >> target;
   here = head;     // встали в начало списка
   count = 1;
   if (here == NULL)
cout << "List is empty!" << endl;
     else {while (here->dat != target
                  && here->link != NULL)
                 { here = here->link;
                    count++;
                 }
           if (here->dat == target)
            cout << "Target element has a number"
                 << count << endl;
            else
              cout<<"Target element is not found!"
                  << endl;
         }
  _getch();

  return 0;
}
```

Контрольные вопросы

1. Что такое динамические структуры?
2. Почему динамические структуры называют самоссылочными?
3. Какой адрес содержится в указателе на структуру?
4. Приведите примеры динамических структур.
5. Перечислите действия, необходимые для создания списка.
6. Сколько информационных полей и какого типа может иметь самоссылочная структура?
7. По какому принципу организован стек и по какому – очередь?
8. Что представляет собой динамическая память?
9. Для чего используется динамическая память?
10. Чем отличаются статические переменные от динамических?
11. Для чего служат указатели?
12. Как можно занять и освободить динамическую память?
13. Как организована структура стека, очереди?
14. Для чего используется списковая организация данных?

Задания

Сформировать динамический список (стек или очередь), считая, что длина списка (количество элементов) задана.

Таблица 12.1. Простые варианты

Вариант	*Задание*
1	В составе программы описать функцию, которая считает сумму элементов списка, стоящих за каждым вхождением элемента E, значение которого введено с клавиатуры
2	В составе программы описать функцию, которая находит среднее арифметическое значение всех элементов сформированного непустого списка
3	В составе программы описать функцию, которая в списке, состоящем из символьных элементов, определяет количество буквенных символов
4	В составе программы описать функцию, которая подсчитывает число вхождений элемента E, значение которого введено с клавиатуры, в списке Q
5	В составе программы описать функцию, которая находит значение и номер минимального элемента в списке
6	В составе программы описать функцию, которая считает среднее арифметическое значение всех четных элементов сформированного непустого списка
7	В составе программы описать функцию, которая считает сумму элементов списка, не равных элементу *E*, значение которого введено с клавиатуры
8	В составе программы описать функцию, которая считает среднее арифметическое значение элементов списка, кратных числу, значение которого введено с клавиатуры
9	В составе программы описать функцию, которая считает количество элементов списка, меньших элемента E, значение которого введено с клавиатуры
10	В составе программы описать функцию, которая формирует список L, включая в него наибольшие из соответствующих элементов из списков L1 и L2. Например, L1={5, 7, 2, 1}, L2={3, 9, 2, 4}, L={5, 9, 2, 4}
11	В составе программы описать функцию, которая находит значение и номер максимального элемента в списке
12	В составе программы описать функцию, которая считает сумму элементов списка, стоящих перед каждым вхождением элемента E, значение которого введено с клавиатуры
13	В составе программы описать функцию, которая в списке, состоящем из символьных элементов, определяет количество цифровых символов

Вариант	*Задание*
14	В составе программы описать функцию, которая заменяет в списке все вхождения элемента E1, значение которого введено с клавиатуры, на элемент E2, значение которого также введено с клавиатуры

Таблица 12.2. Варианты повышенной сложности

Вариант	*Задание*
1	В составе программы описать функцию, которая удаляет из списка за каждым вхождением элемента E, значение которого введено с клавиатуры, один элемент, если такой есть и он отличен от E
2	В составе программы описать функцию, которая вставляет в символьный список за каждым буквенным элементом новый элемент, значение которого в алфавите следует за значением данного элемента, кроме случая, когда данным элементом является буква «z» или «Z»
3	В составе программы описать функцию, которая вставляет в середину списка четные элементы этого списка
4	В составе программы описать функцию, которая формирует список M1 – копию списка M и список M2, представляющий собой «перевернутый» список M
5	В составе программы описать функцию, которая вставляет в список K новый элемент L1 за каждым вхождением элемента L. Значения элементов L и L1 ввести с клавиатуры
6	В составе программы описать функцию, которая формирует список Common, включив в него элементы, которые входят одновременно в список M1 и M2
7	В составе программы описать функцию, которая включает в упорядоченный по убыванию список новое значение, введенное с клавиатуры, таким образом, чтобы не нарушать упорядоченность
8	В составе программы описать функцию, которая объединяет два упорядоченных по убыванию списка в один упорядоченный по невозрастанию список
9	В составе программы описать функцию, которая формирует список L, включая в него элементы, которые входят в один из списков L1 или L2, но не входят в другой
10	В составе программы описать функцию, которая вставляет в список Long за первым вхождением элемента I, значение которого введено с клавиатуры, все элементы списка Short, если I входит в Long

Вариант	*Задание*
11	В составе программы описать функцию, которая оставляет в списке только первые вхождения подряд идущих одинаковых элементов
12	В составе программы описать функцию, которая в списке Group из каждой группы подряд идущих одинаковых элементов оставляет только один
13	В составе программы описать функцию, которая удаляет из списка все вхождения элемента E, значение которого введено с клавиатуры
14	В составе программы описать функцию, которая дублирует вхождение каждого элемента списка One и формирует из этих значений список Double

Лабораторная работа 13. Программирование на языке C++ с использованием классов

Класс – это расширение понятия структуры языка C++. Он позволяет создавать типы и определять функции, которые задают поведение типа. Каждый представитель класса называется объектом.

Определение класса

Определение класса идентично определению структуры в языке C++, но имеет и отличия:

- обычно содержит одну или несколько спецификаций доступа (*public*, *protected*, *private*);
- вместо ключевого слова struct используется слово **class**;
- обычно включает в себя функции (функции-элементы или методы) наряду с данными-элементами;
- обычно в нем имеются некоторые специальные функции, такие как *конструктор* (функция с тем же именем, что и сам класс) и *деструктор* [функция, именем которой является имя класса с префиксом тильда (~)].

Пример 13.1. Определение класса.

```
class str
{   char *s;              //элемент-данное
   public:                //спецификатор открытого доступа
      str (char *word);   //функция-элемент: конструктор
      ~str();             //функция-элемент: деструктор
   void write();          //функция-элемент: метод печати
};
```

Управление доступом

В языке C++ можно ограничить видимость данных и функций класса при помощи меток public, protected, private. Метка-спецификатор доступа применяется ко всем элементам класса, следующим за ней, пока не встретится другая метка или кончится определение класса.

Метка-спецификатор **public** (открытый) используется тогда, когда данные-элементы и функции-элементы класса должны быть доступны для функций-элементов и других функций программы, в которой имеется представитель класса.

Метка-спецификатор **protected** (защищенный) используется в том случае, когда элементы данных и функции-элементы должны быть дос-

тупны для функций-элементов данного класса и классов производных от него.

Метка-спецификатор **private** (закрытый) используется, если данные-элементы и функции-элементы должны быть доступны только для функций-элементов данного класса.

В классе элементы по умолчанию являются закрытыми.

Элементы класса

Элементы класса делятся на две основные категории:

1) данные, называемые данными-элементами;

2) код, называемый функциями-элементами, или методами.

Данные-элементы классов языка C++ идентичны элементам структур языка C++ с некоторыми дополнениями:

- Данными-элементами могут быть перечислимые типы, битовые поля или представители ранее объявленного класса. Также допускается вложенное объявление перечислимого типа данных и создание псевдонимов с помощью typedef.
- Данное-элемент класса может быть указателем или ссылкой на представитель этого класса.

Функция-элемент класса

Функция-элемент класса является функцией, объявленной (описанной) внутри определения класса. Тело функции может также определяться внутри определения класса, в этом случае функция называется встроенной (inline) функцией-элементом. Когда тело функции определяется вне тела класса, перед именем функции ставится префикс из имени класса и операции разрешения видимости (::).

Пример 13.2

```
class str
{   char *s;          // указатель на строку
   public:
      str (char *word) // встроенный конструктор
          { s=new char[strlen (word)+1];
               strcpy (s, word);
           };
   ~str()
   { delete [ ]s; };  // встроенный деструктор
   void write();      // объявление функции-элемента
   };
void str::write()     // определение функции-элемента
{  cout<<s;
};
```

Доступ к данным-элементам класса

Функции-элементы класса находятся в области действия класса, в котором они определены, т. е. они могут обращаться к любому элементу класса, используя просто имя переменной. Обычные функции или функции-элементы другого класса могут получить доступ к данным-элементам с помощью операции . или –>, применяемых к представителю или указателю на представитель класса.

Пример 13.3

```
class coord
{ public: int x, y;   // координаты x и y
};
void main()
{  coord org;     // представитель класса координат
   coord *orgptr = &org;  // указатель на представитель
                          // класса
   org.x = 0;         // задание значения координаты x
   orgptr–>y = 0;  }  // задание значения координаты y
```

Вызов функций-элементов класса

Функции-элементы класса могут вызывать другие функции-элементы того же класса, используя имя функции.

Пример 13.4

```
    class coord
{   int x, y;    // координаты x и y
   public:
   void setcoord (int _x, int _y)  // функция задания
                                   // значений координат
   {  x =_x; y =_y;
   };
   void getcoord (int &_x, int &_y)  //функция получения
                                     // значений координат
   {_x = x; _y = y;};
   };
void main()
{   coord org;          // представитель класса координат
   coord *orgptr = &org;  // указатель на представитель
                          // класса
   org.setcoord (10, 10); // вызов функции-элемента
                          // задания значений
   int col, row;
   orgptr–>getcoord (col, row);  // вызов функции-элемента
                          // получения значений координат
}
```

Указатель this

Каждая нестатическая (не имеющая спецификатора static) функция-элемент класса имеет доступ к объекту, для которого вызвана, через ключевое слово **this**. Указатель this является указателем на *тип_класса* *.

Пример 13.5

```
class simple
{  public:
   simple();
   void greet() { cout<<" Hello!";};
};
simple::simple()
{  greet();             // вызов
   this->greet();       // функции
   (*this).greet();     // greet()
}
```

Так как функции-элементы могут обращаться ко всем элементам класса просто по имени, в основном указатель this используется для возвращения указателя (return this) или ссылки (return *this) на подразумеваемый объект.

Конструктор

Конструктор инициализирует представитель класса (объект) и является функцией-элементом с тем же именем, что и класс. Конструктор вызывается компилятором всегда, когда создается представитель класса. Объект считается созданным в тот момент, когда завершил работу конструктор объекта.

Для конструкторов выполняются следующие правила:

- для конструктора не указывается возвращаемый тип;
- конструктор не может возвращать значение;
- конструктор не наследуется;
- для одного класса может существовать один или несколько конструкторов;
- если конструктор не задан явным образом, то автоматически создаётся пустой конструктор.

Деструктор

Деструктор является дополнением конструктора. Он имеет то же имя, что и класс, но с префиксом тильда (~). Он вызывается всякий раз, когда уничтожается представитель класса. Объект считается уничтоженным, когда завершил работу деструктор объекта. Для деструктора существуют следующие правила:

- деструктор не может иметь аргументов;
- деструктор не может возвращать значения;

- деструктор не наследуется (исключением является виртуальный деструктор);
- для одного класса может существовать только один деструктор;
- если деструктор не задан явным образом, то автоматически создается пустой деструктор.

Пример 13.6

```
//file ctime.h
#ifndef __CTIME_H__
#define __CTIME_H__
class CTime
{  char *timestr;
public:
  CTime (char *str="00:00:00"); //конструктор по умолчанию
  CTime (const CTime& clk);     //копирующий конструктор
  ~CTime();      //деструктор
  void show();   //функция-элемент
};   //обязательно ставить точку с запятой, так как
     // class - объявление типа
#endif
//file ctime.cpp
#include "stdafx.h"
#include <string.h>
#include <iostream>
using namespace std;
#include "ctime.h"
CTime:: CTime (char *str="00:00:00")
{          timestr=new char[strlen (str)+1];
   strcpy (timestr,str);

}
CTime:: CTime (const CTime& clk)
{  timestr=new char[strlen (clk.timestr)+1];
   strcpy (timestr,clk.timestr);
}
CTime::~ CTime()
{  delete [] timestr;
}
void CTime::show()
{  cout<<"Time is "<<timestr<<endl;
}
//file main.cpp
#include "stdafx.h"
#include "ctime.h"
#include <conio.h>
#include <iostream>
using namespace std;
int _tmain (int argc, _TCHAR* argv[])
{
//для a вызывается конструктор по умолчанию
//для b вызывается конструктор по умолчанию
```

```
   CTime a;
   CTime *b=new CTime;
//для e вызывается копирующий конструктор
   CTime e (a);
//вызовем функцию-элемент
    a.show();    //00:00:00
    b->show();   //00:00:00
    e.show();    //00:00:00
_getch();
    return 0;
}
```

В конце области видимости автоматически вызываются деструкторы объектов в порядке, обратном вызову конструкторов, т. е. сначала для **e**, затем для **d** и т. д.

Форматируемый ввод/вывод. Манипуляторы

При вводе/выводе данных можно воспользоваться *манипуляторами*, т. е. специальными функциями форматирования, которые могут находиться в теле оператора ввода/вывода. Если в манипуляторе используются параметры, то необходимо подключение заголовочного файла <iomanip.h>.

Для сохранения и восстановления состояния потока используется функция-метод класса потока flags(). Например:

```
long a;
a=cout.flags();//для сохранения состояния потока в a
cout.flags (a); //для восстановления состояния потока из a
```

Таблица 13.1. Манипуляторы ввода/вывода

Манипулятор	*Назначение*	*Ввод/ вывод*
dec	Вывод числовых данных в десятичной системе счисления	Вывод
hex	Вывод числовых данных в шестнадцатеричной системе счисления	Вывод
oct	Вывод числовых данных в восьмеричной системе счисления	Вывод
endl	Вывод символа новой строки и флэширование	Вывод
ends	Вывод нуля (NULL)	Вывод
flush	Флэширование	Вывод
ws	Пропуск начальных пробелов	Ввод
resetiosflags (long f)	Сброс флагов, задаваемых в f	Ввод/ вывод
setbase (int основание)	Устанавливает основание системы счисления для вывода данных	Вывод
setfill (char ch)	Устанавливает символ заполнения ch	Вывод

Манипулятор	*Назначение*	*Ввод/ вывод*
setiosflags (long f)	Установка флагов, задаваемых в f	Вывод
setprecision (int p)	Задает число символов после десятичной точки, равным p	Вывод
setw (int w)	Задает ширину поля, равной w позиций	Вывод

Пример 13.7. Вывод данных с использованием манипуляторов.

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <math.h>
#include <conio.h>

using namespace std;

void showflags();

int _tmain (int argc, _TCHAR* argv[])
{   double x, y;
    cout << "Input x ";
    cin >> x;
    y = sin (x);
    cout << setprecision (3);
    cout << setw (7) << x;
    cout << setw (7) << y;
    _getch();
    return 0;
}
```

Пример 13.8. Описать и определить класс-список.

Файл list.h содержит описание класса.

```
#include "stdafx.h"
#ifndef __LIST_H__
#define __LIST_H__
struct list
{
   int inf;      // информационное поле
   list *next;   // указатель на следующий элемент списка
};

class CSpisok
{
   list* head;   // указатель на начало списка
public:
   CSpisok (int);
   CSpisok (const CSpisok&);
  void print();
~CSpisok();
};
#endif
```

Файл list.cpp содержит определение функций-элементов.

```
#include "stdafx.h"
#include <stdlib.h>
#include <iostream>
#include <iomanip>
#include "list.h"
using namespace std;
CSpisok:: CSpisok (int n)
//конструктор инициализирует список из n элементов
// по принципу "очередь"
{  head = NULL;
   list *p,*pn;
   for (int i = 0; i<n; i++)
   {
   p = new list;
   p->inf = rand()%100-50;
   p->next = NULL;
   if (head == NULL) head = p;
   else pn->next = p;
   pn = p;
   }
}
CSpisok:: CSpisok (const CSpisok& s)
//конструктор копии класса CSpisok
{  head = NULL;
   list *sp = s.head, *p, *pn;
   while (sp)
   {   p = new list;
   p->inf = sp->inf;
   p->next = NULL;
   if (head == NULL) head = p;
   else pn->next = p;
   pn = p;
   sp = sp->next;
   }
}
CSpisok::~ CSpisok()
//деструктор – уничтожает объект класса CSpisok из памяти
{  list *p;
   while (head)
   {    p = head;
   head = head->next;
   delete p;
   }
}
//функция-элемент печати содержимого списка
void CSpisok::print()
{  list *p = head;
   while (p)
   {   cout<<setw (5)<<p->inf;
   p = p->next;
   }
   cout<<endl;
}
```

Файл main.cpp содержит основную функцию.

