Министерство образования и науки Российской Федерации

Государственное образовательное учреждение
высшего профессионального образования

«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Л. А. АРТЮШИНА, Ю. М. МОНАХОВ, А. А. ВОРОНИН

ИНФОРМАТИКА. ПРОГРАММИРОВАНИЕ: C++

Учебное пособие

В двух частях

Часть 1. Введение в язык C ++

Владимир 2011

УДК 004.056
ББК 32.97
И74

Рецензенты:
Кандидат технических наук, профессор,
зав. кафедрой информатики и вычислительной техники
Владимирского государственного гуманитарного университета
Ю.А. Медведев

Кандидат технических наук, доцент кафедры управления и информатики
в технических и экономических системах
Владимирского государственного университета
С.И. Лиходеев

Печатается по решению редакционного совета
Владимирского государственного университета

Информатика. Программирование: C++ : учеб. пособие. В 2 ч.
И74 Ч. 1. Введение в язык С ++ / Л. А. Артюшина, Ю. М. Монахов, А. А. Воронин ; Владим. гос. ун-т. – Владимир : Изд-во Владим. гос. ун-та, 2011. – 132 с.
ISBN 978-5-9984-0124-4

Ориентировано на изучение основных, базовых конструкций C++ и обретение навыков создания программ на этом языке. Содержит темы, посвященные отдельным конструкциям изучаемого языка, необходимый справочный материал, тексты программ, ответы, указания и решения к заданиям в помощь преподавателю и студентам. К каждому занятию предлагаются практические задания для аудиторной и домашней работы.

Предназначено для проведения практических и лабораторных работ для студентов первого курса специальностей 210302 – радиотехника, 210301 – радиофизика и электроника, 210202 – проектирование и технология электронно-вычислительных средств в рамках дисциплины «Информатика».

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС 3-го поколения.

Ил. 7. Табл. 14. Библиогр.: 9 назв.

УДК 004.056
ББК 32.97
ISBN 978-5-9984-0124-4

Введение

Данное издание предназначено преподавателям информатики вузов и содержит материалы к занятиям по изучению одного из наиболее популярных и перспективных языков программирования - C++. Пособие ориентировано на изучение основных, базовых конструкций C++ и обретение навыков создания программ на этом языке.

В последнее десятилетие в сфере программного обеспечения произошли значительные изменения, поэтому еще одна задача данного учебного пособия состоит в изложении концепции языка C++ в контексте развития программного обеспечения. Пособие представляет собой первую часть курса программирования на языке C++, поэтому в нем изложены основы концепции.

Организация занятий с использованием предложенного материала предполагает знание студентами основ информатики на уровне школьной программы, а также наличие опыта работы с персональным компьютером в операционной системе Windows.

Пособие ориентировано на использование программного пакета Borland CPP версии 5.02 или выше и в качестве введения в практический курс содержит описание полного цикла создания консольного приложения с использованием этой системы. В то же время представленный здесь материал может быть легко адаптирован для использования вместе с другими версиями компиляторов, необходимыми для написания программ.

Пособие состоит из одиннадцати занятий, посвященных отдельным конструкциям изучаемого языка. К каждой изучаемой теме предлагаются практические задания для аудиторной и домашней работы, выполняя которые студенты закрепят теоретические знания на практике. Задания отобраны авторами из сборников задач по программированию, перечисленных в списке литературы.

Пособие также содержит необходимый справочный материал и тексты программ, ответы, методические указания и решения к заданиям в помощь преподавателям, а также образец выполнения отчета по практическим работам (Приложения).

Занятие 1. ВВЕДЕНИЕ В ПРАКТИЧЕСКИЙ КУРС

Краткая историческая справка

Язык C++ развился из C, который был создан на основе двух предшествующих языков - BCPL и B. Язык BCPL был создан в 1967 году Мартином Ричардом как язык для написания компиляторов и программного обеспечения операционных систем. Кен Томпсон предусмотрел много возможностей в своем языке B - дубликате BCPL - и использовал B для создания более ранних версий операционной системы *UNIX* в *Bell Laboratories* в 1970 году на компьютере DEC PDP-7.

Язык C был развит из B Денисом Ритчи и первоначально реализован в 1972 году. Он использует многие важные концепции BCPL и B, а также добавляет типы данных и другие свойства. Первоначально C приобрел широкую известность как язык разработки операционной системы UNIX. Сегодня фактически все новые операционные системы написаны на C или на C++. Он не зависим от аппаратных средств. При тщательной разработке на C можно написать мобильные программы, переносимые на большинство компьютеров.

В конце 1970-х годов C развился в то, что теперь относят к «традиционному C», «классическому C» или «C Кернигана и Ритчи».

Широкое распространение C на различных компьютерах (аппаратных платформах) привело ко многим вариациям языка. Некоторые из них были похожи, но несовместимы друг с другом. Это было серьезной проблемой для разработчиков программ, нуждавшихся в написании совместимых программ. Стало ясно, что необходима стандартная версия C. В 1983 году при Американском национальном комитете стандартов (*ANSI*) в области вычислительной техники и обработки информации был создан технический комитет, основной целью которого стало обеспечение недвусмысленного и машинно-независимого определения языка. В 1989 году стандарт был утвержден. *ANSI* скооперировался с Международной организацией стандартов (*ISO*), чтобы стандартизировать C в мировом масштабе. Совместный стандарт был опубликован в 1990 году и назван ISO/IEC 9899:1990.

C++ - расширение C - был разработан Бьерном Страуступом в начале 1980-х годов в *Bell Laboratories*. C++ обеспечивает ряд свойств, которые «приводят в порядок» язык C, но, что более важно,

он обеспечивает возможность объектно-ориентированного программирования. Это явилось революционной идеей в мире программного обеспечения. Объекты - это эффективные повторно используемые компоненты программного обеспечения, моделирующие элементы реального мира. Объектно-ориентированные программы легче понимать, корректировать и модифицировать.

C++ – гибридный язык, он предоставляет возможность программировать и в стиле C, и в объектно-ориентированном стиле, и в обоих стилях сразу. С середины 1990-х годов C++ становится доминирующим системно-образующим языком.

Программа на языке C++ представляет собой файл с расширением CPP (*.*cpp*). Процесс создания этого файла называют *кодированием*; он, как правило, выполняется с помощью специального редактора кода (*code editor*).

По завершении редактирования исходный код программы необходимо перевести на машинный язык. Этот процесс называют компиляцией, он производится компилятором языка (*compiler*). Результат этой стадии - объектный файл с расширением OBJ (*.*obj*).

Завершает разработку программы фаза компоновки, в результате которой создается исполняемый файл с расширением EXE (*.*exe*), готовый к работе.

Фаза компиляции сопровождается проверкой синтаксиса программы, все найденные ошибки сообщаются пользователю.

Ошибки *Errors* означают нарушения синтаксиса языка C++, которые делают невозможной дальнейшую компоновку программы. Все ошибки должны быть устранены, для чего можно воспользоваться сообщениями компилятора, в которых указывается номер строки и краткое описание ошибки.

Предупреждения *Warnings* означают подозрительные конструкции, которые могут функционировать неправильно - не так, как это задумано программистом. Наличие предупреждений не приводит к остановке процессов компиляции и компоновки. Однако к ним также следует относиться внимательно, так как в большинстве случаев они следствие логических ошибок в программе, которые трудно поддаются обнаружению.

Современные системы программирования часто объединяют в себе все компоненты, необходимые для создания исполняемого файла,

то есть выполняют одновременно функции редактора кода, компилятора, компоновщика, а также некоторые другие - библиотекаря (*librarian*), отладчика (*debugger*), профайлера (*profiler*) и т. д. Такие системы называют интегрированными средами разработки (IDE, integrated development environment). Программный пакет *Borland CPP* 5.02 – пример такой *IDE*, которую мы будем использовать в рамках настоящего курса.

Правила программирования

Это занятие посвящено рассмотрению универсальных правил, по которым пишутся программы, форматированию и правильной записи программ.

1. *Имена переменных.* Именам переменных необходимо давать *содержательные имена*, отражающие суть тех данных, для хранения которых они предназначены. Исключением могут быть только переменные, используемые в циклах, но не участвующие многократно в вычислениях. Например, переменной, используемой в программе для хранения чьего-либо имени, логично дать имя *name*.

2. *Объявление переменной.* Объявление переменной – это определение ее типа и имени, которое всегда должно предшествовать обращению к этой переменной. Если в C++ переменная не объявлена, при компиляции программы (переводе программы на язык, близкий к машинному) будет сгенерирована ошибка.

3. *Инициализация переменных.* После объявления переменной её рекомендуется инициализировать, т.е. присвоить ей какое-либо значение, единицу или ноль. Это очень актуально для переменных, используемых в вычислениях. Дело в том, что при объявлении переменной для нее выделяется (резервируется) память. Резервирование памяти не очищает ячейки от значений, которые ранее в них хранились, поэтому, если за объявлением переменной не следует её инициализация, то текущее значение этой переменной будет непредсказуемым. При некоторых условиях компиляторы могут осуществлять очистку памяти, выделяемой под переменные.

4. *Стиль записи программы.* Рекомендуется придерживаться еди-

ного стиля оформления текста (интервалов, отступов, принципов записи конструкций языка, принципов именования переменных и т.д.) в пределах всей программы. Основная цель – повышение читабельности и, следовательно, понятности программы.

Пример оформления фрагмента программы без использования стилей:

void main ( ) { cout <<"Hello, world"; getch ();};

Пример оформления фрагмента программы с использованием стилей:

```
void main ( )
{
cout <<"Hello, world";
getch ();
}
```

Некоторые рекомендации по использованию стилей:

– знаки (+ - = * /) пишутся через пробел;

– для зрительного разделения отдельных частей программы (больших фрагментов комментариев) используется штриховая линия типа:

// ---------- ;

– скобки выравниваются вертикально по левой границе;

– в программу необходимо включить комментарии. Они должны быть хорошо составлены, иметь правильную пунктуацию, по возможности без сокращений, и выровнены вертикально. Комментарии, в общем, воспринимаются лучше, когда помещаются в многострочных блоках, которые чередуются с блоками текста программы. Для этого комментарий должен описывать на высоком уровне, что делают несколько последующих строк кода. Не стоит перегружать программу комментариями; рекомендуется комментировать фрагменты программы (логические блоки кода, циклы и т.п.), а не каждый отдельный оператор. Например:

```
//------------------------------
// Объявляем переменные:
// a – номер дня недели
// b – номер месяца года
```

```
// с – год
//-------------------------------
int  a, b, c;

// вводим значения a, b, c
cout <<"Vvedite a – ";
cin >>a;
cout <<"Vvedite b – ";
cin >>b;
cout <<"Vvedite c – ";
cin >>c;
```

Практическая работа № 1

1. Цель работы

Приобретение обучающимися умений и навыков в работе с оборудованием компьютерного класса, с системой программирования *Borland CPP* 5.02, правилами безопасной работы.

2. Упражнение

1. Найдите на Рабочем столе своего компьютера ярлык *Borland CPP*, запустите программу.

2. Перед вами появится окно программы. Первая строка экрана содержит все команды главного меню: *File* (файл), *Edit* (редактор), *Search* (поиск), *View* (вид), *Project* (проект), *Script* (скрипт), *Tools* (инструменты), *Debug* (отладка), *Options* (опции), *Window* (окно), *Help* (помощь). Все они имеют собственные подменю. Вызов функций подменю осуществляется перемещением курсора к нужному элементу и нажатием левой кнопки мыши.

3. Программные задачи оформляются в виде проектов. Обычно для каждой программы создается свой проект. Проект (*project*) представляет собой набор файлов, которые совместно используются для создания одной программы. Создайте и запустите первый проект. Для этого в главном меню выберите пункт *File* → *New*.

4. Перейдите на вкладку *Project*. В появившемся окне (рис. 1) в по-

ле *Project Path and Name* укажите каталог для проекта. В поле *Target Name* введите имя проекта – *zadanie1*, в поле *Target Model* выберите *Console*. Нажмите кнопку ОК.

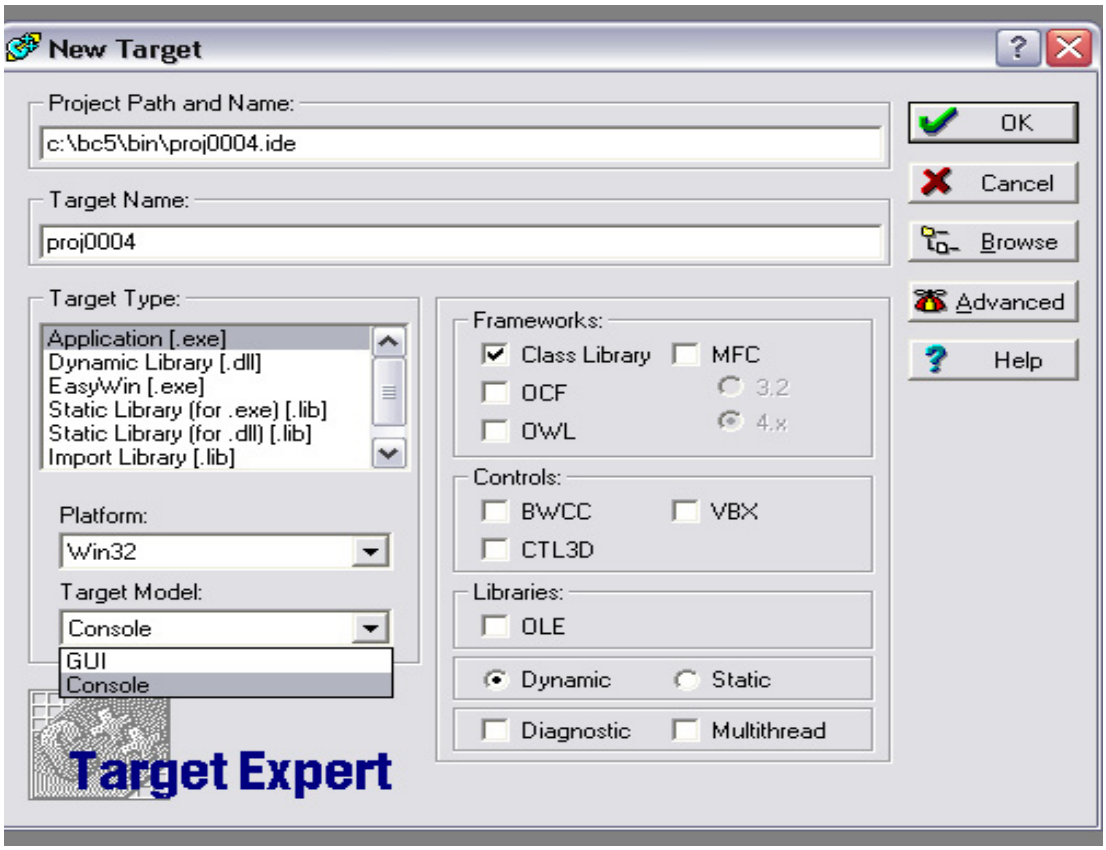

Рис. 1

5. Для текста программы вам понадобится файл *zadanie1.cpp*. Для создания файла *zadanie1.cpp* его необходимо выбрать в окне слева (рис. 2).