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include "list.h"
#include <conio.h>
#include <time.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  srand (unsigned (time (NULL)));
   CSpisok s1 (10),  // создание списка из 10 элементов
   s2 (s1),          // s2- копия списка s1
   s3 (15);          // создание списка из 15 элементов
   s1.print();      // печать s1
   s2.print();      // печать s2
   s3.print();      // печать s3
_getch();
   return 0;
}
```

В проект включены файлы: main.cpp и list.cpp.

Контрольные вопросы

1. Что представляет собой класс?
2. Какие спецификации доступа используются при описании класса?
3. Что является элементами класса?
4. Как осуществляется доступ к элементам класса?
5. Для чего используется указатель this?
6. Что такое конструктор?
7. Что такое деструктор?

Задания

Таблица 13.2. Простые варианты

Вариант	*Задание*
1	Создать класс «Двигатель автомобиля», включающий данные-элементы: марка а/м, объем топливного бака, расход бензина на 100 км. Функции-элементы: · создание и инициализация (конструктор), · заправка а/м (количество бензина – в аргументе), · расход бензина (пройденный путь – в аргументе), · выдача сообщения о том, сколько километров можно проехать на оставшемся бензине, · деструктор
2	Создать класс «Склад пиломатериалов», включающий данные-элементы: № склада, фамилия директора, максимальный объем хранимых пиломатериалов. Функции-элементы:

Вариант	*Задание*
	· создание и инициализация (конструктор), · ввезенные материалы, м^3 (количество – в аргументе), · вывезенные материалы, м^3 (в аргументе), · выдача сообщения, сколько можно еще завезти материалов и сколько есть в наличии, · деструктор
3	Создать класс «Мобильный телефон», включающий данные-элементы: номер, имя владельца, количество денег на счете. Функции-элементы: · создание и инициализация (конструктор), · пополнение счета (сумма – в аргументе), · оплата разговоров (тариф и время – в аргументе), · выдача сообщения об остатке средств на счете, · деструктор
4	Создать класс «Вагон поезда дальнего следования», включающий данные-элементы: номер вагона, класс вагона, количество мест. Функции-элементы: · создание и инициализация (конструктор), · количество пассажиров, вышедших на промежуточной станции (в аргументе), · количество билетов, проданных на промежуточной станции (в аргументе) · выдача сообщения о количестве свободных мест, · деструктор
5	Создать класс «Платная автостоянка», включающий данные-элементы: название, место расположения, количество мест, тариф. Функции-элементы: · создание и инициализация (конструктор), · количество уехавших машин (в аргументе), · количество машин, желающих встать на стоянку (в аргументе), · выдача сообщения о количестве свободных мест, · деструктор
6	Создать класс «Турникет автобуса», включающий данные-элементы: номер билета, срок действия билета, тип билета (разовый, проездной, социальная карта, транспортная карта). Функции-элементы: · создание и инициализация (конструктор), · анализ «годен/просрочен», · выдача сообщения о сроке действия и виде билета, · выдача сообщения «Турникет открыт»/«Турникет закрыт – билет просрочен», · деструктор

Вариант	*Задание*
7	Создать класс «Банковская карта», включающий данные-элементы: номер, имя владельца, количество денег на счете. Функции-элементы: · создание и инициализация (конструктор), · приход (сумма – в аргументе), · расход (сумма – в аргументе), · выдача сообщения об остатке средств на счете, · деструктор
8	Создать класс «Аэробус», включающий данные-элементы: номер рейса, количество мест I класса, количество мест II класса, количество мест III класса. Функции-элементы: · создание и инициализация (конструктор), · количество пассажиров, вышедших из каждого класса в промежуточном аэропорту (в аргументе), · количество билетов каждого класса, проданных на промежуточной станции (в аргументе) · выдача сообщения о количестве свободных мест, · деструктор
9	Создать класс «Абонемент в солярий», включающий данные-элементы: название, место расположения, тип, тариф, оплаченная сумма. Функции-элементы: · создание и инициализация (конструктор), · пополнение счета (сумма – в аргументе), · оплата услуг солярия (время – в аргументе), · выдача сообщения о возможном (оставшемся) количестве минут в солярии, · деструктор
10	Создать класс «Пончиковый аппарат», включающий данные-элементы: заправка тестом (в граммах), начинкой (в граммах), вес теста на один пончик, вес начинки на один пончик. Функции-элементы: · создание и инициализация (конструктор), · приготовлено обычных пончиков (количество – в аргументе), · приготовлено пончиков с начинкой (количество – в аргументе), · выдача сообщения, сколько можно еще приготовить пончиков разного вида, · деструктор

Вариант	*Задание*
11	Создать класс «Цифровой фотоаппарат», включающий данные-элементы: модель, объем встроенной памяти, объем внешней памяти (карта памяти), количество памяти на одну фотографию. Функции-элементы: · создание и инициализация без карты памяти (конструктор), · установка карты памяти в фотоаппарат (объем – в аргументе), · размещение фотографий во встроенной памяти, пока она не закончилась (количество фотографий – в аргументе), с переходом во внешнюю память, · сообщение о том, сколько еще фотографий можно сделать · деструктор
12	Создать класс «Трехступенчатый ракетный двигатель», включающий данные-элементы: вид топлива, объем каждой из ступеней, скорость к моменту отключения каждой из ступеней, время работы каждой из ступеней. Функции-элементы: · создание и инициализация (конструктор), · выдать сообщение о скорости ракеты через определенное время (заданное в аргументе), считая, что скорость растет линейно, · выдать сообщение о том, сколько времени должно пройти к моменту выхода на заданную орбиту, · выдать сообщение о том, какая ступень работает в данный момент (время – в аргументе), если все ступени уже отключились, то выдать сообщение об орбитальном полете и времени такого полета, · деструктор
13	Создать класс «Кофе-машина», включающий данные-элементы: заправка кофе (в граммах), заправка молоком для капучино (в граммах), вес кофе на одну порцию, вес молока на одну порцию. Функции-элементы: · создание и инициализация (конструктор), · приготовлено обычного кофе (количество чашек – в аргументе), · приготовлено капучино (количество чашек – в аргументе), · выдача сообщения, сколько можно еще приготовить кофе разного вида, · деструктор

Вариант	*Задание*
14	Создать класс «Библиотечный абонемент», включающий данные-элементы: название учебника по C++, автор, шифр, количество на абонементе, количество в читальном зале. Функции-элементы: · создание и инициализация (конструктор), · взятые с абонемента и из читального зала учебники (количество тех и других – в аргументе), · возвращенные учебники (количество тех и других – в аргументе), · выдача сообщения об имеющемся в наличии количестве учебников (на абонементе и читальном зале), · деструктор

Таблица 13.3. Варианты повышенной сложности

Вариант	*Задание*
1	Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем копирования другой строки (объекта класса строки). Предусмотреть функции поиска слова в строке и добавления другой строки, начиная с позиции *N*
2	Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем копирования другой строки (объекта класса строки). Предусмотреть функции слияния двух строк и функцию подсчёта предложений в строке
3	Определить класс-строку. В класс включить два конструктора: для определения класса строки строкой символов и путем копирования другой строки (объекта класса строки). Предусмотреть функции сортировки слов в строке по-алфавиту и подсчёта количества слов
4	Определить класс «Список элементов». В определение класса включить два конструктора: для определения списка по его размеру и путем копирования другого списка. Предусмотреть функции подсчета количества элементов списка и добавления одного списка в другой список, начиная с позиции *N*
5	Определить класс «Список элементов». В определение класса включить два конструктора для определения списка по его размеру и путем копирования другого списка. Предусмотреть функции сортировки списка по возрастанию и вывода на экран в обратном порядке

Вариант	*Задание*
6	Определить класс «Список элементов». В определение класса включить два конструктора для определения списка по его размеру и путем копирования другого списка. Предусмотреть функции инверсии списка (123->321) и поиска подсписка в списке
7	Определить класс «Сортированный список элементов». В определение класса включить два конструктора для определения списка по его размеру и путем копирования другого списка. Предусмотреть функции добавления элемента и слияния двух сортированных списков
8	Определить класс «Список элементов». В определение класса включить два конструктора для определения списка по его размеру и путем копирования другого списка. Предусмотреть функции формирования нового списка из элементов, входящих только в один из двух других списков и вычисления суммы элементов списков
9	Определить класс «Матрица». В класс включить два конструктора для определения матрицы по количеству элементов и путем копирования другой матрицы. Предусмотреть функции вычисления детерминанта матрицы и умножения матрицы на число
10	Определить класс «Стек». В класс включить два конструктора для определения стека по его размеру и путем копирования другого стека. Предусмотреть функции вычисления среднего арифметического из элементов стека и нахождения элемента по его номеру
11	Определить класс «Вектор». В класс включить два конструктора для определения вектора по его размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Предусмотреть функции умножения векторов и подсчёта суммы элементов вектора
12	Определить класс «Вектор». В класс включить два конструктора для определения вектора по его размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Предусмотреть функции нахождения максимального и минимального из элементов вектора и умножения вектора на число
13	Определить класс «Вектор». В класс включить два конструктора для определения вектора по его размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Предусмотреть функции сортировки вектора по возрастанию и нахождения среднего арифметического из элементов вектора

Вариант	*Задание*
14	Определить класс «Квадратная матрица». В класс включить два конструктора для определения матрицы по количеству элементов и путем копирования другой матрицы. Предусмотреть функцию нахождения седловой точки матрицы и функцию, меняющую местами элементы матрицы, симметричные относительно побочной диагонали
15	Определить класс «Квадратная матрица». В класс включить два конструктора для определения матрицы по количеству элементов и путем копирования другой матрицы. Предусмотреть функцию поворота матрицы на 90° и функцию нахождения суммы элементов столбца с минимальным диагональным элементом

Лабораторная работа 14. Перегрузка операторов

Дружественные классы

Разрешить элементам другого класса полный доступ к элементам данного класса, объявленным как private или protected, можно включив в определение данного класса описание friend.

Пример 14.1

```
class myclass
{   friend class another_class;   };
```

Дружественные функции

Разрешить обычной функции или функции-элементу другого класса полный доступ к элементам класса, объявленным private или protected, можно с помощью описания friend в определении данного класса.

Пример 14.2

```
class myclass
{  friend void another_class::member (int);
   friend void func_name (float);
};
```

Для друзей существуют следующие правила:

- на описания friend не влияют спецификаторы public, protected или private;
- описания friend не взаимны: если A объявляет B другом, то это не означает, что A является другом для B;
- дружественность не наследуется: если A объявляет B другом, классы, производные от B, не будут автоматически получать доступ к элементам A;
- дружественность не является переходным свойством: если A объявляет B другом, классы, производные от A, не будут автоматически признавать дружественность B.

Перегрузка операций

Язык C++ позволяет определять и применять к классам обозначения операций. Эта особенность, называемая перегрузкой операций дает классам возможность вести себя подобно встроенному типу данных.

Операции, допускающие перегрузку:

```
+   –   *   /   %   ^   &   |   ~   !   =   <   >   +=   –=   *=   /=   %=
^=   &=   |=   <<   >>   <<=   >>=   ==   !=   <=   >=   &&   ||   ++
––   –>*   –>   ()   []
```

Функции-операции и перегрузка операций подчиняются следующим правилам:

- приоритеты операций и правила ассоциации, принятые для встроенных типов данных, остаются неизменными при оценке выражений с перегруженными функциями-операциями;
- функция-операция не может изменить поведение операции по отношению к встроенным типам данных;
- функция-операция должна быть либо элементом класса, либо воспринимать один или несколько аргументов, имеющих тип класса;
- функция-операция не может иметь аргументов по умолчанию;
- функции-операции наследуются, за исключением operator= ().

Пример 14.3. Описать и определить класс-список; операцию «–» (унарный минус), как сортировку списка по убыванию и операцию «[ ]» (индексирование) – получение значения по заданному номеру.

Файл list.h содержит описание класса.

```
struct list
{  int inf;      // информационное поле
   list *next;   // указатель на следующий элемент списка
};
class spisok
{  list* l;      // указатель на начало списка
public:
   spisok (int);
   spisok (spisok&);
   void print();
   int operator [ ] (int);
   friend void operator – (spisok&);
~spisok();
};
```

Файл list.cpp содержит определение функций-элементов.

```
#include "stdafx.h"
#include <stdlib.h>
#include "list.h"
#include <iostream>
#include <iomanip>
using namespace std;

spisok::spisok (int n)
// конструктор инициализирует список из n элементов
// по принципу "очередь"
{  l = NULL;
   list *p,*pn;
   for (int i = 0; i<n; i++)
   {   p = new list;
   p->inf = rand()%100-50;
   p->next = NULL;
   if (l == NULL) l = p;
   else  pn->next = p;
```

```
   pn = p;
   }
}
spisok::spisok (spisok& s)
//конструктор копии класса spisok
{  l = NULL;
   list *sp = s.l, *p, *pn;
   while (sp)
   {   p = new list;
   p->inf = sp->inf;
   p->next = NULL;
   if (l == NULL) l = p;
   else pn->next = p;
   pn = p;
   sp = sp->next;
   }
}
spisok::~spisok()
//деструктор - удаляет объект класса список из памяти
{  list *p;
   while ( l )
   {   p = l;
   l = l->next;
   delete p;
   }
}
void spisok::print()
//функция-элемент печати содержимого списка
{  list *p = l;
   while (p)
   {
       cout<<setw (4)<<p->inf;
   p = p->next;
   }
   cout<<endl;

}
int spisok::operator [ ] (int n)
// перегруженная операция получения значения по заданному
// номеру n
{  list *p = l;
   for (int i = 1; (i<n )&& (p!=NULL); i++, p = p->next);
   if (p)  return p->inf;
   return -1000;
}
void operator – (spisok& s)
// дружественный перегруженный оператор сортировки
// элементов списка по убыванию
{  list *p = s.l;
   while (p)
   {   list *max = p, *pn = p->next;
   while (pn)
     {   if (pn->inf > max->inf) max = pn;
   pn = pn->next;
```

```
    }
   int i = max->inf;
   max->inf = p->inf;
   p->inf = i;
   p = p->next;
   }
}
```

Файл main.cpp содержит основную функцию.

```
#include "stdafx.h"
#include "list.h"
#include <conio.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{
   srand ( (unsigned)time (NULL));
   setlocale (LC_ALL, "Russian");
   spisok s1 (10),   // создание списка из 10 элементов
   s2 (s1),       // s2- копия списка s1
   s3 (15);       // создание списка из 15 элементов
   s1.print();   // печать s1
   s2.print();   // печать s2
   s3.print();   // печать s3
   cout<<"Значение третьего элемента в s1="
       <<s1[3]<<endl;
   -s3;           // сортировка s3
   s3.print();   // и вывод на экран его элементов
    _getch();
   return 0;
}
```

В проект включены файлы: main.cpp и list.cpp, list.h.

Контрольные вопросы

1. Какие классы и функции называются дружественными?
2. Как осуществляется перегрузка операций?
3. Сколько аргументов требуется для определения перегруженной унарной (бинарной) операции?
4. Чем отличается действие перегруженной операции «++» при ее использовании в префиксной форме от использовании в постфиксной форме?

Задания

Таблица 14.1. Простые варианты

Вариант	*Задание*
1	Перегрузить операцию «+» для класса, содержащего два закрытых элемента, так, чтобы результатом был объект, у которого больше сумма его данных-элементов. Не использовать дружественные функции
2	Перегрузить операцию «–» для объекта, содержащего два закрытых элемента, так, чтобы она возвращала произведение всех закрытых элементов двух объектов. Использовать дружественные функции
3	Перегрузить операцию «>» для объекта, содержащего два закрытых элемента, так, чтобы элементы одного из объектов сдвигались вправо на количество байтов, равное числу, находящемуся в соответствующем поле другого объекта. Не использовать дружественные функции
4	Перегрузить операцию «! =» для объекта, содержащего три закрытых элемента, так, чтобы возвращалось значение Y, если все пары значений сравниваемых объектов различна и N, если есть совпадения. Использовать дружественные функции
5	Перегрузить операцию «+ +» для объекта, содержащего два закрытых элемента, так, чтобы возвращалось значение меньшего из данных-элементов. Не использовать дружественные функции
6	Перегрузить операцию «*» для объекта, содержащего три закрытых элемента, так, чтобы возвращался тот объект, у которого больше произведение данных-элементов. Использовать дружественные функции
7	Перегрузить операцию «%» для объекта, содержащего два закрытых элемента, так, чтобы возвращалось значение Y, если все данные-элементы одного объекта кратны всем данным-элементам другого объекта. Не использовать дружественные функции
8	Перегрузить операцию «/» для объекта, содержащего три закрытых элемента, так, чтобы возвращался объект, у которого первый закрытый элемент больше. Использовать дружественные функции
9	Перегрузить операцию «>>» для класса, содержащего два закрытых элемента, так, чтобы она возвращала сумму результатов целочисленного деления соответственно первого элемента первого объекта на первый элемент второго объекта и второго элемента первого объекта на второй элемент второго объекта. Не использовать дружественные функции

Вариант	*Задание*
10	Перегрузить операцию «+» для объекта, содержащего три закрытых элемента, так, чтобы она возвращала объект, у которого меньше сумма его данных-элементов. Использовать дружественные функции
11	Перегрузить операцию «–» для объекта, содержащего три закрытых элемента, так, чтобы она возвращала сумму всех закрытых элементов двух объектов. Не использовать дружественные функции
12	Перегрузить операцию «<» для объекта, содержащего три закрытых элемента, так, чтобы элементы одного из объектов сдвигались влево на количество байтов, равное числу, находящемуся в соответствующем поле другого объекта. Использовать дружественные функции
13	Перегрузить операцию «= =» для объекта, содержащего два закрытых элемента, так, чтобы возвращалось значение 1, если все пары значений сравниваемых объектов равны и 0, если есть различные пары. Не использовать дружественные функции
14	Перегрузить операцию «- -» для объекта, содержащего три закрытых элемента, так, чтобы возвращалось значение наибольшего из данных-элементов. Использовать дружественные функции
15	Перегрузить операцию «*» для объекта, содержащего два закрытых элемента, так, чтобы возвращалось значение данных-элементов того объекта, у которого их сумма больше. Не использовать дружественные функции
16	Перегрузить операцию «%» для объекта, содержащего три закрытых элемента, так, чтобы возвращалась сумма остатков от целочисленного деления закрытых элементов одного объекта на соответствующие элементы другого объекта. Использовать дружественные функции
17	Перегрузить операцию «/» для объекта, содержащего три закрытых элемента, так, чтобы возвращался тот объект, у которого больше результат деления первого закрытого элемента на второй. Не использовать дружественные функции
18	Перегрузить операцию «<<» для класса, содержащего два закрытых элемента, так, чтобы происходил сдвиг элементов первого объекта влево на число байтов соответствующих элементов второго объекта. Использовать дружественные функции
19	Перегрузить операцию «+» для объекта, содержащего два закрытых элемента, так, чтобы возвращала сумму произведений элементов объектов класса. Не использовать дружественные функции
20	Перегрузить операцию «–» для объекта, содержащего три закрытых элемента, таким образом, чтобы она возвращала объект, в ко-

Вариант	*Задание*
	тором данные-элементы возведены в 3-ю степень. – (минус) унарный, использовать дружественные функции
21	Перегрузить операцию «~» (унарная операция «поразрядная инверсия») для класса, содержащего три закрытых элемента. Не использовать дружественные функции
22	Перегрузить операцию «~» (унарная операция) так, чтобы она изменяла знак закрытых данных-элементов на противоположный для класса, содержащего три закрытых элемента. Использовать дружественные функции
23	Перегрузить операцию «!» для класса, содержащего три закрытых элемента, таким образом, чтобы она возвращала значение 1, если сумма закрытых элементов четная и 0 – если нечетная. Не использовать дружественные функции
24	Перегрузить операцию «*» таким образом, чтобы она возводила закрытые элементы в степень, указанную в аргументе. Использовать дружественные функции. Класс содержит три закрытых элемента
25	Перегрузить операцию «%» для класса, содержащего три закрытых элемента, так, чтобы в результате получился класс, закрытые члены которого равны остатку от целочисленного деления их значений на число, стоящее в аргументе. Не использовать дружественные функции
26	Перегрузить операцию «/» для объекта, содержащего два закрытых элемента, так, чтобы возвращался тот объект, у которого меньше результат деления по модулю первого закрытого элемента на второй. Использовать дружественные функции
27	Перегрузить операцию «>» для класса, содержащего два закрытых элемента, так, чтобы происходил сдвиг элементов первого объекта вправо на число байтов соответствующих элементов второго объекта. Не использовать дружественные функции
28	Перегрузить операцию «++» для объекта, содержащего два закрытых элемента, так, чтобы возвращала сумму элементов объектов класса. Использовать дружественные функции

Таблица 14.2. Варианты повышенной сложности

Вариант	*Задание*
1	Определить класс-строку. Для этого в класс включить два конструктора: для определения строкой символов и путем копирования другой строки (объекта класса строки). Определить операции над строками: >> перевертывание строки (запись символов в обратном порядке); ++ нахождение наименьшего слова в строке

Вариант	*Задание*
2	Определить класс-строку. Для этого в класс включить два конструктора: для определения строкой символов и путем копирования другой строки (объекта класса строки). Определить операции над строками: ++ преобразование символов строки в прописные (заглавные) символы; -- нахождение самого короткого слова в строке
3	Определить класс-строку. Для этого в класс включить два конструктора: для определения строкой символов и путем копирования другой строки (объекта класса строки). Определить операции над строками: + конкатенация двух строк; ++ преобразование символов строки в строчные (маленькие) символы
4	Определить класс-строку. Для этого в класс включить два конструктора: для определения строкой символов и путем копирования другой строки (объекта класса строки). Определить операции над строками: - удаление одной строки из другой (если одна строка является подстрокой другой); -- преобразование символов строки в строчные (маленькие) символы
5	Определить класс «список элементов». В определение класса включить два конструктора: для определения списка по его размеру и путем копирования другого списка. Определить операции над списком: + формирование нового списка из двух списков так, чтобы каждый элемент информационного поля нового списка удовлетворял бы условию c= (a > b) ? a : b. Определить функцию-элемент класса для вставки нового элемента в список на определенное место
6	Определить класс «список элементов». В определение класса включить два конструктора: для определения списка по его размеру и путем копирования другого списка. Определить операции над списком: & формирование нового списка из двух списков так, что каждый элемент информационного поля нового списка удовлетворяет условию c= (a < b) ? a : b. Определить функцию-элемент класса для удаления элемента с определенного места списка

Вариант	*Задание*
7	Определить класс «список элементов». В определение класса включить два конструктора для определения списка по его размеру и путем копирования другого списка. Определить операции над списком: ++ сортировка списка по возрастанию; -- расположение элементов списка в обратном порядке
8	Определить класс «список элементов». В определение класса включить два конструктора: для определения списка по его размеру и путем копирования другого списка. Определить операции над списком: [ ] получение значения информационного поля указанного элемента списка; - удаление из первого списка элементов второго, если второй список входит в первый
9	Определить класс «список элементов». В определение класса включить два конструктора: для определения списка по его размеру и путем копирования другого списка. Определить операции над списком: + конкатенация двух списков; & формирование нового списка из двух списка так, что каждый элемент информационного поля нового списка удовлетворяет условию: c= (a > b) ? a : b
10	Определить класс «матрица». В класс включить два конструктора для определения: по количеству элементов и путем копирования другой матрицы. При задании матрицы предусмотреть ее заполнение случайными числами. Определить операции над матрицей: ++ нахождение наибольшего значения матрицы; + получение новой матрицы, каждый элемент которой равен сумме соответствующих элементов двух других матриц
11	Определить класс «матрица». В класс включить два конструктора: для определения по количеству элементов и путем копирования другой матрицы. При задании матрицы предусмотреть ее заполнение случайными числами. Определить операции над матрицей: -- нахождение наименьшего значения матрицы; - получение новой матрицы, каждый элемент которой равен разности элементов двух других матриц

Вариант	*Задание*
12	Определить класс «стек». В класс включить два конструктора: для определения по его размеру и путем копирования другого стека. Определить операции над стеком: + поместить элемент в стек; -- удалить элемент из стека. Определить две функции-элемента класса для выдачи на экран текущего элемента стека и содержимого стека
13	Определить класс «вектор». В класс включить два конструктора: для определения по размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Определить операции над векторами: & формирование нового вектора так, что каждый элемент нового вектора определяется таким образом: c[I]= (a[I]>b[I])?a[I]:b[I]; ++ определить наибольший элемент вектора
14	Определить класс «вектор». В класс включить два конструктора: для определения по размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Определить операции над векторами: | формирование нового вектора так, что каждый элемент нового вектора определяется таким образом: c[I]= (a[I]>b[I])?b[I]:a[I]; -- определить наименьший элемент вектора
15	Определить класс «вектор». В класс включить два конструктора: для определения по размеру и путем копирования другого вектора. При задании вектора по его размеру предусмотреть его заполнение случайными числами. Определить операции над векторами: [ ] нахождение значения элемента вектора по заданному номеру; ++ сортировка элементов вектора по возрастанию

Лабораторная работа 15. Программирование с использованием наследования классов

Язык C++ позволяет классу наследовать данные-элементы и функции-элементы одного или нескольких других классов. Новый класс называют *производным классом*. Класс, элементы которого наследуются производным классом, называется *базовым классом*. В свою очередь производный класс может служить базовым для другого класса. Наследование дает возможность заключить некоторое общее или схожее поведение различных объектов в одном базовом классе.

Наследование позволяет также изменить поведение существующего класса. Производный класс может переопределить некоторые функции-элементы базового, наследуя, тем не менее, основной объем свойств и атрибутов базового класса.

Общий вид наследования:

```
class Base
{
    // …..
};
class Derived: <ключ доступа> Base
{
    // …
};
```

Ключ доступа может быть private, protected, public. Если ключ не указан, то по умолчанию он принимается private.

Наследование позволяет рассматривать целые иерархии классов и работать со всеми элементами одинаково, приводя их к базовому. Правила приведения следующие:

- наследуемый класс всегда можно привести к базовому;
- базовый класс можно привести к наследуемому, только если в действительности это объект наследуемого класса.

Ошибки приведения базового класса к наследуемому отслеживаются программистом.

Доступ к элементам класса

При наследовании ключ доступа определяет уровень доступа к элементам базового класса внутри производного класса. В табл. 15.1 описаны возможные варианты доступа.

Таблица 15.1. Спецификаторы доступа к элементам класса при наследовании

Наследование	*Доступ*	
	в базовом классе	*в производном классе*
public	public protected private	public protected private
protected	public protected private	protected protected private
private	public protected private	private private private

Конструкторы и деструкторы при наследовании

Конструкторы не наследуются. Если конструктор базового класса требует спецификации одного или нескольких параметров, конструктор производного класса должен вызывать базовый конструктор, используя список инициализации элементов.

Пример 15.1

```
#include <string.h>
class Base
{  public:
   Base (int, float);
};
class Derived: Base
{  public:
   Derived (char* lst, float amt);
};
   Derived:: Derived (char* lst, float amt) :
                        Base (strlen (lst),amt)
   {   }
```

В деструкторе производного класса компилятор автоматически генерирует вызовы базовых деструкторов, поэтому для удаления объекта производного класса следует сделать деструктор в базовых классах виртуальным. Для вызова используется delete this либо operator delete.

Виртуальные функции

Функция-элемент может быть объявлена как **virtual**. Ключевое слово virtual предписывает компилятору генерировать некоторую дополнительную информацию о функции. Если функция переопределяется в производном классе и вызывается с указателем (или ссылкой) базового класса, ссылающимся на представитель производного класса, эта информация

позволяет определить, какой из вариантов функции должен быть выбран: такой вызов будет адресован функции производного класса.

Для виртуальных функций существуют следующие правила:

- виртуальную функцию нельзя объявлять как static.
- спецификатор virtual необязателен при переопределении функции в производном классе.
- виртуальная функция должна быть определена в базовом классе и может быть переопределена в производном.

Пример 15.2. Написать программу с наследованием класса стек от класса массив.

```
#include "stdafx.h"
#include <iostream>
#include <stdlib.h>
#include <time.h>
#include <conio.h>
using namespace std;

class massiv
{  int *num;
   int kol;
   public:
   massiv (int n);
   void print();
   virtual int kolich(){return kol;}
   void put (int k,int n){num[k]=n;}
   ~massiv(){delete num;}
};
massiv::massiv (int n)
{  num=new int[n];
   kol=n;
   for (int i=0;i<kol;i++)  num[i]=rand()%100-50;
}
void massiv::print()
{  for (int i=0;i<kolich();i++)  cout<<num[i]<<" ";
   cout<<endl;
}
class stec:public massiv
{  int top;
   public:
   stec (int);
   virtual int kolich() {return top;}
   void pop (int k);
};
stec::stec (int n):massiv (n)
{  top=0;
}
void stec::pop (int k)
{  put (top++,k);  }
```

```
int _tmain (int argc, _TCHAR* argv[])
{
   srand ( (unsigned)time (NULL));
   massiv a (10);
   a.print();
   stec b (10);
   b.pop (rand()%100-50);
   b.pop (rand()%100-50);
   b.pop (rand()%100-50);
   b.print();
 _getch();
  return 0;
}
```

Главное отличие виртуальной функции от просто перегруженной в том, какая функция будет вызываться при рассмотрении производного класса как базового.

Пример 15.3

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;

class Base
{  public:
   Base(){};
   void Print(){   cout<<"I'm a Base print"<<endl;}
   virtual void View(){ cout<<"I'm a Base view"<<endl;}
};
class Derived: public Base
{  public:
   Derived(){};
   void Print(){   cout<<"I'm a Derived print"<<endl;}
   void View(){ cout<<"I'm a Derived view"<<endl;}
};
int _tmain (int argc, _TCHAR* argv[])
{  Base *A=new Base;
   Derived *B=new Derived;
   Base *C;
   A->Print();
   A->View();
   B->Print();
   B->View();
   C= (Base *)B;
   C->Print();
   C->View();
   _getch();
   return 0;
}
```

Результат:

«I'm a Base print»

«I'm a Base view»

«I'm a Derived print»

«I’m a Derived view»
«I’m a Base print»
«I’m a Derived view»

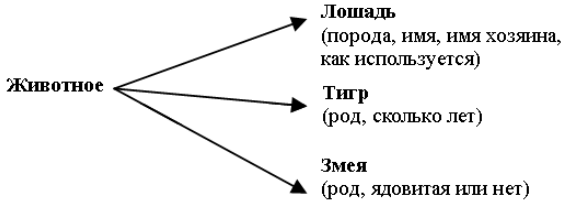
Таким образом, мы видим, что виртуальные функции позволяют нам всегда работать с теми функциями, которые специфичны именно для используемого класса, даже когда мы рассматриваем его как базовый.

Контрольные вопросы

1. Какой класс называется базовым?
2. Какой класс называется производным?
3. Какие ключи доступа используются при наследовании?
4. Наследуются ли конструкторы?
5. Наследуются ли деструкторы?
6. Что собой представляет виртуальная функция?
7. Можно ли виртуальную функцию объявить как static?

Задания

Таблица 15.2. Варианты заданий реализации простого наследования

Вариант	*Задание*
1	Разработать программу с использованием наследования классов, реализующую классы:  Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
2	Разработать программу с использованием наследования классов, реализующую классы:  Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов

Вариант	*Задание*
3	Разработать программу с использованием наследования классов, реализующую классы:  Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
4	Разработать программу с использованием наследования классов, реализующую классы:  Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
5	Разработать программу с использованием наследования классов, реализующую классы:  Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов

Вариант	*Задание*
6	Разработать программу с использованием наследования классов, реализующую классы: **Летательный аппарат** → **Самолет** (название, тип двигателя, пассажирский или транспортный, дальность полета) **Летательный аппарат** → **Дельтаплан** (название, модель, конструктор) **Летательный аппарат** → **Космический корабль** (название, орбитальный или межпланетный, вес, назначение, вид топлива) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
7	Разработать программу с использованием наследования классов, реализующую классы: **Земноводное** → **Крокодил** (ареал обитания, род, длина, вес) **Земноводное** → **Лягушка** (род, ядовитая или нет, цвет, чем питается) **Земноводное** → **Тритон** (ареал обитания, вес, размер, чем питается) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
8	Разработать программу с использованием наследования классов, реализующую классы: **Транспортное средство** → **Велосипед** (название, марка, количество колес) **Транспортное средство** → **Мотоцикл** (название, марка, мощность, максимальная скорость) **Транспортное средство** → **Автомобиль** (название, марка, мощность, грузоподъемность, максимальная скорость) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов

Вариант	*Задание*
9	Разработать программу, реализующую классы с использованием наследования: **Летательный аппарат** → **Помело** (название сказки, кто управляет, скорость, дальность полета) **Летательный аппарат** → **Ступа** (название сказки, кто управляет, вид двигателя, скорость, дальность полета) **Летательный аппарат** → **Ковер-самолет** (название сказки, грузоподъемность, количество посадочных мест, мягкий или нет) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
10	Разработать программу, реализующую классы с использованием наследования: **Военное транспортное средство** → **БТР** (название, назначение, скорость, количество перевозимых людей) **Военное транспортное средство** → **Танк** (название, марка, толщина брони, скорость, дальность поражения) **Военное транспортное средство** → **Самоходное орудие** (название, марка, количество обслуживающего персонала, дальность поражения, скорость) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
11	Разработать программу, реализующую классы с использованием наследования: **Плавсредство** → **Крейсер** (название, грузоподъемность, количество палуб, скорость) **Плавсредство** → **Байдарка** (название, грузоподъемность, марка, количество весел) **Плавсредство** → **Паром** (название, грузоподъемность, вид тяги, количество посадочных мест) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов

Вариант	*Задание*
12	Разработать программу, реализующую классы с использованием наследования: **Графические средства** → **Карандаш** (название, тип: простой или механический); **Авторучка** (название, заправка: чернила или паста); **Краски** (название, состав – акварельные, масляные, гуашь) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
13	Разработать программу, реализующую классы с использованием наследования: **Жилище** → **Дом** (название, общая площадь из какого материала, частный или аренда); **Коттедж** (название, общая площадь, площадь земельного участка, количество этажей); **Юрта** (название, общая площадь, количество жителей) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов
14	Разработать программу, реализующую классы с использованием наследования: **Водоем** → **Море** (название, максимальная глубина, внутреннее или нет, замерзает ли зимой); **Озеро** (название, максимальная глубина, соленое или пресное, площадь поверхности); **Река** (название, максимальная глубина горная или равнинная, протяженность) Используя чистые виртуальные функции, вывести на экран характеристики каждого из объектов

Варианты заданий реализации создания иерархических цепочек

Вариант 1. Разработать программу с использованием наследования классов, реализующую классы:

Человек (Имя, дата рождения)	→	**Школьник (№ школы)**	→	**Студент (Название вуза, специальность)**

Выведите на экран характеристики объектов.

Вариант 2 Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок – координаты начала наследуются из точки (Координаты конца, длина)**	→	**Окружность (Радиус, длина окружности)**

Выведите на экран характеристики объектов.

Вариант 3. Разработать программу с использованием наследования классов, реализующую классы:

Колесо (Радиус)	→	**Садовая тележка (Количество колес, грузоподъемность)**	→	**Грузовик (Марка, мощность)**

Выведите на экран характеристики объектов.

Вариант 4. Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок – координаты начала наследуются из точки (Координаты конца, длина)**	→	**Равносторонний треугольник (Площадь)**

Выведите на экран характеристики объектов.

Вариант 5. Разработать программу с использованием наследования классов, реализующую классы:

Планер (Модель, конструктор, год разработки)	→	**Самолет (Максимальная скорость, максимальная дальность полета)**	→	**Ракета (Назначение, тип топлива, орбитальная или межпланетная)**

Выведите на экран характеристики объектов.

Вариант 6. Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок – координаты начала наследуются из точки (Координаты конца, длина)**	→	**Круг: радиус равен длине отрезка (Площадь)**

Выведите на экран характеристики объектов.

Вариант 7. Разработать программу с использованием наследования классов, реализующую классы:

Рогатка (Дальность боя)	→	**Автомат (Конструктор)**	→	**Баллистическая ракета (Класс, поражающая мощность)**

Выведите на экран характеристики объектов.

Вариант 8. Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок – координаты начала наследуются из точки (Координаты конца, длина)**	→	**Квадрат: сторона равна длине отрезка (Площадь)**

Выведите на экран характеристики объектов.

Вариант 9. Разработать программу с использованием наследования классов, реализующую классы:

Счеты (Разрядность, выполняемые операции)	→	**Калькулятор (Цена)**	→	**Компьютер (Количество процессоров, быстродействие)**

Выведите на экран характеристики объектов.

Вариант 10. Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок: координаты начала наследуются из точки (Координаты конца, длина)**	→	**Ромб: большая диагональ равна длине отрезка, меньшая – половине длины отрезка (Площадь)**

Выведите на экран характеристики объектов.

Вариант 11. Разработать программу с использованием наследования классов, реализующую классы:

Река (Название, период судоходства	→	**Море (Площадь)**	→	**Океан (Максимальная глубина)**

Выведите на экран характеристики объектов.

Вариант 12. Разработать программу с использованием наследования классов, реализующую классы:

Точка (Координаты)	→	**Отрезок: координаты начала наследуются из точки (Координаты конца, длина)**	→	**Прямоугольный треугольник: один из катетов равен длине отрезка, другой – половине длины отрезка (Площадь)**

Выведите на экран характеристики объектов.

Вариант 13. Разработать программу с использованием наследования классов, реализующую классы:

Человек (Имя, дата рождения)	→	**Студент (Название вуза, специальность)**	→	**Аспирант (Тема научной работы)**	→	**Докторант (Ученая степень, ученое звание)**

Выведите на экран характеристики объектов.

Лабораторная работа 16. Работа с файловыми потоками в языке C++. Текстовые файлы

Потоковый ввод/вывод дисковых файлов

Для работы с дисковыми файлами необходимо подключение заголовочного файла <fstream>, содержащего наборы специальных классов:

ifstream – для ввода,

ofstream – для вывода,

fstream – для чтения и записи данных в один и тот же файл.

Чтобы получить возможность работать с дисковым файлом, нужно открыть его с указанием режима доступа (табл. 16.1), который определяется значением константы open-mode класса ios.