6. Перейдите в окно редактора и наберите листинг программы, приведенный ниже.

```
// zadanie1.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
cout <<”Hello”;
getch ();
}
```

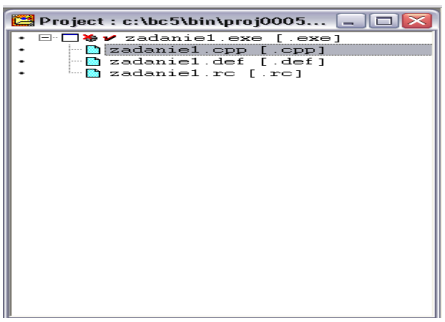

Рис. 2

7. Сохраните текст программы. Для этого в Главном меню выберите последовательно пункты *File* → *Save*.

8. Откомпилируйте программу. Для этого в Главном меню выберите *Project* → *Compile* или нажмите комбинацию клавиш Alt + F9 (рис. 3).

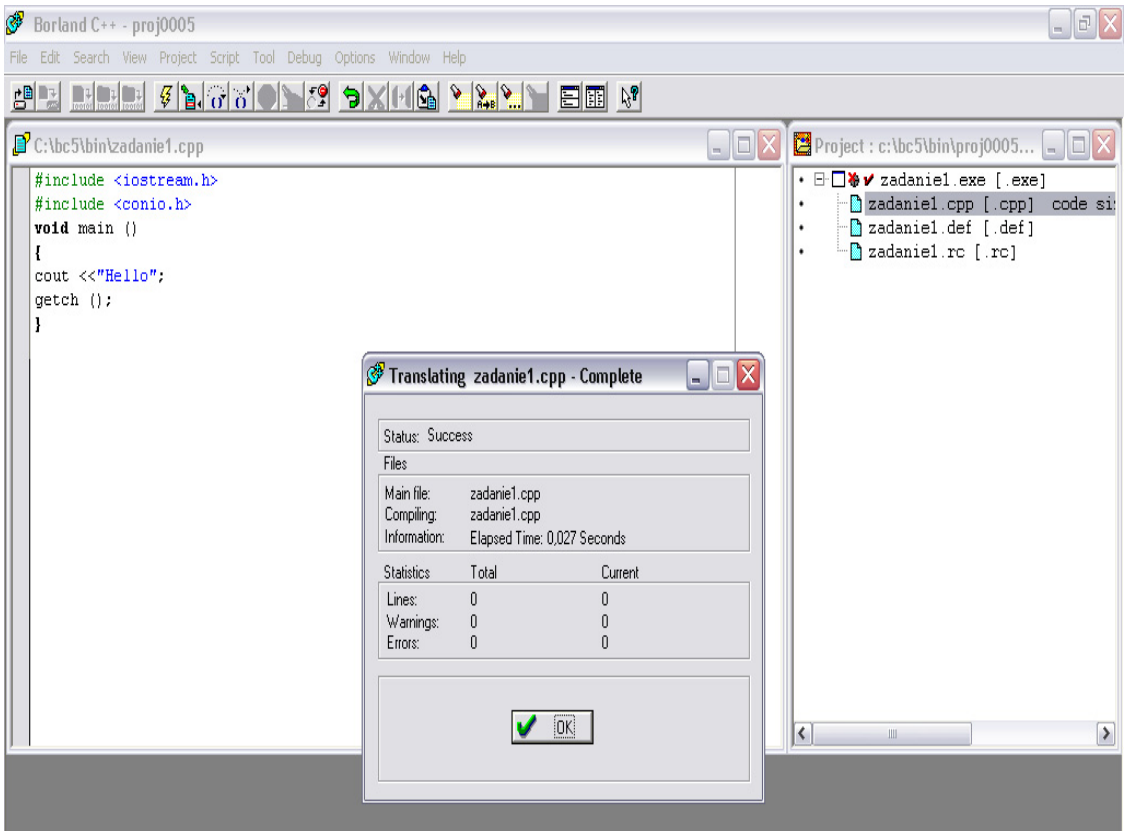

Рис. 3

9. Запустите программу на выполнение. Для этого в Главном меню последовательно выберите *Debug* → *Run* или нажмите комбинацию Ctrl + F9.

Общая структура программы на языке C++

Рассмотрим детально набранную программу, которая начинается с конструкции *#include < iostream.h >*. Эта запись означает подключение библиотеки *iostream.h* – стандартной библиотеки потокового ввода-вывода. Аналогичная запись *#include <conio.h>* подключает библиотеку консольного ввода-вывода (понятие консоль объединяет комплект устройств интерактивного ввода-вывода, подсоединенных непосредственно к компьютеру: монитор, клавиатуру, мышь и т.д.).

Подключение библиотек делает возможным вывод на экран текстовой строки с помощью операции *cout* << из состава библиотеки потокового ввода-вывода и использование функции *getch* () из состава библиотеки консольного ввода-вывода для задержки завершения работы приложения до нажатия любой клавиши.

Программная библиотека представляет собой набор функций, облегчающих работу пользователя. В табл. 1, 2 приведены некоторые из часто используемых функций.

Таблица 1

Функции библиотеки <stdio.h>

Функция	Назначение
printf ()	Вывод текста на экран
scanf ()	Ввод с клавиатуры
fopen ()	Открытие файла
fclose ()	Закрытие файла
gets ()	Ввод строки
puts ()	Вывод строки

Таблица 2

Функции библиотеки <conio.h>

Функция	Назначение
getch ()	Ожидание нажатия клавиши
clrscr ()	Очистка экрана
gotoxy ()	Перемещение курсора
textcolor ()	Выбор цвета текста
kbhit ()	Проверка нажатия клавиши
textmode ()	Изменение режима

Программа содержит заголовок функции с именем *main ()*. Выполнение любой программы на C++ начинается с вызова функции *main ()*, поэтому каждая программа на языке C++ должна ее содержать.

Следующая строка

{

содержит открывающуюся фигурную скобку, обозначающую начало

тела функции *main ()*, состоящего из набора объявлений, определений и операторов. Каждое из них должно завершаться символом точки с запятой.

Последняя строка программы

}

содержит закрывающуюся фигурную скобку, которая обозначает конец функции *main ()* и конец основной части программы (в большинстве случаев – конец программы).

Практическая работа № 2

1. Цель работы

Приобретение обучающимися умений и навыков в работе с элементами языка, структурой С++ – программы в использовании оператора *cout*, функций *main (), getch ()*.

2. Упражнение

Найдите ошибки в записи программы:

а)
```
#include <iostream>
#include <conio.h>
   void main ()
{
  cout << “segodna 2009 god”;
  getch ();
 }
```

б)
```
#include <iostream>;
#include <conio.h>;
   void main ();
{
  cout<<“S dnem rogdenia!!!”;
  getch ();
 }
```

в)
```
#include <iostream.h>
#include <conio.h>
   void main ()
{
  cout << “ Kaniculi !!!”;
  getch ();
```

г)
```
#include <stdio.h>
#include <conio.h>
   void main ()
{
  cout<< “S novim godom !!!”;
  getch ();
 }
```

д)
```
#include <iostream.h>
#include <conio.h>
   void main ()
{
```

```
    cout  “ segodna 2010 god”;
    getch ();
   }
e) #include <iostream.h>
   void main ()
{
  cout  “ segodna 2009 god”;
  getch;
 }
```

3. Вопросы и задания для отчета

1. Расскажите о назначении и синтаксисе комментариев в программе.

2. Приведите структуру простой программы на С++.

3. Перечислите часто используемые функции библиотек *stdio.h* и *conio.h.*

4. Назначение функции *main ().*

Занятие 2. ОБЪЯВЛЕНИЕ И ИНИЦИАЛИЗАЦИЯ ПЕРЕМЕННЫХ. СТАНДАРТНЫЕ ТИПЫ ДАННЫХ

Основная задача большинства компьютерных программ – быстрое выполнение большого количества вычислительных операций. В целях увеличения производительности промежуточные результаты расчетов хранятся в оперативной памяти. Программы С++ для размещения своих данных в оперативной памяти используют переменные. В языке С++ имена, которые используются для обозначения переменных, называются идентификаторами.

Идентификатор может содержать латинские буквы, цифры и символ подчеркивания и начинаться обязан с буквы или символа подчеркивания. В стандарте *ANSI* языка С++ идентификатор определяется своими первыми 32 символами. Строчные и прописные буквы рассматриваются в С++ как разные символы. Идентификатор не должен совпадать с ключевыми словами (командами, конструкциями языка).

В языке С++ все переменные должны быть объявлены до их использования. В нем определены шесть типов переменных, которые можно назвать базовыми (табл. 3).

Таблица 3

Стандартные типы C++

Тип	Название типа	Диапазон возможных значений
char	Символьный	Символы ASCII, числа от –128 до 127
int	Целый	От –32768 до 32767
float	Вещественный	От $3{,}4\cdot10^{-38}$ до $3{,}4\cdot10^{+38}$
double	Вещественный двойной точности	От $1{,}7\cdot10^{-308}$ до $1{,}7\cdot10^{+308}$
void	Пустой, не имеющий значения	–
bool	Логический	*true* или *false*

Если исключить из представления целых чисел знак, то полученный тип данных будет представлять неотрицательные числа с удвоенной верхней границей диапазона представления (табл. 4).

Таблица 4

Беззнаковые целые типы C++

Название типа	Нижняя граница диапазона	Верхняя граница диапазона	Размер в байтах
unsigned char	0	255	1
unsigned int	0	65 535	2
unsigned long	0	4 294 967 295	4

При объявлении переменная также может быть инициализирована (определено ее начальное значение) некоторой величиной из диапазона допустимых значений. Для этой цели используется оператор присваивания «=». Общая форма объявления переменной выглядит так:

Тип_переменной идентификатор_переменной [= начальное значение];

В квадратных скобках указано необязательное выражение. Можно считать, что неинициализированная переменная не имеет определенного значения (точнее, ее значение непредсказуемо).

Объявление переменной может размещаться почти в любом месте программы. Однако оно всегда должно предшествовать первому обращению к этой переменной. Одна и та же переменная может быть объявлена несколько раз в разных блоках программы. Нельзя объявить дважды одну переменную в одном блоке программы (в цикле, функции и т.д.).

Примеры объявления и инициализации переменных:

int x = 10; // переменная x целого типа и начальным значением 10

float a, *b*, *c*; // неинициализированные вещественные переменные *a*, *b*, *c*

char s = ’a’; // инициализируем символьную переменную *s* буквой «*a*»

char s [20]; // текстовая строка из 20 символов с именем *s*

При объявлении двух или более переменных одного типа в форме списка можно одну из них (или несколько) обеспечить начальными значениями. При этом все элементы списка разделяются запятыми. Например, *int a*, *b* = 8, *c* = 19, *d*;

В C++ определен широкий набор операций. Существуют четыре общих класса операций: арифметические, поразрядные, логические и операции отношений. В C++ определены следующие арифметические операции (табл. 5).

Таблица 5

Арифметические операции

Операция	Знак в C++	Запись на C++
Сложение	+	$a + 7$
Вычитание	–	$p - c$
Умножение	*	$b \cdot m$
Деление	/	x / y
Остаток от деления (деление по модулю)	%	$x \ \% \ y$

Кроме арифметических операций C++ дает удобные возможности использования математических функций (табл. 6). Большая их часть

содержится в библиотеке math.h и для их пользования требуется подключение соответствующей библиотеки (директива *#include <math.h>*).

Таблица 6

Наиболее употребительные математические функции

Название функции	Обозначение	Запись С++
Синус	sin x	sin (x)
Косинус	cos x	cos (x)
Тангенс	tg x	tan (x)
Квадратный корень	$\sqrt{x}$	sqrt (x)
Возведение в степень	x^y	pow (x,y)
Экспонента	e^x	exp (x)
Натуральный логарифм	ln x	log (x)
Модуль	$\lvert x \rvert$	fabs (x)
Арксинус	arcsin x	asin (x)
Арккосинус	arccos x	acos (x)
Арктангенс	arctg x	atan (x)

Все перечисленные функции принимают в качестве аргумента вещественную переменную (или константу) и возвращают вещественный результат.

В библиотеке *math.h* также определены некоторые часто используемые математические константы, часть из которых приведена в табл. 7.

Таблица 7

Математические константы

Константа	Запись в С++
π	M_PI
e	M_E
ln2	M_LN2
ln10	M_LN10

Практическая работа № 3

1. Цель работы

Приобретение практических навыков в работе с основными типами переменных, в записи выражений на языке программирования C++.

2. Упражнения

Упражнение 1

Какие из следующих имен переменных недопустимы?

а) *count*

б) *_count*

в) *count*27

г) 67*count*

д) *if*

Упражнение 2

Создайте новое консольное приложение со следующим исходным кодом. Запишите набранную программу в свой каталог под любым именем. Запустите программу на выполнение. Проанализируйте листинг исходного кода и объясните, для каких целей здесь использованы переменные *name*, *cm*, *inch*? Сформулируйте алгоритм работы этой программы.

```
//zadanie2.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
char name [20];
float cm, inch;
// ввод значения переменной name с клавиатуры
cout <<”Enter your name –“;
cin>> name;
cout <<”Enter your height –“;
cin >> cm;
inch = cm/2.54;
cout <<”Your heigt is”<<inch;
getch ();
}
```

Упражнение 3

Исправьте ошибки в программе, проверьте работу программы со следующими числами *a, b*: 32766, 32768; 23, 0.00005; 15, 20.

```
//zadanie3.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
 clrscr ();
 int a,b;
 float c;
 cout<<"vvedite a,b\n";
 cin>>a>>b;
 c = a*b;
 cout<<"\n c = "<<c;
 c = a / b;
 cout<<"\n c = "<<c;
 getch ();
}
```

Упражнение 4

Напишите программу, запрашивающую у пользователя длины сторон треугольника *a*, *b*, *c* и выводящую на экран углы α, β, γ.

Упражнение 5

Выполните программу вычисления математического выражения

$$\frac{(13{,}72+\cos 60^\circ)(7{,}58+\sin 60^\circ)}{\ln 10}-\sqrt{\frac{17{,}51}{\cos 30^\circ}}.$$

3. Вопросы и задания для отчета

1. Какое действие выполняет директива *#include <iostream.h> (#include <conio.h>)?*
2. Какое ключевое слово в C++ служит для объявления данных целочисленного (вещественного, символьного) типа? Каким образом в C++ объявляется текстовая строка?
3. Можно ли использовать переменные типа char для представления небольших целых чисел?

4. Найдите синтаксические и логические ошибки в приведенных ниже конструкциях:

а) *int a*; *b*;
cout >> *b*;
cin << *a*;

б) *float x* = –0,5;
float y = |*x*|;

в) *float x*, *y*, *z*;
float x = 1.5;

г) *float i* = 2∗exp(x));

д) *float x*, *y*, *z*;
z = sin (*x*, *y*);
z = *pow* (*xy*)

е) *int a* = 1;
int b = cos (*a*);

5. Найдите произведение цифр заданного трехзначного числа.

6. Напишите программу, рассчитывающую значение заданной функции и выводящую его на экран. Значения аргумента должны вводиться с клавиатуры.

$$\sqrt{\frac{12x + x^4}{x^2 + \sqrt{234 + z}}}.$$

Занятие 3. ОРГАНИЗАЦИЯ ВВОДА-ВЫВОДА В С++

С++ дает пользователю различные возможности для программирования ввода-вывода. Эти возможности реализуются с помощью функций, входящих в состав различных библиотек. Далее мы рассмотрим два варианта – консольный и потоковый ввод-вывод.

Консольный ввод-вывод. Удобная компьютерная программа – это программа, корректно взаимодействующая с пользователем, запрашивающая все необходимые для ее работы данные, используя при этом операции ввода, и выдающая результат с помощью операций вывода. Стандартным устройством ввода в ПК считается клавиатура, устройством вывода – экран монитора. Совокупность клавиатуры и монитора называется консолью.

Консольный ввод-вывод организуется с помощью функций библиотек *stdio.h* и *conio.h*, что предполагает наличие директив *#include* <*stdio.h*> и *#include* <*conio.h*> в заголовочной части программы.

Функции *printf ()* и s*canf ()* осуществляют форматированный вывод и ввод на консоль. Это значит, что функции могут читать и выводить данные в разном формате, которым пользователь может управлять. Управляющая строка содержит два типа информации: символы, которые непосредственно выводятся на экран, и команды формата, определяющие, как выводить аргументы. Команды формата начинаются с символа %, за которым следует код формата. Коды формата для стандартных типов данных указаны в табл. 8:

Таблица 8

Коды формата для стандартных типов данных

Переменная	Команда формата
Целое десятичное число со знаком	%d
Вещественное число	%f
Вещественное число двойной точности	%lf
Текстовый символ	%c
Целое число без знака	%u

Printf () – функция вывода информации на консоль. С ее помощью в окне приложения можно вывести как строку простого текста, так и значения переменных различных типов.

Общая форма записи функции:

printf (“форматная строка“ [, перем1], [перемен2] […]);

Здесь в круглых скобках указаны обязательные параметры, а в прямоугольных– параметры, которые указываются по необходимости. Например, запись printf (“*Hello*! “) означает вывод на экран простой текстовой строки «*Hello*!».

После выполнения приведенного ниже фрагмента кода программы на экран будут выведены значения целочисленной переменной *i*, вещественной переменной *f* и вещественной переменной двойной точности *d*:

```
int i=5;
float f = 1.5;
double d = 3.141592;
printf ("%d %f %lf", i, f, d);
```

Функция *printf* также дает возможности управления выводом с

помощью эскейп-последовательностей, начинающихся с символа ESC (обратный слэш «\»). Некоторые из них приведены в табл. 9.

Табл. 9

Эскейп-последовательности

Управляющий символ	Название	Действие
\n	*lf* (*line feed*)	Перевод строки
\a	*bel* (*audible bell*)	Звуковой сигнал
\b	*bs* (*backspase*)	Возврат на шаг (забой)
\t	*ht* (*horizontal tab*)	Табуляция
\v	*vt* (*vertical tab*)	Вертикальная табуляция

Scanf () – функция ввода с консоли. Общая форма записи этой функции: *scanf* (“форматная строка”, &перем1 [, &перем2] [, …]).

Аргументы функции *scanf* аналогичны соответствующим аргументам функции *printf*, за исключением того, что в качестве параметров *scanf* принимает не имена переменных, а их адреса. В силу этого перед именем каждой переменной в *scanf* должен стоять знак операции взятия адреса & (амперсанд). Например, команда вводит данные в переменные *a*, *b*, *c* через их адреса в целочисленном формате: *scanf("%d %d %d",&a,&b,&c);*

Потоковый ввод-вывод организуется с помощью библиотеки *iostream.h*, что предполагает наличие директивы *#include <iostream.h>* в заголовочной части программы. В библиотеке определены два потоковых объекта с именами *cin* и *cout*, которые связаны с клавиатурой и экраном компьютера соответственно. Для них определены следующие операции:

– извлечение из потока, т.е. ввод с клавиатуры *cin*;

– размещение в потоке, т.е. вывод на экран *cout*.

Общая форма записи этих операторов: *cin* >> переменная; *cout* << текстовая строка или переменная.

Например:

/* объявляется и одновременно инициализируется переменная целого типа *index* */

int index = 5;

// на экран выводится значение переменной *index*, равное 5

cout <<index;

/* объявляется переменная *x* вещественного типа, ее значение вводится *с клавиатуры* */

float x;

cin >>x;

/* объявляется переменная *a* целого типа, значение переменной *a* вводится с клавиатуры*/

int a;

cout <<"Enter a";

cin>>a;

Практическая работа № 4

1. Цель работы

Приобретение практических навыков в программировании ввода-вывода в C++.

2. Упражнения

Упражнение 1

Определите, что будет выведено на экран в результате выполнения приведенных ниже листингов. Предполагается, что переменные n, *m*, *x*, *y* объявлены следующим образом: *int n* = 1, *m* = 2; *float x* = 2,5, *y* = 5:

а) *cout* <<"*x* = *x*";

б) *cout* << x <<"="<<x;

в) *printf* ("%*d*", *n* + *m*);

г) *float z* = *n* + *m***x*;
printf ("*z*=%*f*", *z*);

Упражнение 2

```
   *
  * *
 *   *
*     *
 *   *
  * *
   *
```

Рис. 4

Напишите программу, выводящую на экран символы звездочки «*» в форме ромба с помощью средств консольного (потокового) ввода-вывода (рис. 4).

Упражнение 3

Используя управляющие последовательности, произведите на экран следующий вывод:

Фамилия	Имя	Адрес	Город
Петров	Василий	Кленовая, 16	Владимир
Иванов	Сергей	Василисина, 6	Владимир
Сидоров	Иван	Березовая, 21	Владимир

3. Вопросы и задания для отчета

1. Какие библиотеки необходимо подключить для организации консольного (потокового) ввода-вывода?

2. В чем заключается разница между потоковым и консольным вводом-выводом?

3. Определите, что будет выведено на экран в результате выполнения приведенных ниже листингов. Предполагается, что переменные *n*, *m*, *x*, *y* объявлены следующим образом: *int n* = 1, *m* = 2; *float x* = 2,5, *y* = = 5:

а) *cout* << ”*x* = ”<<*x*;

б) *cout* << *x* + *y*<<”!”;

в) *printf* (“%*d* %*d*”, *n*, 5).

4. Напишите программу, выводящую на экран символы звездочки «*» в форме ромба, средствами консольного (потокового) ввода-вывода (рис. 5).

```
*********
* *     * *
*   *   *
*********
```

Рис. 5

Занятие 4. СОКРАЩЕННЫЕ ВАРИАНТЫ ЗАПИСИ

Инкремент, декремент. Оператор инкремента выполняет сложение операнда с числом 1. Оператор декремента вычитает единицу из своего операнда. Инструкция x = x + 1 аналогична инструкции ++x. Инструкция x = x – 1 аналогична такой инструкции – –x. Операторы инкремента и декремента могут стоять как перед своим операндом (префиксная форма), так и после него (постфиксная форма).

Если оператор применен в префиксной форме, то C++ сначала выполнит эту операцию, чтобы операнд получил новое значение, которое затем будет использовано остальной частью выражения.

Если оператор применен в постфиксной форме, то C++ использует в выражении его старое значение, а затем выполнит операцию, в

результате которой операнд обретет новое значение Другие сокращенные варианты записи приведены в табл. 10.

Таблица 10

Сокращенные варианты записи

Стандартная запись	Описание	Сокращенная запись
A = A+B	Увеличить	A + = B;
A = A - B	Уменьшить значение A на величину B	A - = B;
A = A · C	Увеличить A в C раз	A* = C;
A = A / D	Уменьшить A в D раз	A/ = D

Порядок выполнения арифметических операций (табл. 11). Операторы одного уровня старшинства вычисляются компилятором слева направо. Для изменения порядка вычислений необходимо использовать скобки.

Таблица 11

Приоритет арифметических операций

Приоритет	Операторы
Наивысший	++ - -
……	* / %
Низший	+ -

Практическая работа № 5

Сокращенные варианты записи

1. Цель работы

Приобретение практических навыков в написании сок ращенных вариантов записи на языке C++.

2. Упражнения

Упражнение 1

Чему будет равно значение переменных *a*, *b* после выполнения фрагмента программы?

```
int a, b;
a = 2; b = 1;
a++;
```

a = 2;
++*a*;
a = 2;
b = *a*++;
a = 2;
b = ++*a*;

Упражнение 2

Чему будет равно значение переменной *i* после выполнения следующего кода программы?

int i = 2;
i+ = 2;

Упражнение 3

Вычислить значения, которые примут переменные после выполнения кода:

int a, *b* = 5, *c* = 7, *d* = 9;
a = *b*++ +*c*++ + ++*d*;

Упражнение 4

Чему будет равно значение переменой *k* после выполнения следующего кода?

int *k* = 1;
k + = ++ *k*;

Упражнение 5

Напишите программу, запрашивающую длину основания и высоту равнобедренного треугольника и вычисляющую длины его сторон.

Упражнение 6

Вычислите площадь треугольника по трем сторонам. $S = \sqrt{p(p-a)(p-b)(p-c)}$ при различных значениях сторон, где $p = \frac{1}{2}(a + b + c)$ – полупериметр треугольника.

3. Вопросы и задания для отчета

1. Чему будет равно значение переменных *x*, *y* после выполнения фрагмента программы?

а) *int x* = 10;
int y = 5;
y = ++*x*;

б) *int x* = 10;
int y = 5;
y = *x*++;

2. Чему будет равно значение переменных *x* и *tl* после выполнения следующего фрагмента программы:

а) *int x* = 10; *x* – = 100;
б) *int a* = 2, *b* = 3;
int tl = *b*∗++*a*;

3. Есть треугольник с основанием *a* и высотой *h*. Найдите площадь треугольника. Входные и выходные данные – дробные числа.

4. Напишите программу, вычисляющую размер регулярных платежей по займу на покупку автомобиля. Данные: основная сумма займа *a*, срок займа в годах *n*, количество выплат в год z и процентная ставка *pr*. Программа вычисляет размер платежа *pl*. Входные и выходные данные – вещественные числа. Для вычисления размера регулярных платежей по займу используется следующая формула:

$$pl = \frac{pr * (\frac{a}{z})}{(1 - (\frac{pr}{z}) + 1)^{-z*n}}.$$

5. Напишите программу, запрашивающую у пользователя радиус круга и выводящую на экран диаметр, длину окружности и площадь этого круга.

Занятие 5. ВЕТВЛЕНИЯ

В C++ существуют несколько типов ветвлений. Рассмотрим каждый из них. Условный оператор *if* служит для выбора направления работы программы в зависимости от условий, сложившихся в данной точке программы на момент ее выполнения.

Общая форма записи условного оператора:

```
if (условие)
{
    блок операторов 1;
}
else
{
    блок операторов 2;
}
```

Если на момент выполнения условие истинно, программа передает управление блоку операторов 1 и далее первому оператору за пределами конструкции *if-else*. При этом блок операторов 2 не выполняется. Если на момент выполнения условие ложно, выполняется блок операторов 2, а блок операторов 1 не выполняется. В табл. 12 указаны простейшие операции отношения.

Таблица 12

Операции отношения

Операция	Запись на С++
Больше	>
Меньше	<
Больше либо равно	>=
Меньше либо равно	<=
Равно	= =
Не равно	!=

Вложенные операторы условия

Операторы условия могут быть вложенными друг в друга в соответствии с тем программным алгоритмом, который они реализуют. Допускается произвольная степень их вложенности. Например:

```
if (a < = b)                        // начало внешнего оператора условия
{
      if (x! = 0) cout<<"x! = 0"// начало вложенного оператора условия
      else                          // начало ветви else, относящейся
                                    // к вложенному оператору условия
      {
            x = 1;
          y=0;
      }                             // конец ветви else, относящейся
                                    // к вложенному оператору условия
}
else                                // начало ветви else, относящейся
                                    // к внешнему оператору условия
{
      a = b;
      cout << a;
}                                   // конец ветви else, относящейся
                                    // к внешнему оператору условия
```

Сокращенные варианты записи

При программировании обыденной является ситуация, когда требуется некоторое действие в ответ на сложившиеся условия. Например, если получены неверные исходные данные от пользователя, то необходимо выдать сообщение об ошибке и выйти из программы. В таких случаях используется сокращенная запись оператора условия с отсутствующим блоком *else*. Общая форма записи:

if (условие)
{
    блок операторов;
}

Здесь в случае истинности условия управление передается блоку операторов в фигурных скобках. В случае ложности условия этот блок пропускается.

Составные логические выражения

В программировании распространены двойные условия, которые в математике записываются в виде f < b < c. В программе такие условия должны быть переформулированы с использованием простых операций сравнения и логических операций «И», «ИЛИ», «НЕ». Обозначение логических операций приведено в табл. 13.

Таблица 13

Логические операции

Логическая операция	Знак С++	Наименование знака
И	&&	Двойной амперсанд
ИЛИ	||	Двойная вертикальная черта
НЕ	~ (!)	Не

Например:

```
if ((a > b) && (a > c))          // если a больше b и a больше c,
    cout <<"a=" << a;            // вывести значение переменной a
else                             // иначе
{
```

```
        if ((b>a) && (b>c))          // если b больше a и b больше c,
              cout <<"b="<<b;        // вывести значение переменной b,
        else
              cout <<"c="<<c;        // иначе вывести значение
    }                                // переменной c
```

Оператор *Switch* используется в том случае, если в программе присутствует большое дерево ветвлений, которые зависят от значения какой-либо одной переменной.

Общий формат записи:

```
switch (n)
{
   case 1:
    оператор 1;
   break;
case 2:
    оператор 2;
   break;
case 3:
    оператор 3;
   break;
…….
}
```

где *n* – целочисленная или символьная переменная;
1, 2, 3 – целочисленная или символьная константа;
оператор 1 – тело первого *case*;
оператор 2 – тело второго *case*;
оператор 3 – тело третьего *case*;
(прервать) break –оператор завершает выполнение ветвления *switch*, если значение переменной в операторе *switch* совпадает с одним из значений констант, указанных внутри ветвления. Если значение переменной в операторе *switch* не совпадет ни с одним из значений констант, то управление будет передано в конец *switch* без выполнения каких-либо действий. В случае отсутствия оператора *break* управление будет передано операторам, относящимся к другим веткам *switch*.

Условная операция

В программировании существует распространенная операция: переменной необходимо присвоить одно значение в случае выполнения некоторого условия и другое значение в случае невыполнения этого условия. С помощью конструкции *if … else* это будет выглядеть следующим образом:

```
if (alfa < beta)
        min = alfa;
else
        min = beta;
```

Подобные действия на практике настолько распространены, что была специально разработана условная операция, выполняющая эти действия, которая записывается с помощью двух знаков и использует три операнда.

С помощью условной операции можно записать предыдущий фрагмент следующим образом: *min = (alfa<beta) ? alfa : beta;*

Правая часть оператора *(alfa<beta) ? alfa : beta* представляет собой условное выражение. Знаки «?» и «:» обозначают условную операцию. Условие стоит перед знаком вопроса *(alfa<beta)* и является условием проверки. Оно вместе с операндами *alfa и beta* составляет тройку операндов условной операции.

Если значение проверяемого условия истинно, то условное выражение становится равным значению *alfa*, в противном случае – *beta*. Скобки необходимы для того, чтобы визуально упростить читаемость этого оператора.

Практическая работа № 6

1. Цель работы

Приобретение практических навыков в работе с основными алгоритмическими конструкциями языка C++.

2. Упражнения

Упражнение 1

Найдите и исправьте ошибки в программе. Проверьте ее работу со следующими числами *a*, *b*, *c*: 1, 2, 3; 5, 5, 5; 2, 3, 3:

```
//zadanie4.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
clrscr ();
int a, b, c;
cout<<"vvedite a, b, c\n";
cin>>a>>b>>c;
if (a==b)
if (b==c)
cout <<"a, b, c ravni\n";
else
cout<<"a, b ne ravni\n";
getch ();
}
```

Упражнение 2

Даны три вещественных числа. Напишите программу, определяющую, могут ли данные числа являться длинами сторон треугольника.