Таблица 16.1. Режимы доступа к элементам файлов

Режим доступа	*Стандарт*	*Действие*
app	Нет	Открывает файл для дозаписи
ate (atend)	Да	При открытии файла устанавливает файловый указатель на конец файла
binary (bin)	Да	Открыть файл в двоичном представлении
in	Да	Открыть файл для чтения (ввода)
nocreate	Нет	Если файл не существует, то новый файл не создается
noreplace	Нет	Если файл уже существует, файл не перезаписывается
out	Да	Открыть файл для записи (вывода)
trunc	Нет	Открывает и усекает существующий файл. Новая информация замещает существующую

Текстовые файлы

Создание и запись

Для создания текстового файла определяют объект класса ofstream и передают конструктору класса имя дискового файла в качестве первого параметра и режим доступа в качестве второго параметра:

```
ofstream out_file ("Out.txt", ios::out);
```

Можно объявить константу, определяющую режим открытия файла, например:

```
const ios::open_mode=ios::out | ios::app;
```

После того, как предпринималась попытка открыть файл, следует убедиться в том, что файл открыт и готов для записи (или перезаписи):

```
if (! out_file)
  {  cerr<<"Error: unable to write to out.txt"
         <<endl;
    exit (1);
   }
```

Все сказанное верно и для файлов, открываемых для чтения (или входных файлов):

```
ifstream in_file(“Input.txt”, ios::in);
if (! in_file)
  {  cerr<<"Error: unable to open Input.txt"
         <<endl;
     exit(1);
  }
```

При работе с текстовыми файлами наиболее часто встречаются четыре действия:

1) посимвольное чтение;
2) посимвольная запись;
3) построчное чтение;
4) построчная запись.

Посимвольное чтение текста

Функция get(), которая является методом istream, применяется для посимвольного чтения текстового файла.

Пример 16.1. Посимвольное чтение файла и вывод его на экран.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <stdlib.h>
#include <conio.h>

using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{char sym;
 ifstream in_file ("d:\\Input.txt", ios::in);
 if (!in_file) { cerr << "Error input file"
                      << endl;
                 exit (1);
               }
 while (in_file) { in_file.get (sym);
                   cout << sym;
                 }
 cout << endl;
 _getch();
 return 0;
}
```

Посимвольная запись текста

Функция put(), которая является методом ostream, позволяет осуществлять посимвольную запись данных в текстовый файл.

Пример 16.2. Посимвольная запись данных в текстовый файл.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <conio.h>

using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{setlocale (LC_ALL, "Russian"); // подключение русификатора
 string quote = "Зорко одно лишь сердце. Самого главного
глазами не увидишь. А. де Сент-Экзюпери";
/* в ранних версиях объявление строки quote выглядит так:
char *quote = "Зорко одно лишь сердце.*/
 ofstream out_file ("Out_file.txt", ios::out);
 if (!out_file) { cerr << "Error output file"
                       << endl;
                  exit (1);
                }
 for (unsigned int i = 0; i < quote.size(); i++)
out_file.put (quote[i]);
/* в ранних версиях длина строки quote
определяется по-другому:
 for (int i = 0; i < strlen (quote) + 1; i++)
                       out_file.put (quote[i]); */
 cout << "Конец записи" << endl;
 _getch();
 return 0;
}
```

Построчное чтение файла

Обычно построчное чтение и запись файлов работают быстрее посимвольных действий. Для чтения строки из файла воспользуемся функцией getline(), которая является методом класса ifstream. Функция читает строку (в том числе и разделители), пока не встретит символ новой строки ‘\n’, помещая ее в буфер (первый аргумент функции). Максимальный размер буфера задается как второй аргумент функции.

Пример 16.3. Построчное чтение файла.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <conio.h>
```

```
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{const int LEN = 80;
 char BUF[LEN];
 ifstream in_file ("d:\\Input.txt", ios::in);
 if (!in_file) { cerr << "Error input file"
                      << endl;
                 exit (1);
               }
 while (in_file)
    { in_file.getline (BUF,LEN);
      cout << BUF << endl;
    }
 _getch();
 return 0;
}
```

Построчная запись текста

Пример 16.4. Построчная запись текста в файл.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <string>
#include <stdlib.h>
#include <conio.h>

using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{setlocale (LC_ALL, "Russian"); // подключение русификатора
 ofstream out_file ("Out_file.txt", ios::out);
 if (!out_file){ cerr << "Error output file"
                      << endl;
                 exit (1);
                  }
 out_file << "Я не знаю, где встретиться\n";
 out_file << "Нам придется с тобой,\n";
 out_file << "Глобус крутится-вертится,\n";
 out_file << "Словно шар голубой\n";
 _getch();
 return 0;
}
```

Записываемые строки являются не объектами класса string, а строками типа *char, завершающимися символом ‘\n’.

Признак конца файла

Признак конца файла приходится искать в файлах, открытых для чтения. Этот признак устанавливается в тот момент, когда в файле не осталось больше данных, которые можно считать.

Признак конца файла анализируется в выражении вида

```
while (! In_file.eof())   { … }
```

Для этой цели нельзя пользоваться циклом

```
do { … } while (! In_file.eof()),
```

поскольку файл может оказаться пустым.

Однако проверка на конец файла не анализирует ошибки, которые могут встретиться в процессе чтения файла. Для проверки как конца файла, так и наличия ошибок при его чтении пользуются условием выхода из цикла:

```
while (In_file.good())  { … }
```

Оператор цикла

```
while (In_file)   { … }
```

Выполняется до тех пор, пока нет ошибок, в том числе и конца файла (EOF).

Для работы с файловыми потоками любого из стандартных типов, нужно перегрузить операторы ввода и вывода под требуемый тип данных или воспользоваться шаблоном класса, задаваемым с помощью ключевого слова template.

Контрольные вопросы

1. Что такое поток?
2. Что представляет собой файловый указатель?
3. Перечислить режимы доступа к файлу.
4. Как открыть и как закрыть файл?

Задания

Таблица 16.2. Простые варианты

Вариант	*Задание*
1	Дан текстовый файл, состоящий из нескольких предложений. Каждое предложение оканчивается точкой. Переформатировать файл так, чтобы каждая строка содержала одно предложение
2	Дан текстовый файл произвольного формата. Переформировать файл таким образом, чтобы каждая строка содержала на 1 символ больше предыдущей. Первая строка содержит 1 символ
3	Дан текстовый файл, содержащий число (2 символа), месяц (2 символа), год (2 символа), температура воздуха (3 символа), видимость на дорогах (5 символов), набранных в указанном порядке в строках произвольной длины. Сделать из этого файла таблицу метеоданных

Вариант	*Задание*
4	Дан символьный файл *f*. Подсчитать число вхождений в файл каждой из букв a, b, c, d, e, f. Результат вывести в файл *g* в виде таблицы с комментариями
5	Дан текстовый файл, состоящий из нескольких предложений. Каждое предложение оканчивается точкой, слова разделены пробелами. Переформатировать файл так, чтобы каждая строка содержала одно слово
6	Дан текстовый файл, содержащий число (2 символа), месяц (2 символа), год (2 символа), температура (3 символа), частота пульса (3 символа), набранных в указанном порядке в строках произвольной длины. Сделать из этого файла таблицу состояний человека
7	Дан текстовый файл произвольного формата. Задать с клавиатуры длину первой строки. Переформировать файл таким образом, чтобы каждая строка содержала на 2 символа больше предыдущей
8	Текстовый файл, состоящий из N строк разной длины, выровнять по правому краю по введенной с клавиатуры длине строки
9	Дан текстовый файл, состоящий из нескольких предложений. Вставить в начало каждой строки номер этой строки
10	Дан текстовый файл, состоящий из нескольких предложений. Предложения оканчиваются точкой, вопросительным знаком или восклицательным знаком. Переформатировать файл так, чтобы каждая строка содержала одно предложение
11	Дан текстовый файл, состоящий из нескольких предложений. Вставить в начало строки маркер. Символ маркера ввести с клавиатуры
12	Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g* все четные числа исходного файла, в файл *h* – все нечетные. Порядок следования чисел сохраняется. Записать в файл *g* и *h* комментарии
13	Дан текстовый файл *f*, состоящий из нескольких строк. Создать новый файл *g* и записать в него строки исходного файла, пронумеровав их
14	Дан текстовый файл, содержащий число (2 символа), месяц (2 символа), год (4 символа), часы (2 символа), минуты (2 символа), секунды (2 символа), набранных в указанном порядке в строках произвольной длины. Сделать из этого файла таблицу временных характеристик

Таблица 16.3. Варианты повышенной сложности

Вариант	*Задание*
1	Дан файл *f*, компоненты которого являются целыми числами. Записать в файл *g*, компоненты файла *f*, исключив повторные вхождения чисел
2	Дан файл *f*, компоненты которого являются действительными числами. Найти: 1) наибольшее из значений компонентов *f*; 2) наименьшее из значений компонентов с четными номерами; 3) наибольшее из значений модулей компонентов с нечетными номерами; 4) сумму наибольшего и наименьшего из значений компонентов файла *f*; 5) разность первого и последнего компонента файла *f*
3	Текстовый файл, состоящий из N строк разной длины, выровнять по правому краю
4	Дан текстовый файл произвольного формата. Задать с клавиатуры длину самой длинной строки. Переформировать файл таким образом, чтобы каждая строка содержала на 1 символ меньше предыдущей. Если некоторая строка в этом процессе будет содержать 1 символ, то все последующие строки тоже должны содержать 1 символ
5	Дан текстовый файл, содержащий программу на языке C. Проверить эту программу на соответствие числа открывающих и закрывающих фигурных скобок
6	Дан символьный файл *f*. Найти и записать в файл *g* самое длинное слово файла *f*, снабдив его комментарием
7	Дан файл *f*, компоненты которого являются целыми числами. Получить в файле *g* все компоненты файла *f*: 1) являющиеся четными числами, 2) делящиеся на 3 и не делящиеся на 7, 3) являющиеся точными квадратами. Записать в файл *g* комментарий
8	Дан файл *f*. Создать два файла; записав в первый из них все четные числа в порядке возрастания, а во второй – все нечетные, расположив их в порядке убывания
9	Дан текстовый файл *f*. Переформатировать исходный файл, разделяя его на строки так, чтобы каждая строка содержала столько символов, сколько содержит самая короткая строка исходного файла
10	Дан файл *f*. Создать два файла, записав в первый из них среднее геометрическое всех четных чисел, а во второй – среднее арифметическое всех нечетных чисел

Вариант	*Задание*
11	Дан числовой файл *f*. Выбрать все значения, которые делятся нацело на 2 и 4, но не делятся на 6. Записать эти значения в файл *g*, а все остальные – в файл *h*
12	Дан текстовый файл *f*. Определить, являются ли первые два символа цифрами и если да, то четно ли это число. Если число четное, записать его в файл *g*, если оно нечетное – в файл *h*
13	Дан текстовый файл *f*. Создать новый файл *g* и переписать в него исходный файл в обратном порядке, разделив пробелами
14	Текстовый файл, состоящий из N строк разной длины, выровнять по центру. Максимальной шириной «страницы» считать длину самой длинной строки

Таблица 16.4. Варианты для работы с типами данных, создаваемых пользователями

Вариант	*Задание*
1	Сформировать массив на диске, содержащий сведения о пациентах глазной клиники. Структурный тип содержит поля: фамилия пациента, пол, возраст, место проживания (город), диагноз. Написать программу, которая выбирает с диска и выводит на экран: · количество иногородних, прибывших в клинику; · список пациентов старше *X* (*X* ввести с клавиатуры) лет с диагнозом *J* (*J* ввести с клавиатуры)
2	Сформировать массив на диске, содержащий сведения о сотрудниках института. Структурный тип содержит поля: фамилия работающего, название отдела, год рождения, стаж работы, должность, оклад. Написать программу, которая выбирает с диска и выводит на экран: · список сотрудников пенсионного возраста на сегодняшний день с указанием стажа работы; · средний стаж, работающих в отделе *X* (*X* ввести с клавиатуры)
3	Сформировать массив на диске, содержащий сведения об отправлении поездов дальнего следования с Казанского вокзала. Структурный тип содержит поля: номер поезда, станция назначения, время отправления, время в пути, наличие билетов. Написать программу, которая выбирает с диска и выводит на экран: · время отравления поездов в город *X* во временном интервале от *A* до *B* часов (значения *A*, *B*, *X* вводятся с клавиатуры); · наличие билетов на поезд с номером *XXX* (*XXX* ввести с клавиатуры)

Вариант	*Задание*
4	Сформировать массив на диске, содержащий сведения о том, какие из пяти предлагаемых дисциплин по выбору желает изучать студент. Структурный тип содержит поля: фамилия студента, индекс группы, пять дисциплин, средний балл успеваемости. Выбираемая дисциплина отмечается символом 1, иначе – пробелом. Написать программу, которая выбирает с диска и выводит на экран список студентов, желающих прослушать дисциплину *X* (*X* ввести с клавиатуры). Если число желающих превышает 4-х человек, то отобрать студентов, имеющих более высокий средний балл успеваемости
5	Сформировать массив на диске, содержащий сведения о нападающих команды «Спартак». Структурный тип содержит поля: имена нападающих, число заброшенных ими шайб, число голевых передач, заработанное штрафное время. Написать программу, которая выбирает с диска и выводит на экран четырех лучших игроков по сумме очков (голы + передачи)
6	Сформировать массив на диске, содержащий сведения об ассортименте обуви в магазине. Структурный тип содержит поля: артикул, наименование, количество, стоимость одной пары. Артикул начинается с буквы Д – для дамской обуви, М – для мужской, П – для детской. Написать программу, которая выбирает с диска и выводит на экран: · сведения о наличии и стоимости обуви артикула *X* (*X* ввести с клавиатуры); · ассортиментный список дамской обуви с указанием наименования и имеющегося в наличии числа пар каждой модели
7	Сформировать массив на диске, содержащий сведения о наличии билетов на рейсы аэрофлота. Структурный тип содержит поля: номер рейса, пункт назначения, время вылета, время прибытия, количество свободных мест в салоне. Написать программу, которая выбирает с диска и выводит на экран: · время вылета самолетов в город *X* (*X* ввести с клавиатуры); · наличие свободных мест на рейс в город *X* (*X* ввести с клавиатуры) с временем отправления *Y (Y* ввести с клавиатуры)
8	Сформировать массив на диске, содержащий сведения о личной коллекции книголюба. Структурный тип содержит поля: шифр книги, автор, название, год издания, местоположение (номер стеллажа). Написать программу, которая выбирает с диска и выводит на экран: · местоположение книги, автора *X* (*X* ввести с клавиатуры) названия *Y (Y* ввести с клавиатуры); · список книг автора *X* (*X* ввести с клавиатуры), находящихся в коллекции;

Вариант	*Задание*
	· число книг издания *XX* (*XX* ввести с клавиатуры) года, имеющихся в библиотеке
9	Сформировать массив на диске, содержащий сведения о сдаче студентами сессии. Структурный тип содержит поля: индекс группы, фамилия студента, оценки по пяти экзаменам. Написать программу, которая выбирает с диска и выводит на экран: · фамилии неуспевающих студентов с указанием индексов групп и количества задолженностей; · средний балл, полученный каждым студентом группы *X* (*X* ввести с клавиатуры), и всей группы в целом
10	Сформировать массив на диске, содержащий сведения об ассортименте игрушек в магазине. Структурный тип содержит поля: название игрушки, цена, количество, возрастные границы (2–5). Написать программу, которая выбирает с диска и выводит на экран: · название игрушек, которые подходят детям от 1 до 3 лет; · стоимость самой дорогой игрушки и ее название; · название игрушки, которая по стоимости не превышает *X* руб. и подходит ребенку в возрасте от *A* до *B* лет. Значения *A*, *B*, *X* вводятся с клавиатуры
11	Сформировать массив на диске, содержащий сведения о телефонах абонентов. Структурный тип содержит поля: фамилия абонента, место жительства (название улицы, номер дома), год установки телефона. Написать программу, которая выбирает с диска и выводит на экран: · номер телефона по вводимой с клавиатуры фамилии абонента; · количество установленных телефонов с *XXXX* года; · список номеров телефонов, принадлежащих жильцам определенного дома и улицы. Год, название улицы и номер дома вводятся с клавиатуры
12	Сформировать массив на диске, содержащий сведения о количестве изделий категорий *A*, *B*, *C*, собранных рабочим за месяц. Структурный тип содержит поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Считая заданными значения расценок *SA*, *SB*, *SC* за выполненную работу по сборке единицы изделия категорий *A*, *B*, *C*, выбрать с диска и вывести на экран: · общее количество изделий категорий *A*, *B*, *C*, собранных рабочим цеха; · средний размер заработной платы рабочих цеха *X* (*X* ввести с клавиатуры)

Вариант	*Задание*
13	Сформировать массив на диске, содержащий сведения о количестве изделий, собранных рабочими цеха за неделю. Структурный тип содержит поля: фамилия сборщика, количество изделий, собранных им ежедневно в течение шестидневной недели, т. е. раздельно в понедельник, вторник и т. д. Написать программу, которая выбирает с диска и выводит на экран: · фамилию сборщика и общее количество деталей, собранных им за неделю; · фамилию сборщика собравшего наибольшее количество изделий, и день, когда он достиг наивысшей производительности труда

Лабораторная работа 17. Работа с файловыми потоками в языке C++. Двоичные файлы

Сохранение данных в двоичных файлах

Сохранение в двоичных файлах данных стандартных типов

Для того, чтобы открыть двоичный файл, необходимо задать режим доступа ios::binary (в некоторых компиляторах языка C++ – ios::bin). Двоичные файлы более компактны и в некоторых случаях более удобны для обработки.

Для создания выходного файла создают объект

```
ofstream out_fil ("Outfil.dat",ios::out | ios::binary);
if (! out_fil)  { cerr<<"Error: Outfil.dat"<<endl;
   exit (1);
   }
```

Для того, чтобы открыть существующий двоичный файл для чтения, нужно создать объект

```
ifstream in_fil ("Infil.dat", ios::in | ios::binary);
if (! in_fil)  { cerr<<"Error: Infil.dat"<<endl;
   exit (2);
   }
```

К сожалению, созданные объекты in_fil и out_fil не слишком приспособлены для работы с двоичными файлами и требуют некоторых дополнительных действий, необходимых для корректной работы.