Упражнение 3

Напишите программу, определяющую количество вещественных корней квадратного уравнения $ax^2 + bx + c = 0$. Значения a, b, c вводятся с клавиатуры. На экран выводится количество корней и их значение.

Упражнение 4

Дано целое число n ($1 \le n \le 99$), определяющее возраст человека (в годах). Для этого числа напечатайте фразу «мне n лет», учитывая при этом, что при некоторых значениях n слово «лет» надо заменить на слово «год» или «года».

Упражнение 5

Мастям игральных карт условно присвоены следующие порядковые номера: «пики» – 1, «трефы» – 2, «бубны» – 3, «червы» – 4. По

заданному номеру масти m ($1 \le m \le 4$) определите название соответствующей масти.

3. Вопросы и задания для отчета

1. Найдите синтаксические ошибки в приведенных ниже языковых конструкциях. Учтите, что переменные *a*, *b*, *c* объявлены как целые, а *x*, *y* – как вещественные числа:

а)
```
if (5>b)
  {
     a = 25*x + y;
     5 = b;
  }
  else
     cout <<”osibra!”;
```

б)
```
if (a = b)
     a>b;
  else
     a<b;
```

в)
```
if (x>0)
  {
     y = 2*sin(x)*exp(-x);
  }
     a = sqrt(x*x);
  else
     y = 0;
```

г)
```
if ((5>b>c) (b>c))
     cout <<”OK!”;
```

2. Определите, что будет выведено на экран в результате выполнения приведенных ниже фрагментов кода. Считайте, что переменные *a*, *b*, *c* объявлены как целочисленные и инициализированы значениями 1, 5 и 10 соответственно:

а)
```
if (5>b)
  {
     a = b + 1;
     b = a - 1;
  }
  else
     cout <<a<<b<<c;
```

б)
```
if (a>0)
     cout <<”OK!”;
  else
     b = a;
     cout <<a<<b<<c;
```

в)
```
if ((a<b) && (b>c))
     a+b;
  else
     b=a;
     cout <<a<<b<<c;
```

г)
```
if (c>a)
  if (c>10)
     printf (“===”);
  else
    printf (“<<<”);
  printf (“>>>”);
```

3. Дано двузначное число. Определите, равен ли квадрат этого числа учетверенной сумме кубов его цифр. Например, для числа 48 ответ положительный, для числа 52 – отрицательный.

4. Определите, попадает ли точка с заданными координатами в область I (для простоты принять, что точка не попадает на границу этой области) (рис. 6).

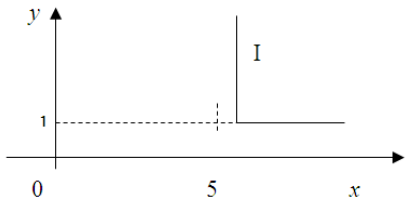

Рис. 6

5. Даны два числа. Если квадратный корень из второго числа меньше первого то увеличьте второе число в пять раз.

6. Составьте программу для вычисления значения функции z (a):

$$z = \begin{cases} 1, \text{если } \alpha > 0, \\ 0, \text{если } \alpha = 0, \\ -1, \text{если } \alpha < 0. \end{cases}$$

7. Даны вещественные положительные числа *a*, *b*, *c*. Если существует треугольник со сторонами *a*, *b*, *c*, то определите его вид (прямоугольный, остроугольный или тупоугольный).

Занятие 6. ЦИКЛЫ

Действие циклов заключается в последовательном повторении определенной части программы некоторое количество раз. Повторение продолжается до тех пор, пока выполняется соответствующее условие. Когда значение выражения, задающего условие, становится ложным, выполнение цикла прекращается, а управление передается оператору, следующему непосредственно за циклом.

В C++ существуют три типа циклов: *for*, *while*, *do*.

Цикл *for*

Цикл *for* организует выполнение фрагмента программы фиксированное число раз. Как правило (но не всегда), этот тип цикла используется тогда, когда число раз, за которое должно повториться исполнение кода, известно заранее.

Синтаксис:

for (<инициализация>*;* <условие продолжения>*;*<изменение счетчика>*)*
{

тело цикла;
}

Цикл *for* начинается с выполнения блока <инициализация>, где определяется начальное значение счетчика цикла. Далее выполняются операторы (оператор), образующие тело цикла. Затем проверяется <условие продолжения>, в случае, если это условие истинно, управление передается заголовку *for* и значение счетчика цикла автоматически изменяется в зависимости от параметра «изменение счетчика».

Рассмотрим следующий пример: напишите программу, выводящую на экран все целые числа от 0 до 99 включительно.

Можно предложить следующий алгоритм решения задачи:

1. Объявляем целочисленную переменную *k* и инициализируем ее значением 0.
2. Выводим на экран *k*.
3. Увеличиваем *k* на единицу.
4. Если k < 100, возвращаемся к п. 2.
5. Завершаем программу.

Переменная *k* называется счетчиком цикла, а повторяющиеся в цикле операторы – телом цикла.

Для решения данной задачи воспользуемся циклом *for*:

```
#include <iostream.h>          // подключаем библиотеки потокового
#include <conio.h>             // и консольного ввода-вывода
void main ()                   // точка начала программы
{                              // задаем тело функции main
clrscr ();                     // очищаем экран
  for (int k=0; k<100; k++)    // цикл по k с шагом 1
        cout <<" "<<k;         // задаем тело цикла
}                              // закрываем тело функции main
getch ();
}
```

В качестве параметра цикла необязательно использовать целочисленный счетчик. Параметром могут выступать символы. Например, следующая программа выводит на экран буквы английского алфавита:

```
#include <iostream.h>
 #include <conio.h>
 void main ()
 {
 clrscr ();
```

```
char ch;
for (ch='A'; ch<='Q'; ch++)
        cout << " "<<ch;
getch();
}
```

Сокращенные варианты записи

Отдельные (или все) блоки в заголовке цикла *for* могут быть пустыми, однако, разделительные точки с запятой обязательны. Например, следующий фрагмент программы, где в записи цикла *for* оператор инициализации пуст, так как основывается на значении переменной, которая была ранее объявлена и проинициализирована:

```
int i = 0;
int j;
int val1 = 0;
int val2;
i = 25;
j = i * 2;
for ( ; i < 100;  i++)
  val1 = i;
```

Вложенные операторы цикла

Аналогично условному оператору циклы могут быть вложены друг в друга, причем степень вложенности также может быть произвольной. В качестве счетчиков такие циклы, как правило, используют различные переменные.

Дополнительные средства управления циклами

К числу дополнительных средств управления циклами относятся операторы *break* (англ. «прервать») и *continue* (англ. «продолжить»).

С помощью оператора *break* организуется досрочное окончание цикла с передачей управления оператору, следующему непосредственно за концом цикла. Пусть, например, нам предстоит суммирование элементов массива *a* до тех пор, пока не встретится первое отрицательное значение:

```
for(s = 0, j = 0; j < n; j++)
  {
    if (a[j]<0) break;
    s += a[j];
  }
```

Однако если оператор *break* употреблен во внутреннем цикле, то с его помощью нельзя выйти за пределы внешнего даже в том случае, когда кажется, что тело внешнего цикла кончается там же, где и тело внутреннего. На самом деле, в конце каждого цикла незримо присутствуют системные вставки, обеспечивающие нормальный выход из цикла. В частности, такие вставки возвращают память, выделенную под переменные, объявленные в заголовке цикла. Кроме того, здесь же находятся команды, возвращающие управление в начало цикла при необходимости повторения итераций.

Цикл *while*

Этот тип цикла используется в том случае, если количество повторений заранее неизвестно.

Синтаксис:

```
while (условие выполнения)
  {
   тело цикла;
  }
```

Предварительно проверяется условие выполнения пока оно истинно, исполнение тела цикла продолжается. Как только оно становится ложным, происходит выход из цикла. Если с самого начала условие выполнения ложно, то тело цикла не выполняется ни разу.

Например, пользователю предлагают ввести серию значений. В том случае, когда вводимое значение оказывается равным 0, происходит выход из цикла:

```
#include <iostream.h>
#include <conio.h>
void main ()
{
clrscr ();
int n=99;
while (n!=0)
```

```
  {
  cout <<"vvedite n\n";
  cin>>n;
  }
}
```

Цикл *do*

Синтаксис цикла следующий:

```
do
{
тело цикла;
}
while (условие выполнения);
```

Оператор действует следующим образом: выполняются операторы циклической части, проверяется условие выполнения, если оно истинно, выполняется тело цикла. Если же оно ложно, то цикл заканчивается.

Например, предлагается ввести два числа: делимое и делитель, а затем произвести целочисленное деление с использованием операции /. После того как произведено вычисление, программа спрашивает пользователя, хочет ли он произвести вычисления еще раз. Если в ответ программа получает символ “*y*”, то вводятся значения делимого и делителя. Если символ “*n*”, то происходит выход из цикла:

```
#include <iostream.h>
#include <conio.h>
void main ()
{
clrscr ();
long a,b;
char ch;
do
{
cout<<"vvedite delimoe\n";
cin>>a;
cout<<"vvedite delitel\n";
```

```
cin>>b;
cout<<"chastnoe="<<a/b<<"\n";
cout<<"eshe raz? (y/n)\n";
cin>>ch;
}
while (ch!='n');
}
```

Практическая работа № 7

1. Цель работы

Приобретение практических навыков в работе с основными алгоритмическими конструкциями языка C++.

2. Упражнения

Упражнение 1

Напишите программу, вычисляющую сумму всех целых чисел в интервале от 8 до 20.

Упражнение 2

Напишите программу, вычисляющую x^k (где x – вещественное, k – целое) с использованием цикла *for*.

Упражнение 3

Напишите программу, вычисляющую факториал заданного целого числа N с использованием цикла *for*. Факториалом называется произведение всех натуральных чисел от 1 до n включительно, обозначается $n!=1\cdot2\cdot3\cdot\ldots\cdot n$. По определению полагают $0!=1$. Факториал определен для целых неотрицательных чисел.

Упражнение 4

Найти сумму четных делителей числа N.

Упражнение 5

Наберите текст следующей программы и запустите ее на выполнение. Проанализируйте листинг исходного кода; как изменяются

значения счетчиков внешнего и вложенного циклов? Сформулируйте алгоритм работы этой программы.

```
#include <iostream.h>
 #include <conio.h>
 void main ()
 {
 clrscr ();
int i,j;
   for (i=1; i<5; i++)
   {
          cout <<"i="<<i;
          for (j=1; j<3; j++)
          cout <<" j="<<j;
    }
}
```

Упражнение 6

Известна зарплата каждого из 12 работников фирмы за каждый месяц первого квартала.

Организовать ввод информации в форме таблицы и определить:

а) общую сумму, выплаченную за квартал всем работникам;

б) зарплату, полученную за квартал каждым работником;

в) общую заработную плату всех работников за каждый месяц.

Работники	Месяц		
	1	2	3
1			
2			
…			
12			

Упражнение 7

Вычислить четвертые степени последовательности целых чисел, начиная с 1, используя цикл *while*. Все выводимые значения должны иметь размер не более 4 символов.

Упражнение 8

Создайте эквивалент калькулятора, выполняющего четыре основных арифметических операции. При написании программы использовать оператор цикла *do*.

3. Вопросы и задания для отчета

1. Найдите синтаксические и логические ошибки в приведенных конструкциях. Считайте, что переменные *i*, *j* объявлены ранее как целые (*int*), переменные *x*, *y* – как вещественные (float).

а)
```
for (j = 0; j<2; i+ = 0.1)
{
 x = sin (j∗0.1);
 y = exp(-2∗j);
}
```

б)
```
{for (j = 0; j < 2; j++);
 x = j∗0.1;
cout <<x;
 }
```

2. Определите, что будет выведено на экран в результате выполнения приведенных ниже фрагментов. Предполагается, что переменные *i*, *j*, *k* объявлены как целочисленные (*int*) и инициализированы значениями 1, 5, 10 соответственно.

а)
```
for (i = 0; i<10; i++)
 if (i<j)
 cout <<i;
```

б)
```
for (i = 0; i<10; i+ = 2)
 if (i<j);
 cout <<j;
```

в)
```
for (i = 0; i<3 i++)
 {
  k- =5;
  if (i<k)
     printf (“$$$”);
  else
     printf (“###”);
 }
```

3. Напишите программу, вычисляющую сумму ряда $S = \sum_k^{10} \frac{k^2}{2^k}$.

4. В соревнованиях по фигурному катанию спортсмены выступают в трех видах многоборья (обязательная, короткая и произвольная про-

граммы). Известны результаты (в баллах) каждого из 15 участников соревнования:

Спортсмен	Программа		
	Обязательная	Короткая	Произвольная
1			
2			
…			
15			

Организовать ввод информации в форме таблицы и определить:

а) среднее количество баллов, полученных каждым спортсменом;

б) среднее количество баллов, полученное по каждому виду программы.

5. Даны натуральные числа n, a_1, a_2, …, a_n. Вычислить: min (a_2, a_4, …) + max (a_1, a_3, …).

6. Измените программу, вычисляющую факториал числа, таким образом, чтобы она циклически запрашивала ввод пользователем числа и вычисляла его факториал, пока пользователь не введет 0. В этом случае программа должна завершиться.

Занятие 7. СТРУКТУРЫ

Структура – объединение простых переменных, которые могут иметь различные типы: *float*, *char*, *int*. Переменные, входящие в состав структуры, называются *полями* структуры.

Структуры – одна из составляющих главных концепций языка – объектов и классов, использующаяся, как правило, в качестве объединения данных.

Определение структуры выглядит следующим образом:

```
 struct  <имя структуры>
{
  члены структуры;
};
```

Простая структура

Начнем рассмотрение со структуры, содержащей три поля, два из

которых имеют целый тип, и одно поле – вещественный тип. Эта структура предназначена для хранения информации о комплектующих деталях изделий, выпускаемых фабрикой. Компания производит несколько типов изделий, поэтому номер модели изделия включен в структуру как первое из ее полей. Номер самой детали представлен вторым полем, а ее стоимость – третьим полем.

```
//parts.cpp
#include <iostream.h>
#include <conio.h>

struct part
{
   int modelnumber;
   int partnumber;
   float cost;
 };
```

Определение структуры *part* необходимо для того, чтобы создавать на его основе переменные типа *part*.

Определение структурной переменной

```
void main ()
{
  part part1;
```

Первый оператор функции *main* () выглядит следующим образом: part part1;

Он представляет собой определение переменной *part*1, имеющей тип *part*. Определение переменной означает, что под эту переменную выделяется память. Важно помнить, что под структурную переменную всегда отводится столько памяти, сколько достаточно для хранения всех ее полей.

Доступ к полям структуры

Когда структурная переменная определена, доступ к ее полям возможен с применением *операции точки (доступ к полю структуры)*. В выражении на первом месте ставят имя структурной перемен-

ной, затем – операции точки, на третьем месте – имя поля.

```
part1. modelnumber = 6244;
part1. partnumber = 373;
part1. cost = 217.55;
```

С полями структурной переменной можно обращаться так же, как с обычными простыми переменными. В приведенном выше примере с помощью оператора присваивания поля структуры инициализируются и выводятся на экран:

```
cout<<      “model “<<part1. Modelnumber<<” “;
cout<<      “detal “<<part1. Partnumber<<” “;
cout<<      “Stoimost “<<part1. cost<<”  “;

getch();
}
```

Структуры обладают достаточно широким набором возможностей. Рассмотрим некоторые из них.