Пример 17.1. Запись значения типа double в двоичный файл.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <conio.h>

using namespace std;

class bin_outstream: public ofstream
{public:
 bin_outstream (const char *fn):
                ofstream (fn, ios::out | ios::binary) {}
 void writeOurDate (const void*, int);
 ofstream &operator<< (double d)
   { writeOurDate (&d, sizeof (d));
     return *this;
   }
};
```

```
int _tmain (int argc, _TCHAR* argv[])
{bin_outstream bin_out ("B_out.dat");
 if (!bin_out)
   { cerr << "Unable to write to B_out.dat" << endl;
     exit (1);
   }
 double d = 5.252;
 bin_out << d;
 bin_out << d*d;
 d = 5.2E-5;
 bin_out << d;
 _getch();
 return 0;
}

void bin_outstream :: writeOurDate (const void *Ptr, int
len)
{ if (!Ptr) return;
  if (len <= 0) return;
  write ( (char*)Ptr, len);
}
```

Пример 17.2. Чтение значений типа double из двоичного файла.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <conio.h>

using namespace std;

class bin_instream: public ifstream
{public:
  bin_instream (const char *fn):
        ifstream (fn, ios::in | ios::binary)  {}
  void readOurDate (void*, int);
  bin_instream &operator>> (double &d)
   { readOurDate (&d, sizeof (d));
     return *this;
   }
};

int _tmain (int argc, _TCHAR* argv[])
{bin_instream bin_in ("B_in.dat");
 if (!bin_in) { cerr << "Unable to open B_in.dat" << endl;
                exit (1);
              }
 double d;
 long count = 0;
 bin_in >> d;
 while (!bin_in.eof())
   { cout << ++count << ":"<< d << endl;
     bin_in >> d;
   }
 _getch();
```

```
 return 0;
}

void bin_instream:: readOurDate (void *p, int len)
{ if (!p)   return;
  if (len <= 0)   return;
  read ( (char*)p, len);
}
```

Для работы с файловыми потоками любого из стандартных типов, нужно перегрузить операторы ввода и вывода под требуемый тип данных или воспользоваться шаблоном класса, задаваемым с помощью ключевого слова template.

Сохранение в двоичных файлах данных, имеющих тип, создаваемый пользователем

Иногда возникает необходимость сохранить в файле данные, структура которых задается программистом. В этих случаях задают класс, содержащий данные и функции, перегруженные операторы ввода и вывода под эти данные.

Пример 17.3. Объявим структуру

```
struct mountine {
    char name[20];      //название горы
    int altitude;       //высота над уровнем моря
    int complicate;     //сложность
    };
mountine mount;
```

Для сохранения информации в двоичном файле выполняют:

```
ofstream fil_out("mountines.txt", ios_base::app);
fil_out << mount.name << ' ' << mount.altitude << ' '
        << mount.complicate  <<"\n";
```

Для сохранения той же информации в двоичном файле выполняют следующее:

```
ofstrem fil_out("mountines.dat", ios_base::app |
                                 ios_base::bynary);
fil_out.write((char *) &mount, sizeof(mountine));
```

Метод write копирует указанное число байтов [в данном случае – sizeof (mountine)] в файл из памяти ЭВМ. Несмотря на то, что сохранение данных происходит в двоичном файле, адрес переменной преобразуется в указатель на тип char.

Для чтения данных из двоичного файла используют метод read:

```
ifstream fil_in (”mountines.dat”, ios_base::binary);
fil_in.read ( (char *) &mount, sizeof (mountine));
```

При записи или чтении классов, не содержащих виртуальных функций, можно использовать тот же самый подход. Чтобы сделать класс потоковым, нужно перегрузить операторы << и >>:

```
friend ostream &operator<< (ostream &, AnyClass &);
friend istream &operator>> (istream &, AnyClass &);
```

Произвольный доступ к элементам файлов

Файловый указатель

Каждый файл имеет два связанных с ним значения: указатель чтения и указатель записи, по-другому называемые файловым указателем или текущей позицией.

При последовательном доступе к элементам файлов перемещение файлового указателя происходит автоматически. Но иногда бывает нужно контролировать его состояние. Для этого используются функции:

seekg() – установить текущий указатель чтения;

tellg() – проверить текущий указатель чтения;

seekp() – установить текущий указатель записи;

tellp() – проверить текущий указатель записи.

Организация доступа к элементам двоичных файлов

Благодаря наличию файлового указателя, в двоичных файлах допустим произвольный доступ к их элементам, который можно реализовать с помощью перегруженных функций – элементов, унаследованных из класса istream:

`istream &seekg (streampos)`
или `stream &seekg (streamoff, ios::seek_dir).`

Типы данных streampos и streamoff эквивалентны значениям типа long, но использовать long в явном виде не рекомендуется из-за неоднозначности работы различных компиляторов. Поэтому их определяют как

```
typedef long streampos;
typedef long streamoff;
```

Первая из перегруженных форм функции seekg позиционирует входной поток на заданном байте, вторая – на смещении относительно одной из трех позиций, определенных значением константы ios::seek_dir (табл. 17.1).

Таблица 17.1

Константа	*Значение*	*Описание*
beg	0	Поиск от начала файла
cur	1	Поиск от текущей позиции файла
end	2	Поиск от конца файла

Для позиционирования внутреннего указателя файла для выходных потоков используют перегруженные функции выходных файловых потоков, унаследованных из класса ostream:

```
ostream &seekp (streampos);
ostream &seekp (streamoff, ios::seek_dir);
```

Пример 17.4. В двоичном файле, содержащем целые числа, заменить максимальное значение файла суммой его четных элементов.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
#include <time.h>
#include <conio.h>

using namespace std;

class bin_stream: public fstream
{public:
  bin_stream (const char *fn):
       fstream (fn, ios::out | ios::in | ios::binary){}
  void doneOurDate (const void*, int, int);
  bin_stream &operator<< (int d)
    { doneOurDate (&d, sizeof (d),0);
      return *this;
    }
  bin_stream &operator>> (int &d)
    { doneOurDate (&d, sizeof (d),1);
      return *this;
    }
};

class bin_outstream: public ofstream
{public:
  bin_outstream (const char *fn):
   ofstream (fn, ios::out | ios::binary) {}
  void writeOurDate (const void*, int);
  ofstream &operator<< (int d)
   { writeOurDate (&d, sizeof (d));
     return *this;
   }
};

int _tmain (int argc, _TCHAR* argv[])
{int i, d, max, i_max=0, sum_even = 0;
 time_t t;
 srand ( (int)time (&t));
 bin_outstream bin_out ("Bin.dat"); // создание файла
 if (!bin_out)
   { cerr << "Unable to write to Bin.dat" << endl;
     exit (1);
   }
 for (i = 0; i < 10; i++)
    { d = rand() % 100;
```

```
      bin_out << d;
      if (d % 2 == 0) sum_even += d;
    }
 cout<<endl;
 cout<<sum_even<<endl;
 bin_out.close();

 bin_stream bin ("Bin.dat"); // обработка файла
 if (!bin)
  { cerr << "Unable to write to Bin.dat" << endl;
    exit (1);
  }
 bin >> max;
 i_max = 0;
 for (i = 1; i < 10; i++)
    { bin >> d;
      if (d > max) { max = d; i_max = i; }
    }
 cout<<endl;
 bin.seekp (sizeof (int) * i_max, ios::beg);
 bin << sum_even;
 bin.seekp (0, ios::beg);
 for (i = 0; i < 10; i++)
    { bin >> d;
      cout <<d <<' ';
    }
 bin.close();
 _getch();
 return 0;
}

void bin_stream:: doneOurDate (const void *Ptr,
                                int len, int sign)
{ if (!Ptr)   return;
if (len <= 0)   return;
if (sign==0) write ( (char*)Ptr, len);
    else  read ( (char*)Ptr, len);
}


void bin_outstream :: writeOurDate (const void *Ptr, int
len)
{ if (!Ptr) return;
  if (len <= 0) return;
  write ( (char*)Ptr, len);
}
```

Контрольные вопросы

1. Что такое поток?
2. Особенности работы с двоичными файлами.
3. Что представляет собой файловый указатель?
4. Как организовать доступ к произвольному месту двоичного файла?

Задания

Таблица 17.2. Простые варианты

Вариант	*Задание*
1	В двоичном файле целого типа найти максимальный и минимальный элементы и их порядковые номера
2	В двоичном файле целого типа определить, что больше сумма четных или нечетных элементов этого файла
3	В двоичном файле целого типа определить, что больше: сумма элементов в его первой половине или во второй
4	В двоичном файле целого типа определить среднее арифметическое значение его элементов, кратных пяти
5	В двоичном файле целого типа определить сумму элементов во второй трети этого файла. Например: 3, 5, 8, ***1, 9, 4,*** 5, 2, 0 – сумма равна 14
6	В первой половине двоичного файла целого типа найти его минимальное значение, во второй – максимальное
7	В двоичном файле целого типа определить сумму его первого, среднего и последнего элементов, считая, что число элементов в файле – нечетно
8	В двоичном файле целого типа определить, что больше: сумма его элементов, стоящих на четных местах, или стоящих на нечетных местах.
9	Даны двоичные файлы *f* и *g* целого типа. Записать в файл *h* разность соответствующих элементов исходных файлов
10	Дан двоичный файл с целыми числами. Вычислить сумму и количество элементов, записанных после его нулевых элементов (принять, что нули в файле имеются)
11	В двоичном файле целого типа определить, что больше сумма четных элементов в его первой половине или сумма нечетных элементов в его второй половине
12	Дан двоичный файл с целыми числами. Определить сумму и количество элементов файла, попадающих во введенный с клавиатуры интервал
13	В двоичном файле целого типа определить сумму элементов в последней четверти этого файла. Например: 3, 5, 8, 1, 9, 4, 5, 2, ***0, 1, 7, 5*** – сумма равна 13
14	Даны двоичные файлы *f* и *g* целого типа. Записать в файл *h* наибольшее из соответствующих элементов исходных файлов

Таблица 17.3. Варианты повышенной сложности

Вариант	*Задание*
1	В двоичном файле целого типа заменить максимальный элемент суммой предыдущих элементов, минимальный – суммой последующих элементов
2	В конец двоичного файла целого типа дописать четные элементы этого файла
3	В начало двоичного файла целого типа дописать нечетные элементы этого файла
4	В середину двоичного файла целого типа поместить элементы этого файла, кратные пяти
5	Перед каждым отрицательным элементом двоичного файла, содержащего целые числа, записать значение числа, введенного с клавиатуры
6	В начало двоичного файла целого типа дописать его минимальное значение, в середину – максимальное
7	В начало двоичного файла целого типа записать элементы, являющиеся делителями максимального элемента этого файла.
8	В середину двоичного файла целого типа записать элементы этого файла, меньшие числа, введенного с клавиатуры
9	Даны двоичные файлы *f* и *g* целого типа. Записать в начало файла *f* положительные компоненты файла *g*, а в конец файла *g* – отрицательные компоненты файла *f* с сохранением порядка их следования
10	Дан двоичный файл с целыми числами. Удалить из него число, записанное после первого нуля (принять, что нули в файле имеются). Результат записать в другой файл
11	Дан двоичный файл с целыми числами. Все его четные элементы заменить нулями. Размер исходного файла неизвестен
12	Дан двоичный файл с целыми числами. Заменить все его элементы, порядковый номер которых кратен 7, на новые значения, которые вводятся с клавиатуры. Размер исходного файла неизвестен
13	Дан двоичный файл с положительными и отрицательными целыми числами. Записать в другой файл сначала отрицательные элементы, а затем положительные
14	Дан двоичный файл с целыми числами. Удалить из него числа, записанные перед каждым нулевым значением (принять, что нули в файле имеются). Результат записать в другой файл
15	В двоичном файле целого типа поменять местами элементы, стоящие на четных местах с элементами, стоящими на нечетных местах

Таблица 17.4. Варианты для работы с типами данных, создаваемых пользователями

Вариант	*Задание*
1	Сформировать массив, содержащий ежедневные метеосводки. Структурный тип содержит поля: число месяца, температура ночная, температура дневная, скорость ветра, видимость на дорогах. Написать программу, которая способна: · выдать самый холодный день с видимостью на дорогах не менее 500 м; · определить день с минимальной среднесуточной температурой и день с максимальной среднесуточной температурой. Вывести всю информацию об этих днях
2	Сформировать массив, содержащий сведения о цветочном магазине. Структурный тип содержит поля: фамилия поставщика, название цветов, количество проданных цветов, цена поставщика, цена магазина. Вывести на экран : · общее количество цветов, полученных магазином от данного поставщика; · выручку магазина от продажи цветов
3	Сформировать массив, содержащий сведения о сотрудниках института. Структурный тип содержит поля: фамилия преподавателя, название кафедры, год рождения, стаж работы, ученая степень, ученое звание, оклад. Написать программу, которая позволяет получить: · список профессоров с указанием стажа работы; · средний оклад сотрудников института
4	Сформировать массив, содержащий информацию о туристских поездках. Структурный тип содержит поля: название тура, необходима ли виза, цена, длительность пребывания, категория отеля, есть ли море, предусмотрены ли экскурсии. Написать программу, которая выводит информацию о турах в соответствии с требованиями: · безвизовый тур с возможностью купаться в море и посещать экскурсии; · тур по Европе, не дороже введенной с клавиатуры стоимости, с категорией отеля, не ниже введенной с клавиатуры
5	Сформировать массив, содержащий сведения о наличии автобусных маршрутов. Структурный тип содержит поля: номер автобуса, пункт отправления, пункт назначения, интервал движения, время в пути. Написать программу, которая выводит информацию: · о возможности добраться от пункта А до пункта Б без пересадок; · будет ли автобус от пункта А до пункта Б в течение времени, введенного с клавиатуры

Вариант	*Задание*
6	Сформировать массив, содержащий сведения о детских садах города. Структурный тип содержит поля: номер, государственный или частный, профиль (обычный, спортивный, художественный, с изучением иностранных языков и т. д.), наличие свободных мест, количество человек в группе, возможность ночного пребывания детей. Написать программу, выдающую информацию: · о наличии мест для двух детей в государственном детском саду спортивного профиля с возможностью ночного пребывания; · о количестве частных детских садов в городе с дополнительными возможностями и наличие свободных мест в них
7	Сформировать массив, содержащий сведения о булочной. Структурный тип содержит поля: номер, название товара, фирма-поставщик, количество полученных единиц данного товара, количество проданных единиц данного товара, цена поставщика на данный товар. Выдать на экран информацию: · о выручке магазина от продажи хлебо-булочных изделий; · есть ли в данный момент булочки, название которых введено с клавиатуры
8	Сформировать массив, содержащий информацию об озерах. Структурный тип содержит поля: название, соленое или пресное, глубина, площадь, судоходное или нет, есть ли места для отдыха, возможна ли рыбалка, замерзает ли зимой. Написать программу, выдающую информацию: · об озерах, где возможна зимняя рыбалка; · озерах, где до места отдыха можно добраться на теплоходе
9	Сформировать массив, содержащий базу данных риэлтера, продающего дачные участки. Структурный тип содержит поля: район, поселок, расстояние от Москвы, шоссе, площадь участка, есть ли дом, наличие электричества, наличие воды, цена. Написать программу, которая укажет: · самый большой участок Истринского района; · участок по Волоколамскому шоссе, не дальше от Москвы, чем введено с клавиатуры, где есть дом и электричество

Вариант	*Задание*
10	Сформировать массив, содержащий данные о горах. Структурный тип содержит поля: название горы, высота, есть ли подъемник, категория сложности, предусмотрены ли горнолыжные трассы, предусмотрены ли альпинистские маршруты. Написать программу, выдающую сведения: · самая высокая гора из тех, по которым проходят альпинистские маршруты; · названия гор, где имеются горнолыжные трассы с подъемниками, с категорией сложности не выше, чем введено с клавиатуры
11	Сформировать массив, содержащий данные о самолетах, участвующих в авиасалоне. Структурный тип содержит поля: место проведения авиасалона, тип самолета, бортовой номер, пилот, воинское звание пилота, опыт пилота (общее количество летных часов), название пилотажной группы. Написать программу, выдающую: · фамилии пилотов пилотажной группы «Стрижи»; · название пилотажных групп, участвующих в авиасалоне, название которого введено с клавиатуры
12	Сформировать массив, содержащий сведения о фильмах на DVD-дисках из частной коллекции. Структурный тип содержит поля: название фильма, фамилия режиссера, год выпуска, время трансляции, цветной или черно-белый. Написать программу, выводящую на экран: · названия всех короткометражных фильмов (короче 30 мин); · названия всех черно-белых фильмов, выпущенных во введенном с клавиатуры году

Лабораторная работа 18. Шаблоны (параметризованные типы)

Использование шаблонов позволяет облегчить процесс создания программ. Создание программы подразумевает собственно процедуру написания кода программы, и последующий процесс ее отладки, модификации и сопровождения.

Механизм шаблонов позволяет отделить общий алгоритм от его реализации применительно к конкретным типам данных, при этом используемый тип данных является в этом случае параметром.

В языке C++ имеются два типа шаблонов – шаблоны функций и шаблоны классов.

Шаблоны функций

Объявление шаблона функции начинается с заголовка, состоящего из ключевого слова **template**, за которым следует список параметров шаблона.

```
// Описание шаблона функции
template <class T>
T min (T a, T b)
{  return a<b ? a : b;  }
```

Ключевое слово **class** в описании шаблона означает тип, идентификатор в списке параметров шаблона T означает имя некоторого типа.

В описании заголовка функции этот же идентификатор означает тип возвращаемого функцией значения и типы параметров функции.

```
...
// Использование шаблона функции
int m, k, l;
cout<<"Input k, l";
cin>>k>>l;
m = min (k, l);
...
Компилятор генерирует (порождает) экземпляр функции для
указанного типа параметров:
int min (int a, int b)
{  return a<b ? a : b;
}
```

В списке параметров шаблона слово class может относиться и к любому из стандартных типов данных. Таким образом, список параметров шаблона <class T> просто означает, что T представляет собой фиктивный тип, который будет определен при вызове. Так как T является параметром, обозначающим тип, шаблоны иногда называют параметризованными типами.

Так как компилятор генерирует экземпляры шаблонов функций согласно типам, заданным при их вызовах, то очень важным моментом яв-

ляется передача корректных типов, особенно если шаблон функции имеет два или более параметров. Хорошим примером является классическая функция max():

```
template <class T>
T max (T a, T b)
{   return a > b ? a : b; }
```

Функция max() будет работать правильно, если оба ее аргумента имеют один и тот же тип:

```
int k,l;
double p,q;
int i = max (k, l);
double d = max (p, q);
```

Однако если аргументы имеют разные типы, то вызов max() приведет к ошибке.

Один из возможных способов решения этой проблемы состоит в использовании приведения типов.