Следующий пример демонстрирует способ, при помощи которого можно инициализировать поля предварительно определенной структурной переменной. В программе используются две структурные переменные *part*1, *part*2.

```
//part.cpp
#include <iostream.h>
#include <conio.h>
struct part
{
   int modelnumber;
   int partnumber;
   float cost;
  };

void main ()
{
 //инициализация переменной в момент ее объявления, первая из величин
// присваивается первому полю, вторая – второму и т.д. через перечисление
part part1={6244, 373, 217.55};
```

```
//объявление второй структурной переменной
part part2;

//вывод полей первой переменной
cout<<      “model “<<part1. modelnumber;
cout<<      “detal “<<part1. partnumber;
cout<<      “Stoimost “<<part1. cost;

//присваивание значений одной структурной переменной другой
part2=part1;

//вывод полей второй переменной
cout<<      “model “<<part2. modelnumber;
cout<<      “detal “<<part2. partnumber;
cout<<      “Stoimost “<<part2. cost;

getch();
}
```

Результат работы программы:

```
model 6244 part 373 cost 217.55
model 6244 part 373 cost 217.55
```

Практическая работа № 8

1. Цель работы

Приобретение практических навыков в работе с простыми структурами языка C++.

2. Упражнения

Упражнение 1

Наберите код приведенной в тексте лекции программы и запустите её на выполнение. Проанализируйте листинг исходного кода и объясните, для каких целей используются переменные *modelnumber*, *partnumber*, *float cost*? Что означают записи *struct part*, *part part1*?

Упражнение 2

Номер телефона, например (4922) (32-67-56), можно условно разделить на три части: код города (4922), номер телефонной станции (326) и номер абонента (756). Напишите программу с использованием структуры, позволяющую раздельно хранить эти три части телефонного номера. Назовите структуру *phone*. Создайте две структурные переменные типа *phone*. Инициализацию одной из них произведите сами, а значения другой запросите с клавиатуры. Затем выведите содержимое обеих переменных на экран. Результат работы программы должен выглядеть следующим образом.

Введите код города, номер станции и номер абонента:
415
555
1212
Мой номер:
(212) 767-8900
Ваш номер:
(415) 555-1212

Проверьте работу программы со следующими номерами: (922) (32-66-63), (922) (32-60-63). Объясните полученный результат работы программы.

Упражнение 3

Создайте структуру с именем *time*. Три ее поля, имеющие тип *int*, будут называться *hours*, *minutes*, *second*. Напишите программу, которая просит пользователя ввести время в формате «часы, минуты и секунды». Программа должна хранить время в структурной переменной типа *time* и выводить количество секунд в введенном времени.

Упражнение 4

Известны данные о стоимости каждого из трех наименований товаров: число рублей и число копеек. Составить программу с использованием структуры, сравнивающую стоимость двух любых наименований товаров (какой товар стоит дороже). Для простоты принять, что стоимости разные.

3. Вопросы и задания для отчета

1. Что объединяет структура?
2. Каким образом осуществляется доступ к полям структуры?
3. Выделяется ли память под переменную при определении структуры?
4. Расположение точки на плоскости можно условно задать с помощью двух координат: *x* и *y*. Сумма двух точек определяется как точка, имеющая координаты, равные сумме соответствующих координат слагаемых.
5. Напишите программу, использующую для интерпретации точки на плоскости структуру с названием *point*. Определите три переменные типа *point*, две из них инициализируйте с помощью значений, вводимых с клавиатуры. Затем присвойте третьей переменной значение суммы первых двух переменных и выведите результат на экран.
6. Дан список, состоящий из названий трех городов, в котором указано количество жителей. Вывести названия городов с численностью жителей более 100 000.

Занятие 8. ФУНКЦИИ

Понятие функции в C++ аналогично понятию подпрограммы, которое имеется в большинстве языков программирования. Их использование связано с принципами модульного программирования, в основе которого лежит разделение программы на автономные фрагменты, или модули.

Таким образом, функции в C++ - это самостоятельные единицы программы, спроектированные для решения конкретных задач, обычно повторяющихся несколько раз. Примером являются все рассмотренные ранее функции стандартных библиотек *stdio.h*, *conio.h*, *math.h*. Однако пользователь может не только использовать готовые функции, но и создавать собственные. Более того, несколько функций пользователя могут быть впоследствии объединены в библиотеку.

Определение функции

Определение содержит код функции. Основная форма определения функции имеет вид:

тип возвращаемого значения <имя функции> (список параметров)
{
тело функции
}

Тип возвращаемого значения – тип значения, возвращаемого функцией. Это может быть практически любой тип. Если функция не возвращает никакого значения, необходимо указать тип *void*. Если функция действительно возвращает значение, оно должно иметь тип, совместимый с указанным в определении функции.

Имя функции, как правило, отражает выполняемое ею действие. В качестве имени можно использовать любой допустимый идентификатор, который еще не был задействован в программе. После имени функции в круглых скобках указывается *список параметров,* который представляет собой последовательность пар (состоящих из типа данных и имени), разделенных запятыми. Параметры – это, по сути, переменные, которые получают значение аргументов, передаваемых функции при вызове. Если функция не имеет параметров, элемент «список параметров» отсутствует, т.е. круглые скобки остаются пустыми.

В фигурные скобки заключено *тело функции*, состоящее из последовательности операторов. Когда происходит вызов функции, программа передает управление первому оператору тела функции. Затем исполняются операторы, находящиеся в теле функции, и, когда достигается закрывающаяся фигурная скобка или инструкции *return*, управление передается обратно вызывающей программе. Для *обращения к функции* (ее вызова) используется имя функции с указанием набора передаваемых ей параметров в круглых скобках.

Примером может служить функция, выбирающая наименьшее из трех чисел в приведенном ниже листинге программы:

```
#include <iostream.h>                  // подключаем библиотеки потокового
#include <conio.h>                     // и консольного ввода-вывода
float min (float a, float b, float c)  // определяем функцию min
{
 float m=a;                            // определяем локальную переменную m
 if (b<m) m=b;                         // выбираем наименьшее из двух чисел
 if (c<m) m=c;                         // выбираем наименьшее из двух чисел
 return m;                             // возвращаем результат к месту вызова
}
```

```
void main ()                          // точка начала программы
{
clrscr ();
float a=2.5, b=3.1, c=5.8;            // задаем три аргумента для функции
float minimal;                        // объявляем переменную для результата
minimal=min (a,b,c);                  // вызываем функцию
cout <<"min="<< minimal;              // выводим результат на экран
getch();
}
```

Прототип функции

Прототип функции – это ее опережающее объявление. В прототипе указывается только имя функции, тип результата и типы аргументов. Он не содержит реализации (тела) функции, а всего лишь указывает компилятору, что ее определение будет дано ниже в тексте программы. Если программа использует прототип функции, то обычно он располагается перед функцией *main* или внутри нее, а определение функции дается после функции *main*.

В следующем примере рассмотренная ранее функция *min* объявлена с помощью прототипа:

```
#include <iostream.h>                 // подключаем библиотеки потокового
#include <conio.h>                    // и консольного ввода-вывода
float min (float a, float b, float c); // объявляем функцию min с помощью
                                        прототипа
void main ()                          // точка начала программы
{
clrscr ();
float a=2.5, b=3.1, c=5.8;            // задаем три аргумента для функции
float minimal;                        // объявляем переменную для результата
minimal=min (a, b, c);                // вызываем функцию
cout <<"min="<< minimal;              // выводим результат на экран
getch ();
}
float min (float a, float b, float c) // определяем функцию min
{
 float m=a;
if (b<m) m=b;
if (c<m) m=c;
return m;
}
```

Передача аргументов в функцию

Аргументом называют единицу данных (например, переменную типа *int*), передаваемую программой в функцию. Аргументы позволяют функции оперировать различными значениями или выполнять различные действия в зависимости от переданных ей значений.

В качестве аргумента могут выступать константы, значения переменных, структурные переменные. В случае передачи констант при вызове функции вместо аргументов в скобках указывают их конкретные значения. В случае передачи значений переменных в функцию типы и имена этих переменных указываются в прототипе при определении функции (см. примеры выше). Структурные переменные в качестве аргументов функций используются как и любые другие переменные стандартного типа.

Рассмотрим пример, в котором аргументом функции будет структура Distance, представляющая длину в английской системе мер.

```
//table.cpp
#include <iostream.h>
#include <conio.h>
// объявляем структуру
struct Distance
{
 int feet;
 float inches;
};
//объявляем функцию
void eng (Distance);
void main ()
{
 clrscr ();
//объявляем переменные типа  Distance
Distance d1, d2, d3;
 cout<<"vvedite chislo futov";
 cin>>d1.feet;
 cout<<"vvedite chislo djumov";
 cin>>d1.inches;

 cout<<"vvedite chislo futov";
```

```
 cin>>d2.feet;
 cout<<"vvedite chislo djumov";
 cin>>d2.inches;
//складываем дюймы
 d3.inches=d1.inches+d2.inches;
 d3.feet=0;
 if (d3.inches>=12.0)
    {
      d3.inches-=12.0;
      d3.feet++ ;
     }
 d3.feet+=d1.feet+d2.feet;
 eng (d1);
 eng (d2);
 eng (d3);
 getch();
 }
//определяем функцию  eng, выводящую результат каждый раз в новой строке
void eng (Distance dd)
 {
        cout<<"\n";
        cout<<dd.feet<<"'-"<<dd.inches<<"" ";
```

Оператор *return*

Оператор *return* имеет два варианта использования. Во-первых, этот оператор может использоваться для возврата значения функции (см. примеры выше). Во-вторых, этот оператор вызывает немедленный выход из текущей функции и возврат в вызывающую программу. Продемонстрируем сказанное на примере функции *power ()*, отображающей результат возведения целочисленного значения в положительную целую степень. Если показатель степени окажется отрицательным, инструкция *return* немедленно завершит эту функцию еще до попытки вычислить результат:

```
//power.cpp
#include <iostream.h>                  // подключаем библиотеки потокового
#include <conio.h>                     // и консольного ввода-вывода
```

```
void power (int base, int exp)    // объявляем и определяем функцию power
{
int i;
// если показатель степени отрицательный, оператор return немедленно
// завершает функцию power
if (exp<0) return;
i=1;
for (;exp; exp--) i=base*i; // в теле функции организуем цикл по показателю
степени
cout<<"otvet="<<i;
}
void main ()                                  // точка начала программы
{
clrscr ();
  power (10,2);                       // вызываем функцию и задаем ее аргументы
power (10,-2);
  getch ();
}
```

Область видимости и класс памяти

Изучив основы работы с функциями, рассмотрим два аспекта, касающихся взаимодействия переменных и функций: область видимости и класс памяти.

Область видимости определяет, из каких частей программы возможен доступ к переменной, а класс памяти – время, в течение которого переменная существует в памяти компьютера.

Существуют три типа области видимости: локальная область видимости, область видимости файла, область видимости класса.

Переменные, имеющие локальную область видимости, доступны внутри того блока, в котором они определены. Блоком считается код, заключенный в фигурные скобки. Например, тело функции представляет собой блок.

Переменные, имеющие область видимости файла, доступны из любого места файла, в котором они определены.

Существуют два класса памяти: *automatic* (автоматический) и *static* (статический).

У переменных, имеющих класс памяти automatic, время жизни равно времени жизни функции, внутри которой они определены.

У переменных, имеющих класс памяти static, время жизни равно времени жизни всей программы.

Переменные, определяемые внутри функции, называются локальными, поскольку их область видимости ограничивается этой функцией. Локальные переменные имеют автоматический класс памяти.

Область видимости переменной определяет участки программы, из которых возможен доступ к этой переменной. Это означает, что внутри области видимости переменной операторы могут обращаться к ней по имени и использовать ее значение при вычислении выражений. За пределами области видимости попытки обращения к переменной приведут к выдаче сообщения о том, что переменная неизвестна.

Например, имеется следующая функция:

```
void somerfunc ()
{
 //локальные переменные
 int somervar;
 float othervar;

 somervar=10;              //корректно
 othervar=11;        //корректно
 nextvar=12;         // некорректно, переменная невидима в функции
}
```

Рассмотрим классы памяти локальных переменных.

Локальная переменная не существует в памяти до тех пор, пока не будет вызвана функция, в которой она определена. Это означает, что в памяти нет места, выделенного для их хранения. Когда управление передается в функцию, переменные создаются и под них отводится место в памяти. Затем, когда выполнение функции завершается и управление передается вызывающей программе, переменные уничтожаются, а их значения теряются. Название автоматического класса памяти как раз указывает на то, что переменные автоматически создаются при входе в функцию и автоматически уничтожаются при выходе из нее.

В отличие от локальных переменных, определяемых внутри функции, глобальные переменные определяются вне каких-либо функций. Глобальная переменная видима из всех функций данного

файла, потенциально – из других файлов. По этой причине определения глобальных переменных располагают в начале листинга.

Следующий пример демонстрирует использование глобальной переменной тремя функциями:

```
//global.cpp
#include <iostream.h>
# include <conio.h>
 char ch='a'; //глобальная переменная ch
//прототипы функций
 void getachar ();
 void putachar ();
/* пока не нажата клавиша Enter, вводим символ и он отражается на экране, т.е. переменная ch доступна двум функциям */
 void main ()
 {
   clrscr ();
   while (ch!='\r')
   {
     getachar ();
     putachar ();
     }
}
void getachar ()
{
ch=getch();
}
void putachar ()
{
//cout<<"\n";
cout<<ch;
}
```

Рассмотрим еще один тип переменных, называемый статическими локальными переменными. Существуют и статические глобальные переменные, но они используются в многофайловых программах, которые мы не рассматриваем.

Статическая локальная переменная имеет такую же область видимости, как и автоматическая. Однако время жизни у статической локальной переменной совпадает с временем жизни глобальной переменной, с той разницей, что существование статической локальной

переменной начинается при первом вызове функции, к которой она принадлежит. Далее переменная существует на протяжении выполнения программы.

Статические локальные переменные используются в тех случаях, когда необходимо сохранить значение переменной в памяти после того, как выполнение функции будет завершено.

Следующий пример демонстрирует использование статической локальной переменной:

```
//static.cpp
#include <iostream.h>
#include <conio.h>
float favg (float);
void main ()
{
clrscr ();
float data=1, avg;
while (data !=0)
{
 cout<<"Vvedite chislo:";
 cin>>data;
 avg=favg (data);
 cout<<"\nsrednee znachenie:"<<avg;
 }
 getch ();
 }
 // функция favg () находит среднее арифметическое всех введенных
// значений
 float favg (float newdata)
 {
  //инициализация статических переменных при первом вызове
  static float total=0;
  static int count=0;
  //увеличение счетчика
  count++;
  //добавление нового значения к сумме
  total+=newdata;
  //возврат нового среднего значения
  return total/count;
  }
```

Перегруженные функции

Перегруженная функция выполняет различные действия, зависящие от типов данных, передаваемых ей в качестве аргументов.