```
int i = max ( (int)'a', 100);
```

Другой способ решить проблему состоит в создании шаблона функций, который имеет параметры различных типов.

```
template <class T1, class T2>
T1 max (T1 a, T2 b)
{   return a > (T1)b ? a : (T1)b;
}
```

Использование этой новой версии max() не приведет к ошибкам трансляции в случае использования двух различных типов. Например, если написать

```
max ('a', 100);
```

то компилятор будет использовать два заданных (посредством аргументов типа) и построит версию функции max() с заголовком

```
char max (char, int);
```

Далее компилятор перед выполнением операции сравнения приведет тип второго аргумента к типу первого аргумента. Формально такой способ допустим, однако использование такой функции принесет скорее вред, чем пользу. К тому же сложно все время держать в памяти, что max ('a', 100) дает значение типа char, в то время как max (100, 'a') передает в вызывающую программу int.

Шаблоны классов

Можно создавать шаблоны как для функций, так и для классов, что позволяет им работать с любыми типами данных. Фактические типы об-

рабатываемых данных при этом задаются в качестве параметров при создании объектов этого класса.

Синтаксис шаблона класса:

```
template <class Ttype> class имя_класса
    {  // описание класса
    }
```

Здесь Ttype – фиктивное имя типа, который будет определен (задан) при создании экземпляра класса. После определения шаблона класса можно создать конкретный экземпляр этого класса:

```
имя_класса <type> объект;
```

где type – имя типа данных, с которым будет оперировать класс.

Функции-элементы родового класса автоматически становятся родовыми. Для них не обязательно указывать ключевое слово template.

Пример 18.1. Реализация класса stack в виде шаблона для хранения объектов любого типа.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <conio.h>

using namespace std;

#define SIZE 10

// создание шаблона класса stack
template <class StackType> class stack
{ StackType stck[SIZE];  // массив, содержащий стек
  int tos;               // индекс вершины стека
 public:
  void init() {tos = 0;}     // инициализация стека
  void push (StackType ob);   // поместить объект в стек
  StackType pop();           // извлечь объект из стека
};

template <class StackType>
void stack <StackType>:: push (StackType ob)
{ if (tos == SIZE)
    { cout << "Stack is full" << endl;
      return;
    }
  stck[tos] = ob;
  tos++;
}

template <class StackType>
StackType stack < StackType>:: pop()
{ if (tos == 0)
    { cout << "Stack is empty" << endl;
      return 0;
    }
```

```
  tos--;
  return stck[tos];
}
int _tmain (int argc, _TCHAR* argv[])
{int i;
 char sym = 'a';
 // символьный стек
 stack <char> st_ch;
 st_ch.init();
 for (i = 0; i < 7; i++)
{ st_ch.push (sym); sym++; }
 for (i = 0; i < 7; i++)
 cout << st_ch.pop() << ' ';
 cout << endl;
 // стек типа int
 stack <int> st_int;
 st_int.init();
 for (i = 0; i < 8; i++)  st_int.push (i * 10);
 for (i = 0; i < 8; i++)
cout << st_int.pop() << ' ';
 cout << endl;
 // стек типа double
 stack <double> st_fl;
 st_fl.init();
 for (i = 0; i < 5; i++)  st_fl.push (i * 2.5);
 for (i = 0; i < 5; i++)
cout << st_fl.pop() << ' ';
 cout << endl;
 _getch();
 return 0;
}
```

Пример 18.2. Создание шаблона класса, реализующего односвязный список.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <conio.h>
using namespace std;
template <class DatList> class list
{ public:
   DatList info;
   list *next;
   list (DatList data)
    { info = data;
      next = 0;
    }
   void insert (list *node)
    { node->next = this;  // поместить объект в список
      next = 0;
    }
   list *get_next()      // получить значение поля next
```

```
    { return next; }
   DatList get_info()    // получить значение поля info
    { return info; }
};
int _tmain (int argc, _TCHAR* argv[])
{ list<char> *head, *tail, *p;
  int i;
  head = new list<char> ('a');
  head->info = 'a';
  tail = head;
  for (i = 1; i < 26; i++)
     { p = new list<char> ('a' + i);
       p->insert (tail);
       tail = p;
     }
  p = head;
  while (p) { cout << p->get_info();
              p = p->get_next();
            }
  _getch();
  return 0;
}
```

Контрольные вопросы

1. Что такое шаблоны и с какой целью они используются?
2. Какого типа шаблоны используются в программах?
3. Как оформляются шаблоны функций?
4. Какие требования предъявляются к фактическим параметрам шаблонов?
5. Какие преимущества программы обеспечиваются при использовании шаблонов классов?

Задания

Таблица 18.1. Простые варианты

Вариант	*Задание*
1	Описать параметризованный класс «односвязный список». Предусмотреть выполнение функций формирования списка, вставки элемента в нужное место списка, удаления элемента
2	Опишите параметризованный класс «очередь». Предусмотреть выполнение функций формирования очереди, добавления и удаления элемента
3	Опишите параметризованный класс «стек». Предусмотреть выполнение функций формирования стека, вставки и удаления элемента

Вариант	*Задание*
4	Опишите параметризованный класс «стек» ограниченной емкости (параметр – тип и число). Предусмотреть выполнение функций формирования стека, вставки и удаления элемента
5	Опишите параметризованный класс «геометрическая фигура» на плоскости (параметр – тип и число). Предусмотреть выполнение функции вычисления и вывода площади фигуры
6	Опишите параметризованную функцию сортировки вставкой. Продемонстрируйте использование функции для разных типов данных
7	Опишите параметризованную функцию сортировки выборкой. Продемонстрируйте использование функции для разных типов данных
8	Опишите параметризованную функцию сортировки пузырьком. Продемонстрируйте использование функции для разных типов данных
9	Опишите параметризованную функцию нахождения элемента в неупорядоченном массиве. Продемонстрируйте использование функции для разных типов данных
10	Опишите параметризованную функцию нахождения элемента в упорядоченном массиве. Продемонстрируйте использование функции для разных типов данных
11	Опишите параметризованную функцию замены одного элемента массива на другой. Продемонстрируйте использование функции для разных типов данных
12	Опишите параметризованную функцию инверсии массива элементов. Продемонстрируйте использование функции для разных типов данных
13	Опишите параметризованную функцию вычисления среднего арифметического значения массива элементов. Продемонстрируйте использование функции для разных типов данных
14	Описать параметризованный класс «массив» с двумя аргументами. Предусмотреть выполнение функций формирования массива, поиска заданного значения в массиве
15	Описать параметризованный класс «сортированный массив» с двумя аргументами. Предусмотреть выполнение функций формирования массива, и вставки заданного значения в соответствующее место массива

Лабораторная работа 19. Исключения в языке C++

Механизм обработки исключительных ситуаций в языке C++ предназначен для отслеживания ошибочных вычислительных ситуаций и безопасной реакции программы на них. Традиционно к исключительным ситуациям относятся, например, деление на ноль, вычисление логарифма отрицательного числа, выход за границы массива и т. д. Однако допустимо использование этого механизма для отслеживания любой вычислительной ситуации.

Синтаксически обработка исключений реализуется с помощью ключевых слов **try**, **catch** и **throw**:

```
try { /* блок try, где должен находиться оператор,
         генерирующий исключения */
    }
catch (type1 arg1) { /* блок catch перехвата и обработки
                        исключения типа type1 */
                   }
catch (type2 arg2) { /* блок catch перехвата и обработки
                        исключения типа type2 */
                   }
…
catch (typeN argN) { /* блок catch перехвата и обработки
                        исключения типа typeN */
                   }
```

Оператор throw должен выполняться либо внутри блока try, либо в любой функции, которую этот блок вызывает:

```
throw исключительная_ситуация;
```

Пример 19.1

```
#include "stdafx.h"
#include <iostream>
#include <stdlib.h>
#include <conio.h>
using namespace std;

const int n=10;
int _tmain (int argc, _TCHAR* argv[])
{  setlocale (LC_ALL,"Russian");
   int i;
   int mas[n];
   try {  for (i=0;i<=n;i++) mas[i]=i;
 if (i>=n) throw i;
    if (i<0) throw "Empty array";
}
//обработка массива согласно целевому алгоритму
catch (int ) {cerr<<"Out of array";
```

```
              _getch();
             exit (1);}
catch (char ) {cerr<<"Index is negative";
              _getch();
             exit (2);}
   _getch();
   return 0;
}
```

Приведенный пример легко может обойтись без использования синтаксиса обработки исключений.

В общем случае для каждой ситуации следует подбирать наиболее эффективный и надежный метод обработки ошибок. Исключения очень подходят для случаев, когда нет возможности передать возвращаемое значение, указывающее на то, что произошла ошибка. Эта ситуация наиболее вероятна в конструкторах или в функциях, не имеющих возвращаемого значения. Другой случай эффективного использования исключений, когда генерация ошибки и ее обработчик находятся далеко друг от друга.

Пример 19.2 Создание стека-массива, использующего исключения для отслеживания его верхней и нижней границы.

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <math.h>
#include <conio.h>

using namespace std;

const int RANG = 10; // максимальный размер стека
class Stack
 { int stk[RANG];
   int top;          // индекс вершины стека
  public:
   class Range       // класс исключений для класса Stack,
    {                // тело класса пусто
    };
   Stack() { top = -1; } // конструктор

   void push (int variable) // функция помещения данных
                            // в стек
    { if (top >= RANG - 1)    // если стек заполнен,
                              // генерировать исключение
          throw Range();
      stk[++top] = variable;  // иначе - протолкнуть
                              // число в стек
    }

   int pop()        // функция извлечения данных из стека
    { if (top < 0)   // если стек пуст,
 // генерировать исключение
```

```
          throw Range();
      return stk[top--]; // иначе – извлечь число из стека
    }
 };                  // class Stack

int _tmain (int argc, _TCHAR* argv[])
{setlocale (LC_ALL, "Russian"); // подключение русификатора
 Stack s_int;
 int INP;
 try { for (int i = 0; i < RANG; i++)
         { cout << "Input value";
           cin >> INP;
           s_int.push (INP);
         }
       s_int.push (INP);   // генерация исключения –
                           // стек полон
       for (int i = 0; i < RANG; i++)
          cout << i + 1 << ':'
               << s_int.pop() << endl;
       cout << "11: "  // генерация исключения – стек пуст
            << s_int.pop()<< endl;
     }             // end try
 catch (Stack:: Range)  // обработчик исключений
   { cout <<"Исключение: нарушение границы стека" << endl;
   }
 cout << "Завершение работы программы" << endl;
 _getch();
 return 0;
}
```

Контрольные вопросы

1. Какую цель преследует использование в программе обработки исключений?
2. Как оформляется блок обработки исключений?
3. Что такое обработчики исключений?

Задания

Таблица 19.1. Варианты заданий

Вариант	*Задание*
1	Опишите функцию вычисления тангенса, обработайте ошибку вычисления арктангенса при аргументе, косинус которого равен 0
2	Опишите функцию вычисления арктангенса, обработайте ошибку вычисления арктангенса при аргументе, равном 1 и –1
3	Опишите функцию вычисления логарифма, обработайте ошибку вычисления логарифма 0

Вариант	*Задание*
4	Переопределите оператор «++» для указателя на массив целых, обработайте ошибку выхода за границы массива
5	Опишите функцию анализа номера телефона, обработайте ошибку задания номера в неверном формате [допустимый формат: +7 (495) 555-44-33] и ошибку корректности вводимых данных
6	Опишите оператор «[]» для вектора элементов, обработайте ошибку выхода за границы массива
7	Опишите функцию вычисления котангенса, обработайте ошибку вычисления арктангенса при аргументе, синус которого равен 0
8	Опишите функцию анализа времени, обработайте ошибку задания даты в неверном формате (допустимый формат: дд-мм-гггг) и ошибку корректности вводимых данных
9	Опишите функцию умножения двух чисел с плавающей запятой, обработайте ошибку переполнения сверху (overflow)
10	Опишите функцию вычисления логарифма, обработайте ошибку вычисления логарифма отрицательного числа
11	Опишите функцию деления двух чисел с плавающей запятой, обработайте ошибку деления на ноль (zero division)
12	Опишите функцию вычисления квадратного корня, обработайте ошибку вычисления корня из отрицательного числа
13	Опишите функцию анализа времени, обработайте ошибку задания времени в неверном формате (допустимый формат: чч:мм:сс) и ошибку корректности вводимых данных
14	Опишите функцию умножения двух целых, обработайте ошибку переполнения сверху (overflow)
15	Опишите функцию деления двух целых, обработайте ошибку переполнения снизу (underflow)

Лабораторная работа 20. Стандартная библиотека шаблонов. Строковый класс

Основные определения STL

Библиотека стандартных шаблонов (Standard Template Library, STL) – наиболее совершенный инструмент языка программирования C++.

Ядро библиотеки стандартных шаблонов состоит из четырех основных элементов:

- контейнеров,
- итераторов,
- алгоритмов,
- функциональных объектов.

Контейнеры – это объекты, предназначенные для хранения других объектов. Контейнеры бывают различных типов: массивы, очереди, списки и т. д.

Алгоритмы – выполняют операции над содержимым контейнеров. Существуют алгоритмы для инициализации, сортировки, поиска или замены содержимого контейнеров.

Итераторы – это объекты, которые по отношению к контейнерам играют роль указателей. Они позволяют получать доступ к содержимому контейнера так, как указатель позволяет получить доступ к элементу массива.

Функциональные объекты – это объекты, действующие как функции. Они могут быть объектами класса или указателями на функции, включающими в себя имя функции, поскольку именно оно работает как указатель.

В настоящее время STL включает в себя 11 структур данных контейнеров, реализованных в виде шаблонов классов. Контейнеры, определенные в STL, представлены в табл. 20.1.

Таблица 20.1. Контейнеры, определенные в STL

Контейнер	*Описание*	*Заголовок*
bitset	Множество битов	<bitset>
deque	Двунаправленный список	<deque>
list	Линейный список	<list>
map	Ассоциативный список для хранения пар ключ/значение, где с каждым ключом связано только одно значение	<map>
multimap	Ассоциативный список для хранения пар ключ/значение, где с каждым ключом связано два или более значений	<map>
multiset	Множество, в котором каждый элемент не обязательно уникален	<set>

Контейнер	*Описание*	*Заголовок*
priority_queue	Очередь с приоритетом	<queue>
queue	Очередь	<queue>
set	Множество, в котором каждый элемент уникален	<set>
stack	Стек	<stack>
vector	Динамический массив	<vector>

Методы, общие для всех контейнеров, приведены в табл. 20.2.

Таблица 20.2. Методы, общие для всех контейнеров [7]

Имя	*Назначение*
size()	Возвращает число элементов в контейнере
empty()	Возвращает true, если контейнер пуст
max_size()	Возвращает максимально допустимый размер контейнера
begin()	Возвращает итератор на начало контейнера
end()	Возвращает итератор на конец контейнера
rbegin()	Возвращает реверсивный итератор на конец контейнера (итерации происходят в обратном направлении)
rend()	Возвращает реверсивный итератор на начало контейнера (итерации происходят в обратном направлении)

Алгоритмы предназначены для обработки контейнеров. Каждый контейнер имеет собственный базовый набор операций, но стандартные алгоритмы обеспечивают более широкий спектр действий. Кроме того, они позволяют одновременно работать с двумя контейнерами, имеющими разные типы. Для обеспечения доступа к алгоритмам STL, нужно подключить заголовочный файл <algorithm>.

Основные алгоритмы STL приведены в табл. 20.3.

Таблица 20.3. Основные алгоритмы STL [7]

Алгоритм	*Действие*
find	Возвращает первый элемент с указанным значением
count	Считает количество элементов, имеющих указанное значение
equal	Сравнивает содержимое двух контейнеров и возвращает true, если все соответствующие элементы эквивалентны
search	Ищет последовательность значений в одном контейнере, которая соответствует такой же последовательности в другом
copy	Копирует последовательность значений из одного контейнера в другой или в другое место того же контейнера
swap	Обменивает значения, хранящиеся в разных местах
sort	Сортирует значения в указанном порядке
merge	Комбинирует два сортированных диапазона значений для получения возможно большего диапазона
accumulate	Возвращает сумму элементов в заданном диапазоне

Алгоритм	*Действие*
for_each	Выполняет указанную функцию для каждого элемента контейнера
reverse	Меняет на обратный порядок расположения элементов в контейнере

Векторы

Вектор представляет собой, фактически, динамический массив и предоставляет непосредственный доступ к любому элементу данных, используя операцию индексирования.

Для класса vector определена операция индексирования [], а также операции сравнения ==, !=, <, <=, >, >=.

Наиболее важные функции-элементы класса vector представлены в табл. 20.4.