Существуют две функции: *starline()*, выводящая на экран линию из 45 символов «*», и *repchar ()*, выводящая заданный символ заданное число раз, используя два аргумента. Создадим третью функцию *charline* (), которая всегда печатает 45 символов, но позволяет задавать печатаемый символ. Все эти функции выполняют схожие действия, но их имена различны. Очевидно, было бы гораздо удобнее использовать одно и то же имя для всех трех функций, несмотря на то что они используют различные наборы аргументов. Программа overload демонстрирует, каким образом это можно осуществить:

```
//overload.cpp
#include <iostream.h>
#include <conio.h>

//прототипы функций
void repchar ();
void repchar (char);
void repchar (char, int);
void main ()
{
clrscr();

//задаем различные наборы аргументов функций
repchar ();
cout<<"\n";
repchar ('=');
cout<<"\n";
repchar ('+',30);
getch ();
}
//функция  выводит 45 раз символ *
void repchar ()
{
for (int j=0; j<45; j++)
cout<<"*";
}
```

```
 //функция выводит 45 заданных символов
void repchar (char ch)
{
for (int j=0; j<45; j++)
cout<<ch;
}
//функция выводит заданный символ заданное количество раз
void repchar (char ch, int n)
{
for (int j=0; j<n; j++)
cout<<ch;
}
```

Практическая работа № 9

1. Цель работы

Приобретение практических навыков в работе с пользовательскими функциями в языке C++.

2. Упражнения

Упражнение 1

Составьте программу вычисления $z = \frac{a^5 + a^{-5}}{2a^m}$, используя перегруженную функцию power из текста лекции, вычисляющую результат возведения целочисленного значения в положительную и отрицательные целые степени.

Упражнение 2

Напишите функцию, вычисляющую факториал заданного целого числа *N*. Используйте эту функцию для вывода на экран факториалов всех целых чисел от 0 до 10.

Упражнение 3

Напишите функцию с именем *hms* (), имеющую три аргумента типа *int*: часы, минуты и секунды. Функция должна возвращать эквивалент переданного ей временного значения в секундах (типа *long*). Создайте программу, которая будет циклически запрашивать у поль-

зователя ввод значения часов, минут и секунд и выводить результат на экран.

2. Вопросы и задания для отчета

1. Определение пользовательской функции. Аргументы, возвращаемое значение и тело функции. Приведите пример.

2. Вызов функции и возврат из нее. Оператор return. Приведите пример.

3. Прототип функции. Общая форма записи и назначение. Приведите пример.

4. Чем аргумент функции отличается от параметра? Если функция использует параметр (аргумент), где он объявляется?

5. Напишите функцию, вычисляющую расстояние между двумя точками на плоскости ($x1$, $y1$) и ($x2$, $y2$). Используйте эту функцию для расчета суммарной длины ломаной линии, заданной набором из трех точек.

6. Напишите функцию, которая при каждом вызове будет выводить на экран количество раз, которое она вызывалась ранее. Напишите программу, которая будет вызывать эту функцию не менее десяти раз. Реализуйте данную функцию двумя различными способами: с использованием глобальной и статической локальной переменных для хранения числа вызовов функции. Какой из способов предпочтительнее? Почему для решения задачи нельзя использовать обычную локальную переменную?

Занятие 9. ОБЪЕКТЫ И КЛАССЫ

Основополагающая идея объектно-ориентированного подхода – объединение данных и действий, производимых над этими данными, в единое целое, которое называется *объектом.*

Функции объекта, называемые в C++ методами, или функциями-членами, обычно предназначены для доступа к данным объекта. Если необходимо считать какие-либо данные объекта, нужно вызвать соответствующий метод, который выполнит считывание и возвратит требуемое значение. Прямой доступ к данным невозможен. Данные сокрыты от внешнего воздействия, что защищает их от случайного из-

менения. Говорят, что данные и методы *инкапсулированы.* Термин инкапсуляция – ключевой в описании объектно-ориентированных языков.

Когда мы говорим об объектах, считается, что они – экземпляры классов. *Класс* – это своего рода форма, определяющая, какие данные и функции будут включены в объект класса. При объявлении класса не создаются никакие его объекты, по аналогии с тем, что существование типа *int* еще не означает существование переменных этого типа.

Таким образом, класс – это описание совокупности сходных между собой объектов.

Определение класса

Перед использованием класс должен быть определен. Определение класса начинается с ключевого слова *class*, за которым следует имя класса. Тело класса заключается в фигурные скобки, после которых ставится точка с запятой (;).

Тело класса может содержать два ключевых слова: *private* и *public*. Ключевая особенность объектно-ориентированного программирования – возможность сокрытия данных. Для этой цели используется ключевое слово *private*. Термин «сокрытие» понимается в том смысле, что данные заключены внутри класса и защищены от несанкционированного доступа функций, расположенных вне класса. Данные, описанные с ключевым словом *public*, напротив, доступны за пределами класса. Зачем это нужно? Сокрытие данных в толковании C++ означает ограждение данных от тех частей программы, которые не имеют необходимости использовать эти данные. В более узком смысле это означает сокрытие данных одного класса от другого. Программисты сами могут создавать средства доступа к закрытым данным, что значительно снижает вероятность случайного или некорректного доступа к ним и позволяют избежать ошибок.

Методы класса

Методы класса – это функции, входящие в состав класса. Если методы описаны с ключевым словом *public*, они доступны за преде-

лами класса. Если методы описаны с ключевым словом *private*, они доступны только в пределах класса.

Определение объектов и вызов методов класса

Определение класса задает вид будущего объекта. Определение объекта подобно определению переменной. Например, определяются два объекта *s*1 и *s*2: *smallobj s*1, *s*2;

Доступ к методам класса возможен только через конкретный объект этого класса, поэтому вызов метода класса имеет вид: объект.метод ();

Операцию точки называют *операцией доступа к члену класса.* Скобки позади имени метода говорят о том, что мы совершаем вызов функции. Иногда вызовы методов объектов называют *сообщениями.*

Следующий листинг программы представляет собой пример создания простого класса:

```
//klass.cpp
#include <iostream.h>
#include <conio.h>
class smallobj                    // определение класса
{
  private:
  int somedata;                   //поле класса
  public:
 void setdata (int d)             //метод класса, изменяющий значение поля
    {
      somedata=d;
    }
 void  showdata()                 //метод класса, отображающий значение поля

  {
     cout<<”znachenie polja ravno”<<somedata;
  }
};
void main ()                  //точка начала программы
{
clrscr();
smallobj s1, s2;              //определяем объекты класса
//вызов методов класса
```

```
s1.setdata (1066);
s2. setdata (1776);
s1. showdata ();
s2.showdata ();
getch ();
}
```

Заметим, что объекты C++ могут выступать в качестве переменных типа, определенного пользователем.

Конструкторы

В предыдущем примере мы инициализировали поле объекта класса, явно вызывая соответствующий метод. Как правило, удобнее инициализировать поля объекта автоматически в момент его создания. Такой способ инициализации реализуется с помощью особого метода класса, называемого *конструктором.*

У конструктора есть несколько особенностей, отличающих его от других методов класса:

1) имя конструктора в точности совпадает с именем класса;

2) у конструкторов не существует возвращаемого значения. Это объясняется тем, что конструктор автоматически вызывается системой, и, следовательно, не существует вызывающей программы или функции, которой конструктор мог бы возвратить значение. Следовательно, отсутствует и тип возвращаемого значения.

Список инициализаций

Одна из наиболее часто возлагаемых на конструктор задач – инициализация полей объекта класса. Рекомендуется оформлять инициализацию следующим образом: она должна располагаться между прототипом метода и телом функции и предваряться двоеточием. Инициализирующее значение помещается в скобках после имени поля. Затем в фигурных скобках – пустое тело. Например: *some (): t (0)*

{пустое тело конструктора}

Если необходимо инициализировать сразу несколько полей класса, то значения разделяются запятыми, и в результате образуется *список инициализаций*. Например: *some ( ): t (0), m2 (33), m3 (45)*

{пустое тело конструктора}

В качестве примера создадим класс, объекты которого могут быть полезны для любой программы. Счетчик – это средство, предназначенное для хранения количественной меры какой-либо изменяющейся величины. Счетчик может хранить число обращений к файлу, число раз, которое пользователь нажал клавишу Enter, или количество положительных чисел в массиве и т.д. Обращение к счетчику происходит, как правило, для того, чтобы узнать текущее значение той величины, для измерения которой он предназначен.

В процедурных языках программирования счетчик был бы представлен в виде глобальной переменной, однако, это не в идеологии объектно-ориентированного программирования. Пример *counter* использует счетчик, значение которого может быть изменено только с помощью его собственных методов. В функции *main ()* создаются два объекта класса *counter* с именами *c*1 и *c*2. В начале на экране выводятся начальные значения *c*1 и *c*2, равные 0. Затем значение счетчика *c*1 инкрементируется один раз, а значение счетчика *c*2 – два раза, и программа вновь заставляет объекты вывести значение своих полей.

```
//counter.cpp
#include <iostream.h>
#include <conio.h>
class counter
{
 private:
  unsigned int count;              //значение счетчика
 public:
  counter (): count (0)            //конструктор
  { }
  void inc_count ()                //инкрементирование счетчика
   {count++;}
  int get_count ()                 //получение значения счетчика
   {return count;}
 };

void main ()
{
clrscr();
//определение объектов класса с инициализацией
counter c1, c2;
cout<<"\nc1="<<c1.get_count();       //вывод начального значения c1
```

```
cout<<"\nc2="<<c2.get_count();        //вывод начального значения c2
//инкрементирование c1 и c2
c1.inc_count ();
c2.inc_count ();
c2.inc_count ();
//вывод
cout<<"\nc1="<<c1.get_count();
cout<<"\nc2="<<c2.get_count();
getch ();
}
```

Практическая работа № 10

1. Цель работы

Приобретение практических навыков в создании классов и объектов языка C++.

2. Упражнения

Упражнение 1

Создайте класс, единственное поле которого должно иметь тип *int*. Создайте методы, которые будут устанавливать значение поля равным 0, инициализировать его целым значением, выводить значение поля на экран и складывать два значения типа *int*. Напишите программу, в которой будут созданы три объекта класса *int*, два из которых будут инициализированы. Сложите два инициализированных объекта, присвойте результат третьему, а затем отобразите результат на экране.

Упражнение 2

Создайте класс, основой для которого послужит структура *part* из темы «Структуры».

Упражнение 3

Внесите изменения в программу упражнения 1. Инициализируйте значение поля класса нулем с помощью конструктора.

2. Вопросы и задания для отчета

1. Для чего необходимо определение класса?

2. Создайте класс *empl,* который должен включать в себя поле типа *int* для хранения номера сотрудника и поле типа *float* для хранения величины его оклада. Методы класса должны позволять пользователю вводить и отображать данные класса. Напишите функцию *main (),* которая запросит пользователя ввести данные для трех сотрудников и выведет полученную информацию на экран.

3. Для чего необходимо использование конструктора в программе?

4. Форма записи инициализации в конструкторе в случае одного поля (нескольких полей).

5. Представьте пункт для взимания платежей за проезд по автостраде. Каждая проезжающая машина должна заплатить за проезд 50 центов, однако, часть машин платит за проезд, а часть проезжает бесплатно. В кассе ведется учет числа проехавших машин и подсчитывается суммарная выручка от платы за проезд.

6. Создайте модель такой кассы с помощью класса, содержащего два поля: одно из них – типа *unsigned int* – предназначено для учета количества проехавших автомобилей, а второе, имеющее тип *double*, будет содержать суммарную выручку от платы за проезд. Конструктор должен инициализировать оба поля нулевыми значениями. Один метод инкрементирует число машин и увеличивает на 0,50 суммарную выручку, другой – увеличивает на единицу число автомобилей, но оставляет без изменения выручку; третий метод выводит оба значения на экран. Там, где это возможно, сделайте методы константными.

Создайте программу, которая продемонстрирует работу класса. Программа должна предложить пользователю нажать одну клавишу для того, чтобы сымитировать заплатившего автолюбителя, и другую клавишу, чтобы сымитировать недобросовестного водителя. Нажатие клавиши *q* (*quit* – выход) должно привести к выдаче текущих значений количества машин, выручки и завершению программы.

Занятие 10. ОДНОМЕРНЫЕ МАССИВЫ

В занятии 2 были рассмотрены типы данных, которые называются базовыми, или встроенными. На основе этих типов данных язык C++

позволяет строить другие типы и структуры данных. Массив – одна из наиболее простых и известных структур данных. Под массивом в языке С++ понимают набор данных одного и того же типа, собранных под одним именем. Каждый элемент массива определяется именем массива и порядковым номером элемента, который называется индексом. Индекс в языке С++ всегда начинается с нуля и является целым числом.

Объявление массива в программе

Как и обычная переменная, перед использованием массив должен быть объявлен. Основная форма объявления массива размерности n следующая: тип данных *имя массива* [размер 1] [размер 2] … [размер n];

При описании одномерного массива объявляющая запись имеет вид: тип *имя массива* [размер];

где тип – базовый тип элементов массива; размер – количество его элементов.

Примеры объявлений:

```
int a[15];                  // массив из 15 целочисленных элементов
                            // с именем a
float x[3];                 // массив из трех элементов вещественного
                            // типа с именем x
```

Одномерный массив может быть представлен в следующим виде (табл. 14):

Таблица 14

Структура одномерного массива

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	…	A[n-2]	A[n-1]
5	21	-3	10	0	15		15	-9

Выбор отдельного элемента одномерного массива осуществляется указанием имени массива и его индекса в квадратных скобках. Например, в результате выполнения команды *cout <<A[0];* на экран будет выведено число 5, являющееся значением первого элемента массива из табл. 14. А командой *A[3]=5.5;* четвертому элементу массива будет присвоено значение 5.5.

Передача массивов в функции

Массивы могут быть использованы как аргументы функций. Рассмотрим в качестве примера программу sale, в которой массив объема продаж передается в функцию, выводящую данные в виде таблицы.

```
//sale.cpp
#include <iostream.h>
#include <conio.h>
//задаем размер массива
const int d=4;
//в объявлении функции массивы-аргументы представлены типом данных
//и их размером
void display (double [d]);
//точка начала программы
void main ()
{
clrscr();
//инициализируем массив
double sales [d]={1432.07, 234.50, 654.01, 322.00};
//вызываем функцию. При вызове функции в качестве аргумента
//используется только имя массива. Это имя в действительности
//представляет собой адрес массива в памяти.
display (sales);
getch ();
}
//функция для вывода на экран массива. Для записи аргумента-массива
//используются тип его данных, имя массива и его размерности.
void display (double sales [d])
{
int m;
cout<<"\nmesaz\n";
cout<<"\t\t1"<<"\t\t 2"<<"\t\t 3"<<"\t\t4\n";
for (m=0; m<d; m++)
  cout<<"\t\t"<<sales [m];
}
```

Массивы структур

Массивы могут содержать в себе не только данные основных типов, но и структуры. Массивы структур – это полезный тип данных,

используемый в различных ситуациях. Можно хранить в массиве структур личные данные сотрудников, географические особенности городов и многое другое.

```
//sale.cpp
#include <iostream.h>
#include <conio.h>
const int Size=4;
struct part
{
  int modelnumber;
  int partnumber;
  float cost;
};
void main ()
{
clrscr ();
int n;
//имя типа part указывает на то, что массив содержит данные более сложного типа
part apart [Size];
//доступ к данным, членам структуры, которые являются элементами массивов
for (n=0; n<Size; n++)
{
   cout<<"\n vvedite nomer modeli: ";
   cin>> apart[n].modelnumber;
   cout<<"\n vvedite nomer chasti: ";
   cin>> apart[n].partnumber;
   cout<<"\n vvedite stoimost: ";
   cin>> apart[n].cost;
 }
 for (n=0; n<Size; n++)
{
   cout<< "Model "<<apart[n].modelnumber;
   cout<< " Chast "<<apart[n].partnumber;
   cout<<" Stoimost "<< apart[n].cost<<"\n";
 }
getch ();
}
```

Массивы как члены классов

Массивы могут быть использованы в качестве полей класса. Рассмотрим пример моделирования компьютерной структуры данных – стека.

```
//stack.cpp
#include <iostream.h>
#include <conio.h>
class Stack
{
  private:
   enum {MAX=10};            // определение переменной MAX,
                             // используемой в классе, внутри класса
        int st [MAX];        //стек в виде массива
        int top;             //вершина стека
public:
Stack ()                     //конструктор
    {top=0;}
void push (int var)          //поместить в стек
   {st[++top]=var;}
int pop ()                   //взять из стека
  {return st [top--];}
};
void main ()
{
      clrscr ();
      Stack s1;
      s1.push(11);
      s1.push (22);
      cout<<"1:"<<s1.pop()<<endl;
      cout<<"2:"<<s1.pop()<<endl;
      s1.push(33);
      s1.push(44);
      s1.push(55);
      s1.push(66);
      cout<<"3:"<<s1.pop()<<endl;
      cout<<"4:"<<s1.pop()<<endl;
      cout<<"5:"<<s1.pop()<<endl;
      cout<<"6:"<<s1.pop()<<endl;
 getch();
 }
```

Практическая работа № 11

1. Цель работы

Приобретение практических навыков в работе с простыми структурными типами данных языка C++.

2. Упражения

Упражнение 1

Наберите код следующей программы и запустите её на выполнение. Объясните алгоритм её работы, смысл использования дополнительной переменной sum в этой программе.

```
//u1.cpp
#include  <conio.h>
#include  <iostream.h>
void main ()
{
   int A[10], i;

   for (i=0; i<10; i++)
    {
        cout << “input  A[“<<i<<”]=”;
        cin>> A[i];
      };
   int sum=0;
   for (i=0; i<10; i++)
        sum=sum+A[i];
   cout << “\nSumma=”<<sum;
getch ();
}
```

Упражнение 2

Наберите код следующей программы и запустите её на выполнение. Объясните алгоритм её работы. Чем он отличается от алгоритма работы программы из упражнения 1?

```
#include  <conio.h>
#include  <iostream.h>
void main ()
```

```
{
   int A[10]={5,4,-8,3,3,0,2,24,56};
int i;

int sum=0;
   for (i=0; i<10; i++)
        sum=sum+A[i];
   cout << "\n Summa="<<sum;
getch ();
}
```

Упражнение 3

Наберите код следующей программы и запустите её на выполнение. Объясните смысл использования переменной *n*.

```
#include <conio.h>
#include <iostream.h>
const int n=10;
void main ()
{
 clrscr ();
 int a[n];
 for (int i=0; i<n; i++)
 {
  cout <<"input a["<<i<<"]";
  cin >>a[i];
  };
  float s=0;
 for (i=0; i<n; i++)
     s=s+a[i];
 s=s/n;
 cout <<"s="<<s;
 getch ();
 }
```

Упражнение 4

Наберите код следующей программы и запустите её на выполнение. Объясните алгоритм её работы. Чем он отличается от алгоритма работы программ из упражнений 1, 2, 3?

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

```
const int n=10;
void main ()
{
 clrscr ();
 int a[n];
 int i;
  for (i=0; i<n; i++)
     a[i]=i*2;
 for (i=0; i<n; i++)
     cout <<" "<<a[i];
getch ();
 }
```

Упражнение 5

Напишите программу, запрашивающую у пользователя *N* вещественных чисел и выводящую на экран сумму чисел с чётными номерами.

Упражнение 6

Оценки, полученные спортсменом в соревнованиях по фигурному катанию (в баллах), хранятся в массиве из 18 элементов. В первых шести элементах записаны оценки по обязательной программе, в седьмом - двенадцатом – по короткой программе, в остальных – по произвольной программе. Выяснить, по какому виду программы спортсмен показал лучший результат.

Упражнение 7

Напишите программу, которая позволяет пользователю вводить целые числа, а затем сохраняет их в массиве типа *int*. Напишите функцию *max* (), которая обрабатывая элементы массива один за другим, находит наибольший. Функция должна принимать в качестве аргумента адрес массива и количество элементов в нем, а возвращать индекс наибольшего элемента. Программа должна вызывать эту функцию, а затем вывести наибольший элемент и его индекс. В случае, если таких элементов несколько, вывести последний.

Упражнение 8

Модернизируйте программу упражнения 4 из занятия, посвященного структурам. Для хранения полных описаний товаров (наименование, стоимость в рублях и в копейках) используйте массив.

3. Вопросы и задания для отчета

1. Что такое одномерный массив? Для чего используются одномерные массивы? Как они описываются?

2. Как в программе использовать значение конкретного элемента одномерного массива?

3. Как называется номер элемента одномерного массива?

4. Как можно заполнить одномерный массив?

5. Дан массив. Напечатать третий, шестой и другие элементы.

6. В массиве хранятся сведения о количестве осадков, выпавших за каждый день июня. Определить:

   а. в какой период выпало больше осадков: в первую половину июня или во вторую;

   б. в какую декаду месяца выпало больше всего осадков.

7. Очередь – это устройство для хранения данных, похожее на стек. Отличие в том, что в стеке последний сохраненный элемент будет первым извлеченным, тогда как в очереди первым извлеченным элементом будет первый сохраненный. Перепишите программу stack.cpp из текста лекции, включив в нее класс *order* так, чтобы она моделировала компьютерную структуру данных – очередь. Класс *order* должен содержать два метода: один – для помещения элемента в очередь, другой – для извлечения элемента из очереди.

Занятие 11. ПРЕОБРАЗОВАНИЕ ОДНОМЕРНЫХ МАССИВОВ

Линейный поиск по условию

Поиск необходимого элемента в большом объеме данных – одна из часто возникающих задач в разнообразных компьютерных приложениях. Для решения этой задачи разработано множество алгоритмов. Выбор того или иного алгоритма поиска зависит в основном от степени структурированности (или внутренней упорядоченности) исходных данных. Если они не структурированы, программисты вынуждены применять малопроизводительные, хотя и простые в реализации методы. Для упорядоченных данных могут использоваться значительно более скоростные и специализированные методы. Здесь будет рассмотрен простейший алгоритм линейного поиска по условию, который может применяться к любым массивам в C++.

Пусть перед нами стоит следующая задача: в массиве данных, все элементы которого различны, требуется найти элемент, удовлетворяющий заданному условию (например, равный 0). Для простоты будем считать, что он существует.

Организуем перебор всех элементов массива, начиная с нулевого. Для каждого элемента будем проверять выполнение заданного условия. В случае, если оно выполнено, запоминаем индекс найденного элемента. При выходе из цикла искомый элемент может быть получен через его индекс.

Решение данной задачи показано в листинге 1:

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

void main ()
{
 clrscr ();
 int a[5];

int k=0;

for (int i=0; i<5; i++)                 // организуем ввод элементов
 {
 cout <<"input a["<<i<<"] ";
 cin >>a[i];
 };

 for (i=0; i<5; i++)                    // организуем перебор всех элементов
  if (a[i]==0) k=i;
 cout <<"N elementa ="<<k;    // вывод на экран номера элемента, равного 0
getch ();
 }
```

Поиск максимального (минимального) элемента

Методы поиска максимального (минимального) элемента в массиве несколько отличаются от метода поиска по условию в силу того, что искомое значение заранее не известно. Модифицируем указанный выше алгоритм. Введём вспомогательную переменную (например, max) и запишем в неё значение нулевого элемента массива. Далее ор-

ганизуем перебор всех оставшихся элементов. Если по ходу встречается элемент со значением, большим max, то записываем его в max, переопределяя таким образом значение этой переменной. После перебора всех элементов в переменной max будет храниться максимальный элемент массива.

Решение данной задачи показано в листинге 2:

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

void main ()
{
 int a[5]={3,5,7,1,0};


 clrscr();
 int i, max=a[0];                    // объявляем целочисленные
                                     // переменные i, max
                                     // инициализируем переменную
                                     // max значением нулевого
                                     // элемента массива
    for (i=1; i<5; i++)              // организуем перебор
                                     // оставшихся элементов массива
   if (a[i]>max) max=a[i];
   cout<<"max="<<max;                // вывод максимального
                                     // элемента на экран
   getch ();
}
```

Сортировка методом пузырька

Под сортировкой будем понимать размещение набора данных в определенном порядке, что используется для облегчения поиска нужного элемента или группы элементов. От эффективности алгоритма сортировки зависит, например, производительность работы баз и банков данных.

Рассмотрим один простейший метод – метод «пузырька», получивший своё название оттого, что продвижение максимальных элементов массива к его вершине происходит постепенно, подобно всплытию пузырька на поверхность воды. Этот метод требует не-

скольких проходов массива, на каждом из которых сравнивается пара соседних друг с другом элементов. Если пара расположена в порядке возрастания, переставляем эти элементы местами. В противном случае элементы остаются на исходных позициях. Процедура должна быть повторена n-1 раз для гарантированного достижения результата. Пример поэтапной работы алгоритма показан на рис. 7. Изображённые на каждом проходе перестановки должны выполняться последовательно слева направо.

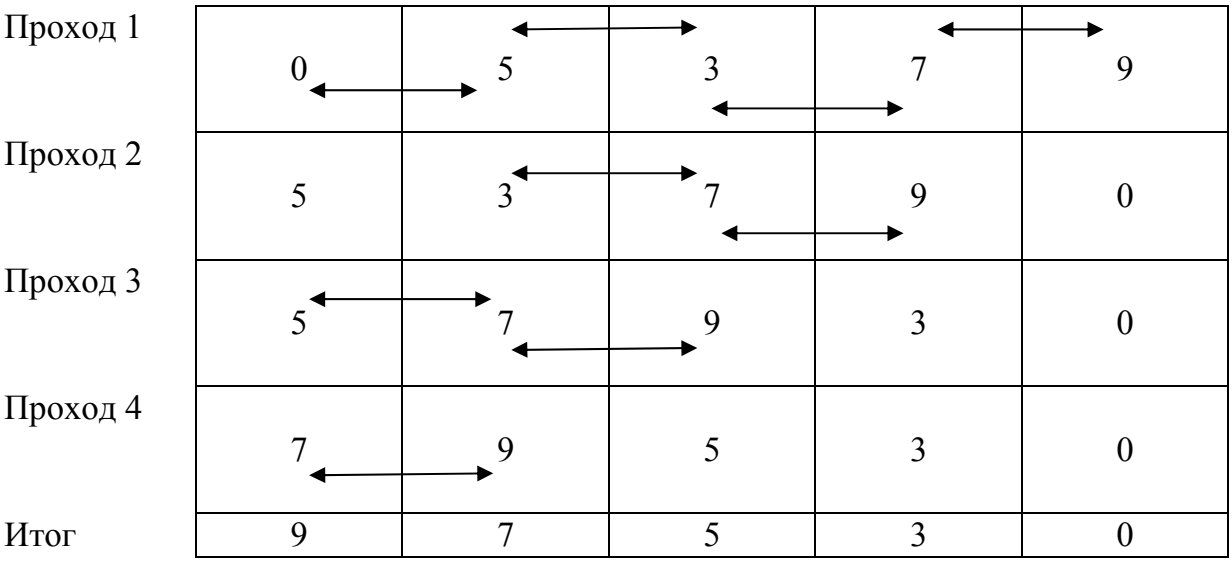

Рис.7

Листинг 3 программы демонстрирует реализацию этого метода:

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

void main ()
{
 int a[5]={0,5,3,7,9};

 clrscr();
 for (int i=1; i<5; i++)              // организуем n-1 проход
 {
  for (int j=0; j<5; j++)
  if (a[j]<a[j+1])                    // если порядок неправильный,
       {                              // запоминаем a[j] в дополнительной
         int temp=a[j];               // переменной temp
        a[j]=a[j+1];                  // меняем местами a[j] и a[j+1]
```

```
          a[j+1]=temp;
         }
 }
 for (i=0; i<5; i++)                    // выводим на экран отсортированный
    cout<<a[i]<<" ";                    // массив
 getch ();
}
```

Практическая работа № 12

1. Цель работы

Приобретение практических навыков в работе с одномерными массивами в языке C++.

2. Упражнения

Упражнение 1

Отредактируйте программу в листинге 2 текста занятия для случая поиска минимального элемента.

Упражнение 2

Проанализируйте представленные фрагменты кода программы. Сформулируйте, какое действие выполняет каждый фрагмент. Предполагается, что в каждом из примеров массив инициализирован некоторым набором значений (на месте многоточия «…»).

а)
```
int a[10]={…}, s=0;
for (int i=0; i<10; i+=2)
s+=a[i]
```

б)
```
int M[25]={…};
for (int k=0; k<25; k++)
    if (M[k]==5)
        cout<<”!”;
```

в)
```
float x[50]={…};
for (int n=0; n<50; n++)
   if ((0<x[n] && (x[n]<1))
       cout<<x[n];
```

г)
```
float y[50]={…};
for (int m=0; m<49; m++)
   y[m]=y[m+1];
```

Упражнение 3

Определите, что будет выведено на экран в результате выполнения приведенных фрагментов кода. Объясните работу этих конструкций пошагово.

а)
```
int c[3]={1,2,3};
for (int i=0; i<3 i++);
c[0]+=c[i];
cout<<c[0];
```

б)
```
int c[3]={1,2,3};
for (int i=0; i<3; i++)
    c[i]+=c[2-i];
cout<< c[0]+c[1]+c[3];
```

в)
```
int i[3]={3,2,1};
for (int j=0; j<3; j+=2)
    cout<<i[[j];
```

г)
```
int S[3]={1}, f=1;
for (int i=0; i<3; i++)
    f+=S[i];
cout<<f;
```

Упражнение 4

Дан набор из N вещественных чисел X_k. Напишите программу, рассчитывающую количество элементов, попадающих на отрезок [AB], то есть удовлетворяющих условию $A \le X_k \le B$, где A и B – заданные вещественные числа.

Упражнение 5

Дан набор из N целых чисел. Напишите программу, определяющую, в какой порядковой позиции P находится элемент с заданным значением F. Значение P вывести на экран. Исходный массив и F вводятся с клавиатуры. Варианты задания:

а) найти первый по порядку элемент;

б) найти последний по порядку элемент.

Упражнение 6

Медианой упорядоченного набора чисел x_1, x_2, …, x_n называется его «середина», то есть значение среднего элемента. Напишите программу, вычисляющую медиану неупорядоченной последовательности N чисел.

Упражнение 7

Заданный массив A сдвинуть циклически на m позиций вправо. При циклическом сдвиге вправо выталкиваемые элементы с конца массива заполняют освобождающиеся места в начале массива. Например, при сдвиге вправо на три разряда массива A (1, 2, 3, 4, 5, 6, 7) получаем массив A (5, 6, 7, 1, 2, 3, 4).

Упражнение 8

Пусть дан массив из пяти целых чисел. Отсортировать его по возрастанию методом прямого выбора (перебором). Алгоритм сортировки этим методом выглядит следующим образом:

1) просмотреть массив, начиная от первого элемента, найти минимальный элемент и поместить его на место первого элемента, а первый - на место минимального;

2) просмотреть массив, начиная от второго элемента, найти минимальный элемент и поместить его на место второго элемента, а второй – на место минимального и т.д., пока не будет отсортирован весь массив.

3. Вопросы и задания для отчета

1. Общий алгоритм работы линейного поиска по условию, пример его работы.

2. Общий алгоритм поиска максимального и минимального элементов, пример его работы.

3. Общий алгоритм сортировки методом «прямого выбора», пример пошаговой реализации.

4. Общий алгоритм сортировки методом «пузырька», пример пошаговой реализации.

5. Объясните, почему при поиске максимального и минимального элементов в примере переменным min и max первоначально присвоены значения A[0]? Можно ли для этой цели использовать другие элементы массива? Чем должен определяться выбор начальных значений min и max?

6. В массиве хранится информация о количестве побед, одержанных двадцатью футбольными командами. Определить номера команд, имеющих меньше трех побед.

7. Пусть дан массив из пяти целых чисел. Отсортировать его по возрастанию методом Шелла. Основная идея алгоритма заключается в том, чтобы вначале устранить массовый беспорядок в массиве, сравнивая далеко отстоящие друг от друга элементы, и постепенно уменьшить интервал до единицы. Это означает, что на последних шагах сортировка сводится просто к перестановке соседних элементов (если, конечно, такие перестановки необходимы).

ОТВЕТЫ И РЕШЕНИЯ

Практическая работа № 2

Упражнение

а) в записи *#include <iostream>* отсутствует «*h*»;

б) в записи *#include <iostream>* отсутствует «*h*»; в первой, второй и третьей строках необходимо убрать «;»;

в) отсутствует последняя строка программы };

г) в первой строке вместо *<stdio.h>* должна быть указана библиотека *<iostream.h>*;

д) у оператора *cout* отсутствует <<;

е) отсутствует директива *<conio.h>*, у оператора *cout* отсутствует <<, у функции *getch* отсутствуют ();

Вопросы и задания для отчета

1. Комментарии служат для пояснения текста программы. Они начинаются с двойной косой черты и заканчиваются концом строки, могут начинаться как в начале строки, так и после любого оператора. Они должны быть хорошо составлены, иметь правильную пунктуацию, по возможности без сокращений, и выровнены вертикально. Рекомендуется комментировать фрагменты программы (логические блоки кода, циклы и т.п.), а не каждый отдельный оператор.

2. Любая С-программа начинается с директив препроцессора, включающих в код программы заголовочные файлы необходимых программных библиотек. Вслед за ними необходимо определить функцию *main*. Следующая строка должна содержать открывающуюся фигурную скобку, означающую начало тела функции *main*, состоящего из набора объявлений, определений и операторов. Каждое из них должно завершаться символом точки с запятой. Последняя строка программы содержит закрывающуюся фигурную скобку, которая обозначает конец функции *main*.

3. См. табл. 1, 2.

4. Функция *main* обязательна в любой программе на языке С++, так как служит точкой начала программы. Выполнение программы начинается с первого оператора внутри *main* и продолжается последовательно шаг за шагом до тех пор, пока управление не достигнет последнего оператора внутри функции *main*.

Практическая работа № 3

Упражнение 1

Недопустимы имена переменных под буквами г), д).

Упражнение 2

Переменная *name* используется для хранения имени, переменная *cm* – для хранения значения роста человека, в переменной *inch* рост человека дан в дюймах.

Упражнение 3