Таблица 20.4. Функции-элементы класса vector

Функция	*Описание*
size()	Возвращает текущий размер вектора. Поскольку размер вектора может изменяться динамически, важно иметь возможность определять его размер в процессе работы программы, а не во время компиляции
begin()	Возвращает итератор начала вектора, являющийся, фактически, указателем на его начало
end()	Возвращает итератор конца вектора, являющийся, фактически, указателем на его конец
push_back()	Помещает значение в конец вектора. Если необходимо, размер вектора при этом автоматически увеличивается
insert()	Помещает значение в середину вектора
erase()	Удаляет элементы из вектора

Пример 20.1. Основные операции, производимые над векторами.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <conio.h>

using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{ vector<int> vect; // создание вектора 0 длины
  /* создание вектора, состоящего из семерок,
     длина которого равна 5:
     vector<int> vect(5, 7);
  */
  unsigned int i;
  cout << "Length of vector: "
       << vect.size() << endl;   // 0
```

```
  for (i = 0; i < 5; i++)
      vect.push_back(i * i);
  cout << "Length of vector: "
       << vect.size() << endl;   // 5
  for (i = 0; i < vect.size(); i++)
     cout << vect[i] << ' ';
  cout << endl;
  for (i = 0; i < vect.size(); i++)
     vect[i] = vect[i] * vect[i];
  // вывод вектора с помощью итератора:
  vector<int>:: iterator ip = vect.begin();
  while (ip != vect.end())
     { cout << *ip << ' ';
       ip++;
     }
  ip = vect.begin();
  ip += 3; // установили итератор на 4 элемент
// вставить 5 элементов, каждый из которых
// равен 13, начиная с того места, куда указывает
// итератор:
  vect.insert(ip, 5, 13);
  for (i = 0; i < vect.size(); i++)
       cout << vect[i] << ' ';
  cout << endl;
  ip = vect.begin();
  ip += 2; // установили итератор на 3 элемент
  // удалить 3 элемента, начиная с того места,
  // куда указывает итератор
  vect.erase(ip, ip + 3);
  for (i = 0; i < vect.size(); i++)
     cout << vect[i] << ' ';
  cout << endl;
  _getch();
  return 0;
}
```

Использование векторов для хранения объектов аналогично использованию их для хранения данных стандартных типов. Разница заключается в том, что в угловых скобках, определяющих параметризованный тип, указывается тип класс.

Пример 20.2. Использование векторов для хранения объектов.

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <conio.h>
using namespace std;
class MyVect
 { int iVal;
  public:
   MyVect()  { iVal = 0;}
   MyVect ( int n)  { iVal = n;}
   MyVect &operator = (int n)
     { iVal = n;
```

```
        return *this;
      }
    int get_Val() { return iVal; }
     // В случае необходимости можно перегрузить
     // соответствующие  операторы, например:
     // friend bool operator < (MyVect ob1, MyVect ob2);
     // friend bool operator == (MyVect ob1, MyVect ob2);
 };

/*  bool operator < (MyVect ob1, MyVect ob2)
     { return ob1.get_Val() < ob2.get_Val(); }
    bool operator == (MyVect ob1, MyVect ob2)
     { return ob1.get_Val() == ob2.get_Val(); }
*/

int _tmain (int argc, _TCHAR* argv[])
{   vector<MyVect> vect;
    unsigned int i;
    for (i = 0; i < 5; i++) vect.push_back (i * i);
    for (i = 0; i < vect.size(); i++)
        cout << vect[i].get_Val()  << ' ';
    cout << endl;
    for (i = 0; i < vect.size(); i++)
        vect[i] = vect[i].get_Val() * 2;
    for (i = 0; i < vect.size(); i++)
        cout << vect[i].get_Val()  << ' ';
    cout << endl;
    _getch();
    return 0;
}
```

Списки

Списки представлены последовательными наборами двусвязных элементов и поддерживаются классом list.

Для класса list определены операции сравнения ==, !=, <, <=, >, >=.

Наиболее важные функции-элементы класса list представлены в табл. 20.5.

Таблица 20.5. Функции-элементы класса list

Функция	*Описание*
push_back()	Помещает элемент в конец списка
push_front()	Помещает элемент в начало списка
insert()	Помещает элемент в середину списка
merge()	Выполняет слияние двух упорядоченных списков
pop_back()	Удаляет последний элемент списка
pop_front()	Удаляет первый элемент списка

Для любого типа данных, которые необходимо хранить в списке, должен быть определен конструктор по умолчанию.

Пример 20.3. Сортировка списка.

```
#include "stdafx.h"
#include <iostream>
#include <list>
#include <cstdlib>
#include <conio.h>

using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{setlocale (LC_ALL, "Russian");
 list <int> l_int;
 int i;
 for (i = 0; i < 10; i++)
      l_int.push_back (rand() % 100);
 cout << "Исходный список:" << endl;
 list <int>:: iterator ip = l_int.begin();
 while (ip != l_int.end())
{ cout << *ip << ' ';
       ip++;
     }
 cout << endl;
 // сортировка
 l_int.sort();
 cout << "Отсортированный список:" << endl;
 ip = l_int.begin();
 while (ip != l_int.end())
{ cout << *ip << ' ';
       ip++;
     }
 cout << endl;
 _getch();
 return 0;
}
```

Строковый класс

В языке C++ встроенный строковый тип отсутствует, но для более удобной работы со строками был разработан строковый класс.

Для работы со строковым классом необходимо подключить заголовочный файл <string>. Строковый класс является классом-контейнером, поэтому поддерживает все алгоритмы библиотеки стандартных шаблонов.

Методы size() и length() возвращают количество символов в строке.

Поиск в строке. В классе stringсуществует множество методов поиска подстрок и отдельных символов в строке табл. 20.6.

Таблица 20.6. Методы поиска в строке

Метод	*Описание*
find	Находит первое вхождение подстроки, начиная с начала строки
rfind	Находит первое вхождение подстроки, начиная с конца строки
find_first_of	Находит индекс первого вхождения любого из символов подстроки
find_last_of	Находит индекс последнего вхождения любого из символов подстроки
find_first_not_of	Находит индекс первого из символов, не входящих в подстроку
find_last_not _of	Находит индекс последнего из символов, не входящих в подстроку

Пример 20.4. Методы поиска в строке.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  string ABC = "abcdefghijklmnopqrstuvwxyz";
   string substr = "klmnop";
       // поиск строковых литералов
   cout << ABC.find ("rstuv") << endl;    // 17
   cout << ABC.find ("lkji") << endl;     // -
       // поиск строковых объектов
   cout << ABC.find (substr) << endl;     // 10
       // с указанием начальной позиции поиска
   cout << ABC.find (substr, 5) << endl;  // 10
       // поиск символов из представленного набора
   cout << ABC.find_first_of ("aeiou")
        << endl;     // 0
   cout << ABC.find_last_not_of ("aeiou")
        << endl;  // 25
   _getch();
   return 0;
}
```

Модификация строки. Модификация строки может заключаться в удалении, вставке или замене ее фрагмента. Эти действия выполняются с помощью методов, которые представлены в табл. 20.7.

Таблица 20.7. Методы модификации строк

Метод	*Описание*
append	Добавляет подстроку в конец строки
erase	Удаляет фрагмент из строки
insert	Вставляет подстроку в указанное место строки
replace	Заменяет одну подстроку на другую

Пример 20.5. Методы модификации строк.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  setlocale (LC_ALL,"Russian");
   string s1 = "Мишка очень любит мед!";
   string s2 = "пчел!";
   string s3 = " не ";
   cout << s1 << endl;
// заменяем слово "мед" на "пчел"
   s1.replace (18, 4, s2);
// добавляем слово "не" после слов "Мишка очень"
   s1.insert (11, s3);   cout << s1 << endl;
   _getch();
   return 0;
}
```

Сравнение строк. Объекты класса string можно сравнивать, используя перегруженные операции или метод compare(). Равными считаются полностью идентичные строки; меньшей считается та, что стоит раньше в алфавитном порядке.

Пример 20.6. Сравнение строк с помощью перегруженных операций.

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <conio.h>
using namespace std;

int _tmain (int argc, _TCHAR* argv[])
{  setlocale (LC_ALL,"Russian");
   string str1, str2;
   cout << "Введите строку str1: ";
   cin >> str1;
   cout << "Введите строку str2: ";
   cin >> str2;
   if (str1 == str2) cout << "Строки равны" << endl;
   if (str1 != str2) cout << "Строки не равны: ";
   if (str1 > str2) cout << "строка str1 больше строки str2"
                         << endl;
   if (str1 < str2) cout << "строка str1 меньше строки str2"
                         << endl;
```

```
    _getch();
    return 0;
}
```

Перегруженные операции сравнения вызывают метод compare() шаблона basic_string, поддерживающий дополнительные параметры, которые можно использовать для сравнения подстрок.

Контрольные вопросы

1. Из чего состоит ядро библиотеки стандартных шаблонов?
2. Дайте определения контейнеров, алгоритмов, итераторов, функциональных объектов.
3. Что представляет собой контейнерный класс?
4. Перечислите методы, общие для всех контейнеров.
5. Как осуществляется ввод и вывод строк?
6. Перечислите: а) методы поиска подстрок и отдельных символов в строке; б) методы модификации строки.

Задания

Таблица 20.8. Варианты для работы с STL

Вариант	*Задание*
1	В векторе целого типа заменить максимальный элемент суммой предыдущих элементов, минимальный – суммой последующих элементов
2	В конец вектора целого типа дописать четные элементы этого вектора
3	В начало вектора целого типа дописать нечетные элементы этого вектора
4	В середину вектора целого типа поместить элементы этого вектора, кратные пяти
5	В начало вектора целого типа дописать его минимальное значение, в середину – максимальное
6	В начало вектора целого типа записать элементы, являющиеся делителями максимального элемента этого файла
7	В середину вектора целого типа записать элементы этого файла, которые меньше числа, введенного с клавиатуры
8	Дан вектор с положительными и отрицательными целыми числами. Записать в другой файл сначала отрицательные элементы, а затем положительные

Вариант	*Задание*
9	Сформировать динамический список, считая, что длина списка (количество элементов) задана. Описать функцию, которая удаляет из списка за каждым вхождением элемента *E* (значение которого введено с клавиатуры), один элемент, если такой есть и он отличен от *E*
10	Сформировать динамический список, считая, что длина списка (количество элементов) задана. Описать функцию, которая формирует список *M*1 – копию списка *M* и список *M*2, представляющий собой «перевернутый» список *M*
11	Сформировать динамический список, считая, что длина списка (количество элементов) задана. Описать функцию, которая включает в упорядоченный по убыванию список новое значение, введенное с клавиатуры, таким образом, чтобы не нарушать упорядоченность
12	Сформировать динамический список, считая, что длина списка (количество элементов) задана. Описать функцию, которая объединяет два упорядоченных по убыванию списка в один упорядоченный по невозрастанию список
13	Найти сумму элементов вектора целого типа, находящихся между минимальным и максимальным элементами, включая и сами эти числа
14	Сформировать динамический список, считая, что длина списка (количество элементов) задана. В составе программы описать функцию, которая в списке из каждой группы подряд идущих одинаковых элементов оставляет только один элемент

Таблица 20.9. Варианты для работы со строковым классом

Вариант	*Задание*
1	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «and» запятую. Определите сколько в строке симметричных слов
2	Дана строка слов, разделенных пробелами. Сформируйте новую строку, вставив перед каждым вхождением слова «по» запятую. Подсчитайте количество подстрок между запятыми. Определите сколько слов в строке, у которой первая буква содержится в слове более одного раза
3	Дана строка слов. Сформируйте новую строку, удалив пробелы, с которых может начинаться строка, а каждую внутреннюю группу пробелов замените одним пробелом. Подсчитайте количество слов в данной строке и количество слов, у которых первая и последняя буквы совпадают

Вариант	*Задание*
4	Дана строка слов, разделенных пробелами. Определите количество слов, которые встречаются более одного раза. Сформируйте строку из неповторяющихся слов
5	Дана строка слов, разделенных пробелами. Сформируйте строку из неповторяющихся слов, расположив их в алфавитном порядке
6	Дана строка слов, разделенных пробелами, запятыми, точками. Сформируйте новую строку из пяти самых длинных слов. Определите количество слов, начинающихся первой буквой алфавита (русского или латинского)
7	Дана строка символов и некоторый символ *n*. Сформируйте новую строку, вставив после каждого вхождения символа *n* пробел. Подсчитайте количество различных слов в образовавшейся строке
8	Дана строка символов и некоторый символ *n*. Сформируйте новую строку, вставив после каждого вхождения символа *n* запятую. Определите самое большое слово в строке
9	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество подстрок, заключенных между запятыми, в строке. Определите длину самого короткого слова
10	Дана строка слов, разделенных пробелами и запятыми. Подсчитайте количество слов в строке и сформируйте новую строку из самых длинных слов подстрок, заключенных между запятыми
11	Дана строка символов, представляющих собой арифметическое выражение. Порядок операций определен слева направо. Подсчитайте результат данного выражения
12	Дана строка слов, разделенных пробелами. Сформируйте новую строку, заменив каждую группу внутренних пробелов одним пробелом. Оставьте в строке только первые вхождения слов. Определите самое короткое слово
13	Дана строка слов, разделенных пробелами, запятыми, точками. Сформируйте новую строку из трех самых коротких слов. Определите количество слов, заканчивающихся последней буквой алфавита (русского или латинского)
14	Дана строка слов. Сформируйте новую строку, вставив перед каждым из слов «а» и «но» запятую. Подсчитайте количество подстрок, разделенных запятыми. Сформируйте строку из слов, с которых начинаются подстроки
15	Дана строка слов. Сформируйте новую строку, вставив перед каждым из слов «а» и «но» запятую. Определите самую короткую подстроку и слово, с которого она начинается

Лабораторная работа 21. Начальные сведения об интегрированной среде разработки программ C++Builder 6

Основные компоненты интегрированной среды разработки

Интегрированная среда разработки (ИСР) программ включает в себя (рис. 21.1):

- главное меню;
- левую инструментальную панель (быстрые кнопки);
- правую инструментальную панель (палитру компонентов библиотеки визуальных компонентов);
- дополнительную инструментальную панель (правее главного меню) выбора конфигураций окна ИСР;
- дерево объектов (Object Tree View);
- инспектор объектов (Object Inspector);
- окно формы.

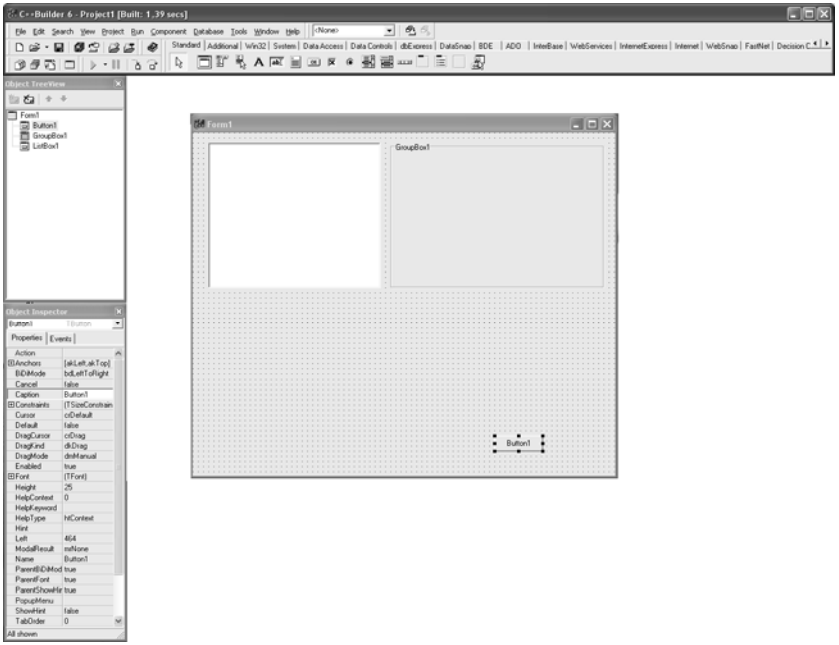


Рис. 21.1. Основное окно ИСР C++Builder6

Дерево объектов отображает иерархическую связь визуальных и невизуальных компонентов и объектов создаваемого приложения.

Инспектор объектов – это основной инструмент, с помощью которого задаются свойства компонентов и обработчики событий.

Рассмотрим основные компоненты главного меню.

File (файл) – создает новый проект, новую форму, открывает существующий проект или форму, сохраняет проекты или формы в файлах с заданными именами.

Edit (редактирование) – позволяет выполнять обычное редактирование, операции с буфером обмена, выравнивание групп, размещенных на форме компонентов по размеру и местоположению.

Search (поиск) – осуществляет поиск и контекстную замену.

View (просмотр) – осуществляет вызов на экран различных окон.

Project (проект) – добавляет и удаляет из проекта формы, компилирует проект (без запуска на выполнение), задает опции проекта.

Run (выполнение) – выполняет проект в различных режимах (нормальном, пошаговом, с точками останова, с возможностью просмотра значений переменных и т. д.

Component (компонент) – создает и устанавливает новые компоненты, конфигурирует палитру инструментов.

Database (база данных) – позволяет использовать инструментарий для работы с базами данных.

Палитра инструментов – это набор пиктограмм библиотеки визуальных компонентов (Visual Component Library – VCL). Пиктограммы сгруппированы в страницы, названия которых видны в верхней части палитры компонентов (рис. 21.2).

Рис. 21.2. Палитра инструментов

Для переноса компонентов на форму, нужно открыть соответствующую страницу библиотеки и указать курсором мыши необходимый компонент. Двойной щелчок мыши разместит выбранный объект в центре формы, после чего его можно переместить с помощью курсора мыши в нужное место формы, а также изменить его размеры.

Имя компонента, соответствующего данной пиктограмме, можно узнать из ярлычка, появляющегося, если задержать курсор мыши на этой пиктограмме. Если выбрать компонент и нажать F1, будет выдана информация об этом компоненте. Однако в языке C++Builder есть небольшое несоответствие имен. Имя на ярлычке выглядит, например, Button или MainMenu, а имя класса, соответствующего такому компоненту, бу-

дет соответственно Tbutton и TMainMenu, так как все имена классов в языке C++Builder начинается с T.

В языке C++ Builder форма является одним из базовых компонентов, поскольку именно на ней размещаются все остальные компоненты разрабатываемого приложения.

Другой, не менее важной частью ИСР C++Builder, является окно редактора кода (рис. 21.3). Это полноценный программный редактор, настраиваемый на различный стиль работы, в котором применяется выделение цветом и шрифтом синтаксических элементов программного кода.

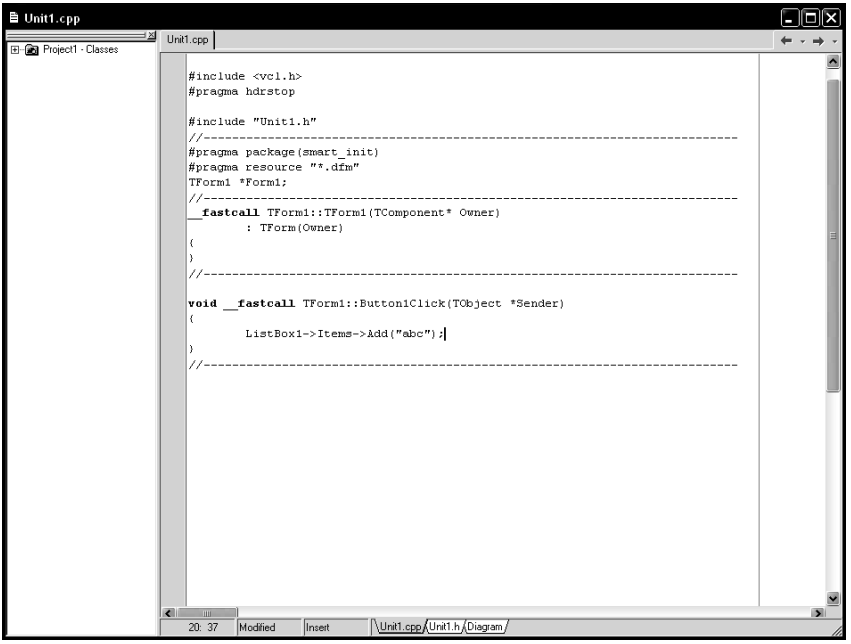


Рис. 21.3. Окно редактора кода

В заголовке окна редактора кода размещается имя файла разрабатываемого приложения. Если приложение многомодульное, то закладки в верхней части окна позволяют быстро переключаться с одного модуля на другой.

В момент размещения элемента на форме в окне редактирования кода появляется заготовка функции, описывающей данный элемент. В заготовку остается вписать код, соответствующий алгоритму решаемой задачи. Однако это не единственная возможность работать с окном редактирования кода, поскольку оно позволяет загрузить и редактировать любую программу на языке C++, а также любые текстовые файлы и даже файлы HTML.

Инспектор объектов (Object Inspector) позволяет изменить свойства объектов C++Builder и управлять событиями, на которые реагирует объект. Окно инспектора объектов (рис. 21.4) имеет две страницы.

Страница свойств (Properties) инспектора объектов показывает свойства выделенного в данный момент объекта. Свойства можно не только просматривать, но и изменять.

Страница событий (Events) содержит все события, на которые может реагировать выбранный объект. При необходимости написания обработчика событий достаточно сделать двойной щелчок в пустом окне справа от имени события. После этого в окне редактора кода автоматически появится заготовка, которую останется только заполнить.

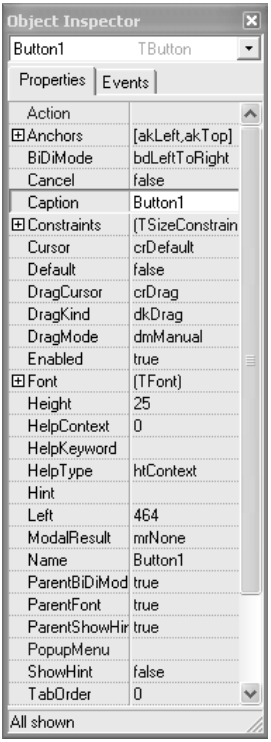


Рис. 21.4. Окно инспектора объектов

Пример 21.1. Создать приложение, в котором при щелчке мыши на кнопке появляется приветствие.

1. Запустить C++Builder.
2. Командой File/New открыть новое приложение, выбрав в открывшемся каскадном меню раздел Application.
3. Перенести на открывшуюся пустую форму кнопку TButton со страницы Standard палитры компонентов. Для этого выделяют пиктограмму и щелкают мышью в нужном месте формы. На форме появляется кнопка с именем по умолчанию Button1.
4. Выделить на форме кнопку с именем Button1, присвоенным по умолчанию. Перейти в инспектор объектов и изменить ее свойство Caption на значение «Начало».
5. Перенести на форму со страницы Standard палитры компонентов метку Label1. Здесь будет появляться текст при нажатии кнопки «Начало».
6. Для задания стиля шрифта сообщения в метке Label1 в окне инспектора объектов используют свойстваFont (шрифт), Style (стиль) и т. д.
7. Убрать текст в свойстве Caption метки Label1, чтобы до нажатия кнопки «Начало» экран оставался пустым.

8. Поскольку текст на экране должен появляться при нажатии кнопки «Начало», следует написать обработчик события OnClick. Заготовку для этого обработчика можно получить двумя способами:
   - выделить кнопку Button1 на форме и перейти в инспектор объектов, открыть в нем страницу событий (Events), найти событие кнопки OnClick, сделать двойной щелчок в окне справа от имени этого события;
   - сделать двойной щелчок на компоненте Button1 на форме. После этого произойдет переход в окно редактора кода, в котором окажется заготовка функции обработчика событий:

```
void _ _fastcall TForm1::Button1Click (TObject *Sender)
{
}
```

Между фигурными скобками следует написать оператор:

```
Label1->Caption = "Мы начинаем осваивать C++Builder.”;
```

Пример 21.2. Обработчик щелчка левой кнопкой мыши на форме.

```
void _ _fastcall TForm1::FormClick (TObject *Sender)
{ MessageDlg ( "A study of events", mtInformation,
TMsgDlgButtons()<<mbOK, 0);
}
```

Все шаги для создания этой программы аналогичны приведенным в примере 21.1. При запуске этой программы каждый раз при щелчке левой кнопкой мыши на форме появляется диалоговое окно вида, показанного на рис. 21.5.

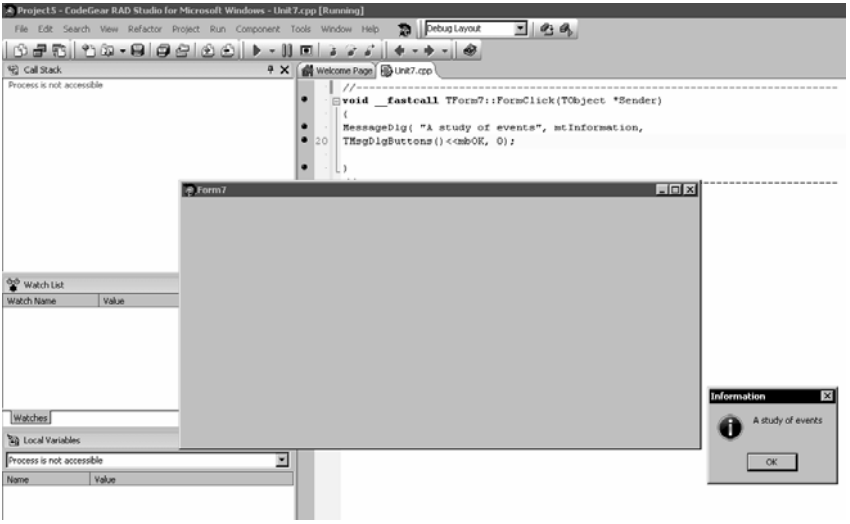


Рис. 21.5. Окно формы

При возникновении определенного события операционная система сопровождает передачу сообщения о событии несколькими служебными информационными битами. Например, при нажатии кнопки мыши программа получает данные о том, где произошло событие и какая именно кнопка мыши была нажата. Доступ к этой информации можно получить в окне Object Inspector (вкладка Events).

Пример 21.3. Обработка событий OnMouseDown.

```
void __fastcall TForm10::FormMouseDown (TObject *Sender,
   TMouseButton Button,
     TShiftState Shift, int X, int Y)
{  if (Shift.Contains (ssRight))
   { Canvas->Brush->Style = bsClear;
               Canvas->TextOut (X, Y, "MouseRightKey");
   }
}
```

Каждый раз при нажатии правой кнопки мыши на экране появляется надпись MouseRightKey. При этом устанавливается стиль кисти bsClear, который делает фон прозрачным.

Функция Canvas→Text распечатывает текст, начиная с позиции, заданной перемещенными X и Y, где X и Y – координаты курсора мыши в момент нажатия клавиши.

Пример 21.4. Ввод чисел в двух разных полях и умножение введенных данных.

Открыть новое приложение и разместить в нем следующие компоненты: два окна редактирования (LabeledEdit или Edit) со страницы Additional, одну панель Panel, одну кнопку Button, одну метку Label (для подписи) со страницы Standard и одну метку (три метки в случае окна редактирования Edit).

Изменить надписи в метках компонентов на «Множимое», «Множитель», «Произведение», а свойство Caption кнопки Button на «Посчитаем!».

Обработчик нажатия кнопки имеет вид:

```
void __fastcall TForm13::Button1Click (TObject *Sender)
{  Panel1->Caption =  LabeledEdit1->Text+"*"+
    LabeledEdit2->Text+"="+
    FloatToStr (StrToFloat (LabeledEdit1->Text)*
    StrToFloat (LabeledEdit2->Text));
}
```

При использовании компонента Edit LabeledEdit нужно заменить на Edit.

Графические возможности C++ Builder

Многие компоненты в C++ Builder имеют свойство Canvas, которое представляет собой графические возможности компонента. Canvas имеют формы, компоненты Image, PaintBox, BitMap и др.

Каждая точка графического поля (канвы) имеет координаты X и Y, при этом X возрастает при перемещении слева направо, Y – сверху вниз. Здесь координаты измеряются в пикселях. *Пиксель* – наименьший графический объект на экране. Кроме координат пиксель имеет еще одно свойство – цвет, для описания которого используется тип TColor. Цвет можно задать с помощью предопределенных констант: clRed – красный, clBlack – черный, clGreen – зеленый и т. д. или с помощью так называемого RGB-кода:

$00FF0000 – синий,
$0000FF00 – зеленый,
$000000FF – красный,
$00000000 – черный,
$00FFFFFF – белый.

Рисовать можно двумя способами: по пикселям и с помощью пера. При рисовании по пикселам рисунок получается дискретным, т. е. состоящим из отдельных точек, при рисовании пером – рисунок непрерывный.

При рисовании по пикселям используется свойство канвы Pixels, которое представляет собой матрицу Canvas->Pixels[int X][int Y], каждая точка которой определяет цвет пикселя на канве.

Пример 21.5. Рисование по пикселям.

```
float X, Y;      //координаты точек изображаемой функции
int PX, PY;      //координаты соответствующих им пикселей
float Xmax, Xmin, Ymax, Ymin;  //координаты максимальных и
// минимальных значений X и Y для данного отрезка функции,
// необходимые для масштабирования графика и его
// равномерного расположения на экране;
// F(X) – функция, график которой мы хотим построить
for (PX = 0; PX <= Image->Width; PX++)
      { X = Xmin+PX*(Xmax – Xmin) / Image1->Width;
         Y = F(X);
       // PY = Image1->Height- (Y – Ymin)*Image1->Height /
       //                                  (Ymax – Ymin);
       // Будем рисовать красным цветом:
          Image1->Canvas->Pixels[PX][PY] = clRed;
      }
```

Здесь Image1->Height и Image1->Width – высота и ширина экрана (в пикселях).

Пример 21.6. Рисование функции y = cos (x) на отрезке $x \in [0, 2\pi]$.

Для данной функции X изменяется на отрезке от 0 до 2π, Y – от -1 до 1, поэтому Ymin = -1, Ymax = 1.

Создаем форму и помещаем на нее компонент Image и кнопку с надписью y = cos (x).

Обработчик события OnDblClick (двойной щелчок левой кнопкой мыши) будет иметь вид:

```
#define PI 3.14159   //или const float Pi = 3.14159;
float X, Y, Xmax=2*PI, Xmin=0, Ymax=1,Ymin=-1;
int PX, PY;
for (PX=0; PX<=Image1->Width; PX++)
    { X=Xmin+PX* (Xmax-Xmin) / Image1->Width;
      Y=cos (X);
      PY=Image1->Height- (Y+Ymax)* Image1->Height /
                                                (Ymax-Ymin);
      Image1->Canvas->Pixels[PX][PY]=clRed;
    }
```

Если нужен непрерывный рисунок, используют свойство Pen – перо, которое в свою очередь, тоже имеет ряд свойств: Color – цвет, Width – ширина линии в пикселях (по умолчанию – 1), Style – вид линии.

Свойство канвы PenPos определяет текущую позицию пера. Перемещение пера без прорисовки линии осуществляется методом канвы MoveTo (X, Y). Метод LineTo (X, Y) рисует линию от текущего положения пера к точке с координатами (X, Y).

Пример 21.7. Рисование графикаиз примера 13.6 с помощью пера.

```
#define PI 3.14159    //или const float Pi = 3.14159;
float X, Y, Xmax=2*PI, Xmin=0, Ymax=1,Ymin=-1;
int PX, PY;
for (PX=0; PX<=Image1->Width; PX++)
    { X=Xmin+PX* (Xmax-Xmin) / Image1->Width;
      Y=cos (X);
      PY=Image1->Height- (Y+Ymax)* Image1->Height /
                                              (Ymax-Ymin);
      Image1->Canvas->Pixels[PX][PY]=clWhite;
      Image1->Canvas->LineTo (PX, PY);
    }
```

С помощью пера можно рисовать не только линии, но и фигуры (табл. 21.1).

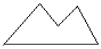
Таблица 21.1. Методы объекта Canvas

Arc	*Дуга*
Chord	Дуга + хорда
Ellipse	Эллипс или окружность
Pie	Сектор
Polygon	Замкнутый полигон
Polyline	Разомкнутый полигон
Rectangle	Прямоугольник или квадрат
RoundRect	Прямоугольник или квадрат со скругленными углами

Пример 21.8. Изобразить бирюзовый эллипс, ограниченный желтым контуром

```
void _ _fastcall TForm::FormPaint (TObject*Sender)
```

```
    { Canvas->Pen->Color=clYellow;        //цвет контура
       Canvas->Brush->Color=clCyan;       //цвет заливки
       Canvas->Ellipse (15, 15, 60, 60);  //эллипс
    }
```

Пример 21.9. Изобразить Polygon  и Polyline 

```
{ TPoint points[5];   //задаем массив точек многоугольника
         points[0]=Point (30, 150);
         points[1]=Point (40, 130);
         points[2]=Point (50, 140);
         points[3]=Point (60, 130);
         points[4]=Point (70, 150);
   Image1->Canvas->Polygon (points, 4);
   Image1->Canvas->TextOut (30, 170, "Polygon");
   points[0].x+=100;
   points[1].x+=100;
   points[2].x+=100;
   points[3].x+=100;
   points[4].x+=100;
   Image1->Canvas->Polyline (points, 4);
   Image1->Canvas->TextOut (130, 170, "Polyline");
}
```

Контрольные вопросы

1. Каковы основные элементы ИСР и каково их назначение?
2. Что такое визуальные и невизуальные компоненты формы?
3. Каким образом можно задать свойства компонента Edit?
4. Каким образом элемент формы настроить на выполнение определенного действия?
5. Что такое Инспектор объектов и для чего он служит?
6. Каким образом можно отформатировать текст компонента Label?
7. Какие существуют способы рисования графиков и в чем их особенность?

Задания

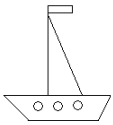
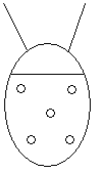
Таблица 21.2. Варианты заданий для работы с пикселями

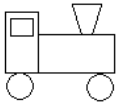
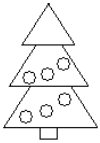
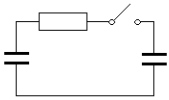
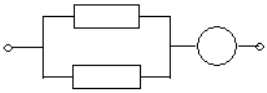
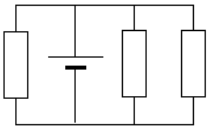
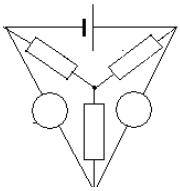
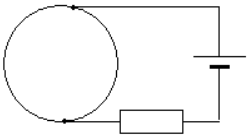
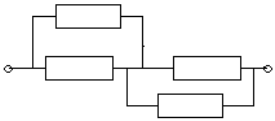
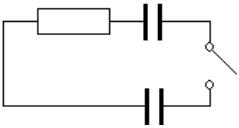
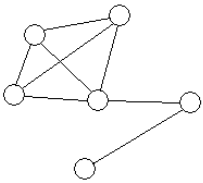
Вариант	*Задание*
1	На отрезке $x \in [-5, 5]$ нарисовать график функции $y = x^3 + 6x^2 - 20$
2	На отрезке $x \in [-2, 2]$ нарисовать график функции $y = 0.25 + \sin x - x$
3	На отрезке $x \in [-3, 3]$ нарисовать график функции $y = x - \sin 5x$

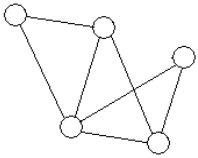
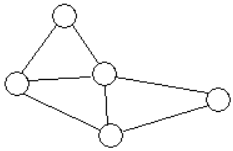
Вариант	*Задание*
4	На отрезке $x \in [0.6, 1.4]$ нарисовать график функции $y = x^3 + x^2 - 3$
5	На отрезке $x \in [-2, 2]$ нарисовать график функции $y = x^3 + 12x - 2$
6	На отрезке $x \in [-1, 1]$ нарисовать график функции $y = x^4 + 2x^3 - x - 1$
7	На отрезке $x \in [-1, 1]$ нарисовать график функции $y = 2x^3 + 4x - 1$
8	На отрезке $x \in [0, 3]$ нарисовать график функции $y = x + \ln(x + 0.5) - 0.5$
9	На отрезке $x \in [-1, 1]$ нарисовать график функции $y = x \sin^2 x + 1$
10	На отрезке $x \in [-3, 3]$ нарисовать график функции $y = 2x^3 + 5x + 4$
11	На отрезке $x \in [-1, 1]$ нарисовать график функции $y = x - \cos 2x$
12	На отрезке $x \in [0, 5]$ нарисовать график функции $y = (x - 1)^2 / (x + 1)$
13	На отрезке $x \in [0, 3]$ нарисовать график функции $y = \sqrt[3]{x(x-2)^2}$
14	На отрезке $x \in [-1, 1]$ нарисовать график функции $y = \sin 2x - 2x + 1$
15	На отрезке $x \in [-2, 2]$ нарисовать график функции $y = x^4 - x - 5$

Варианты работы с библиотечными элементами и рисования пером

Выполнить рисунок и раскрасить его

Вариант 1	Вариант 2
	

Вариант 3 	Вариант 4 
Вариант 5 	Вариант 6 
Вариант 7 	Вариант 8 
Вариант 9 	Вариант 10 
Вариант 11 	Вариант 12 
Вариант 13 	Вариант 14 

Вариант 15 	Вариант 16 

Литература

Ашарина И. В., Березовский А. В., Крупская Ж. Ф., Соколова Н. Ю. Язык С++ и ООП в языке С++: Лабораторный практикум. М.: РИО МИЭТ, 2005.

Керниган Б., Ритчи Д. Язык программирования Си. М.: Финансы и статистика, 1992.

Сван Т. Освоение Borland C++ 4.5: Практический курс. Киев: Диалектика, 1996.

Сван Т. Освоение Borland C++ 4.5: Энциклопедия функций. Киев: Диалектика, 1996.

Уэйт М., Прата С., Мартин Д. Язык Си: Руководство для начинающих. М.: Мир, 1998.

Прата С. Язык программирования Си: Лекции и упражнения. М.: Вильямс, 2006.

Прата С. Язык программирования Си++: Лекции и упражнения. М.: Вильямс, 2007.

Лафоре Р. Объектно-ориентированное программирование в языке С++. М.: ПИТЕР, 2004.

Содержание