```
//u3.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
 clrscr ();
 int a;
 float b;
 double c;
 cout<<"vvedite a,b\n";
 cin>>a>>b;
 c = a*b;
 cout<<"\n c = "<<c;
 c = a/b;
 cout<<"\n c = "<<c;
getch ();
}
```

Упражнение 4

/* u4.cpp

воспользуемся теоремой косинусов, которую последовательно применим для нахождения всех углов треугольника. Рассчитанные значения углов переведем из радианов в градусы, умножив их на */ $180^0/\pi$.

```
//---------------------------------------------------------------------------
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main()
{
```

```
clrscr ();
float a,b,c;
cout<<"vvedite a,b,c\n";
cin>>a>>b>>c;
float alfa,betta,gamma;
alfa = acos ((b*b + c*c – a*a)/(2*b*c));
betta = acos((a*a + c*c – b*b)/(2*a*c));
gamma=acos((a*a + b*b – c*c)/(2*a*b));
cout<<"\n alfa = "<<alfa/M_PI*180;
cout<<"\n betta = "<<betta/M_PI*180;
cout<<"\n gamma = "<<gamma/M_PI*180;
getch ();
}
```

Упражнение 5

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
float a, b, c;
//вычисление уменьшаемого
a = (13.72 + cos(60*M_PI/180))*(7.58 + sin(60*M_PI/180))/M_LN(10);
//вычисление вычитаемого
b = sqrt (17.51/cos(30*M_PI/180));
//вычисление разности
c = a - b;
cout<<"c ="<<c;
getch ();
}
```

Вопросы и задания для отчета

1. Директива *#include < iostream.h >* подключает библиотеку *iostream.h* / – стандартную библиотеку потокового ввода-вывода. Аналогичная запись *#include <conio.h>* подключает библиотеку консольного ввода-вывода (понятие консоль объединяет комплект устройств интерактивного ввода-вывода, подсоединенных непосредственно к компьютеру: монитор, клавиатуру, мышь и т.д.).

2. Для объявления данных целочисленного типа используется ключевое слово *int*, для вещественного – *double* или *float*, символьного – *char*. Текстовая строка – *char* <имя переменной> [*n*], где имя переменной – имя текстовой строки, *n* - количество символов в строке.

3. Да. Числа без знака – от 0 до 255. Числа со знаком от – 128 до 127.

4. Синтаксические и логические ошибки:

а) int a, b;
cout << b;
cin >>a;

б) float x = - 0.5;
float y = fabs (x);

в) float x, y, z;
x = 1.5;

г) float i = 2*exp(x);

д) float x, y, z;
z=sin (x);
z = pow (x, y)

е) float a = 1 ;
float b = cos (a);

5. Возможный вариант программы:

```
//u5.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
int a, b, c, d;
cout<<"vvedite chislo";
cin>>a;
/* ---------------------------------------------------------------------------------------
последовательно вычисляем число единиц, число десятков,
число сотен
---------------------------------------------------------------------------------------*/
b = a  %  10;
a =a / 10;
c = a % 10;
a = a / 10;
d = a * b * c;
/*---------------------------------------------------------------------------------------
вывод результата на экран
---------------------------------------------------------------------------------------*/
cout <<"\n"<<d;
getch ();
}
```

6. Программа выглядит следующим образом:

```
//u6.cpp
#include <iostream.h>
```

```
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
float x, y;
cout<<"vvedite x\n";
cin>>x;
y=sqrt ((12*x + pow(x, 4)) /( pow (x,2)+sqrt (234+z)));
cout<<"y(x) = "<<y;
getch ();
}
```

Практическая работа № 4

Упражнение 1

а) x = x

б) 2.5 = 2.5

в) 3

г) *z* = 6.000000

Упражнение 2

1. Вывод на экран символов «*» средствами консольного ввода-вывода:

```
//---------------------------------------------------------------------------
//u2.cpp
#include <conio.h>
#include <stdio.h>
void main()
{
  clrscr ();
  printf("      *\n");
  printf("     * *\n");
  printf("    *   *\n");
  printf("     * *\n");
  printf("      *");
  getch ();
}
```

2. Вывод на экран символов «*» средствами потокового ввода-вывода:

```
//---------------------------------------------------------------------------
//u2.cpp
```

```
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr();
 cout<<"      *\n";
 cout<<"     * *\n";
 cout<<"    *   *\n";
 cout<<"     * *\n";
 cout<<"      *";
 getch ();
}
```

Упражнение 3

```
//u3.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
cout<<"Familia\t"<<"Imja\t"<<"Adress\t\t"<<"Gorod\n";
cout<<"-----------------------------------------------------\n";
cout<<"Petrov\t"<<"Vasilij\t"<<"Klenovaja, 16\t"<<"Vladimir\n";
cout<<"Ivanov\t"<<"Sergej\t"<<"Vasilisina, 6\t"<<"Vladimir\n";
cout<<"Sidorov\t"<<"Ivan\t"<<"Berezovaja, 26\t"<<"Vladimir\n";
getch();
}
```

Вопросы и задания для отчета

3. а) $x = 2.5$ б) 7.5! в) 15

4. Варианты вывода:

1. Вывод на экран символов «*» средствами консольного ввода-вывода:

```
//u4.cpp
#include <conio.h>
#include <stdio.h>
void main()
{
 clrscr ();
 printf(" *********\n");
 printf(" *  *     *  *\n");
```

```
  printf(" *     *      *\n");
  printf(" *********\n");
   getch ();
}
```

2. Вывод на экран символов «*» средствами потокового ввода-вывода

```
#include <iostream.h>
#include <conio.h>
void main()
{
  clrscr();
  cout<<" *********\n";
  cout<<" *  *     *  *\n";
  cout<<" *     *      *\n";
  cout<<" *********\n";
  getch ();
}
```

Практическая работа № 5

Упражнение 1

```
int a, b;
a=2; b=1;
a++;          // a = 2 + 1 = 3
a = 2;        // a = 2
++a;          // a = a + 1 = 3
a = 2;        // a = 2
b = a++;      // b = 2, a = 3
a = 2;        // a = 2
b = ++a;        // a = 3, b = 3
```

Упражнение 2

$i = 4$

Упражнение 3

Выражение можно записать следующим образом:

$a = b++ + c++ + ++d;$

$a = 5 + 7 + 10 = 22.$

Значения переменных b и c увеличатся на 1 только после выполнения фрагмента кода. Таким образом: $a = 22$, $b = 6$, $c = 8$, $d = 10$.

Упражнение 4

Выражение можно записать следующим образом: $k = k+ \ ++k$, $k = = 2 + 2 = 4$.

Упражнение 5

```
// u5. cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
float h,b,c,a;
cout<<"vvedite dlinu osnovania a\n";
cin>>a;
cout<<"vvedite h\n";
cin>>h;
a/ = 2;
b=sqrt (pow(h, 2)+pow(a, 2));
cout<<"b = "<<b;
cout<<"c = "<<b;
getch ();
}
```

Упражнение 6

```
//u6.cpp
#include <conio.h>
#include <iostream.h>
#include <math.h>
void main ()
{
clrscr ();
float s, p, a, b, c;
cout<<"vvedite a, b, c\n";
cin>>a>>b>>c;
p = (a + b + c)/2;
s= sqrt (p*(p - a)*(p - b)*(p - c));
cout<<s;
getch ();
}
```

Вопросы и задания для отчета

1. а) $x = 11$, y = 11; б) $y = 10$, $x = 11$.
2. а) $x = 90$; б) $tl = 9$.
3. Площадь треугольника:

```
//u3.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
  clrscr ();
  float a, h, s;
  cout<<"vvedite a, h\n";
  cin>>a>>h;
  s=a * h / 2;
  cout<<"S="<<s;
getch ();
}
```

4. Программа, вычисляющая размер регулярных платежей по займу на покупку автомобиля:

```
//u4.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
double a, pr, pl, s, e, m, d;
int n, z;
cout<<"vvedite ishodnuju summu zaima\n";
cin>>a;
cout<<"vvedite prozentnuju stavku\n";
cin>>pr;
cout<<"vvedite kolichestvo viplat v god\n";
cin>>z;
cout<<"vvedite srok zaima";
cin>>n;
d = (d / 12) / 100;
s=pr*a/z;
e= - (z * n);
m= (pr / z) + 1;
d= 1 – pow (m, e);
```

```
pl=s / d;
cout<<"razmer platega po zaimu "<<pl<<" rub";
getch ();
}
```

5. Заданная программа выглядит следующим образом:

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
float r, d, l, s;
cout<<"vvedite r\n";
cin>>r;
d=2 * r;
l=2 * M_PI * r;
s=2 * M_PI * pow (r, 2);
cout<<"\n d="<<d;
cout<<"\n l="<<l;
cout<<"\n s="<<s;
getch ();
}
```

Практическая работа № 6

Упражнение 1

```
//u1.cpp
#include <iostream.h>
#include <conio.h>
 void main ()
 {
  clrscr ();
  int a,b,c;
  cout<<"vvedite a, b, c\n";
  cin>>a>>b>>c;
  if (a==b)
    {
     if (b==c)
    cout<<"a,b,c ravni\n";
    }
  else cout<<"ne ravni";
```

```
 getch ();
 }
```

Упражнение 2

```
//u2.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
clrscr ();
float a, b, c;
cout<<"vvedite dlini storon treugolnika \n";
cin>>a>>b>>c;
if ((a<b+c) && (b<a+c) && (c<a+b))
  cout<<"treugolnik sushestvuet";
else
   cout<<"treugolnik ne sushestvuet";
getch ();
}
```

Упражнение 3

```
//u3.cpp
# include <iostream.h>
# include <conio.h>
#include <math.h>

void main ()
{
clrscr ();
float a, b, c, d, x1, x2;
cout <<"vvedite a b c\n";
cin >>a>>b>>c;
d = b * b – 4 * a * c;
if (d>0)
        {
          x1=(-b + sqrt(d)) / (2 * a);
          x2=(-b - sqrt(d)) / (2 * a);
          cout <<x1<<x2;
         }
else
```

```
    {
      if (d==0)
        {
          x1=-b/(2*a);
            cout <<x1;
         }
       else
            cout <<"kornej net";
    };
getch();
}
```

Упражнение 4

```
//u4.cpp
# include <iostream.h>
# include <conio.h>
void main ()
{
clrscr ();
int n, x, r;
char*  s;
cout<<”vvedite chislo n\n”;
cin>>n;
x=n / 10;
r=n % 10;
if  ((n==1) || (x>1) && (r==1))
  s=”god”;
else
     {
       if ((n>1) && (n<5) || ((x>1) && (x<5) && (r>1) && (r<5)))
         s=”goda”;
      };
if (r>=5)
  s=”let”;
cout<<”mne ”<<n<<”  “<<s;
getch ();
}
```

Упражнение 5

```
//u5.cpp
#include <iostream.h>
```

```
#include <conio.h>
void main ()
{
clrscr ();
int x=10, y=10;
cout<<"vvedite m" ;
cin>>m;
switch (m)
{ case 1:
          cout<<"piki";
          break;
  case 2: cout<<"trefi";
           break;
  case 3: cout<<"bubni";
           break;
  case 4: cout<<"chervi";
           break;
}
getch ();
}
```

Вопросы и задания для отчета

1. а) ошибка в записи 5 = *b*; необходимо записать *b* = 5;

б) возможны два варианта : после условия и служебного слова *else* необходимо вместо условия записать операторы, выполняющиеся в случае истинности (ложности) условия, или операторы условия могут быть вложены друг в друга. В этом случае необходимо после условия и служебного слова *else* записать другие операторы *if*;

в) необходимо символ закрывающейся фигурной скобки записать после выражения *a = sqrt* (*x*∗*x*);

г) неправильно записано условие. Верна следующая запись: *if*((*b*<5) && (*b* > *c*)).

2. а) 1 5 10 б) ОК! в) 1 1 10 г) <<<
>>>

3. Возможный вариант программы:

```
//u3.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
```

```
{
int x,y,a,r,s;
clrscr ();
cout<<”vvedite x\n”;
cin>>x;
y=pow (x, 2);
A=x /10;
r=x % 10;
s=4*(pow(a, 3) + pow(r, 3));
If (y==s)
  cout<<”pologitelno”;
else
  cout<<”otricatelno”;
getch();
}
```

4. Возможный вариант программы:

```
//u4.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
 clrscr ();
 float x, y;
 cout<<”vvedite koordinati x,y\n”;
 cin>>x>>y;
   if  ((x>5) && (y>1))
       cout<<”tochka popadaet”;
   else
       cout<<”tochka ne popadaet”;
getch ();
}
```

5. Программа будет выглядеть следующим образом:

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
 clrscr ();
```

```
  float x, y;
  cout<<”vvedite dva chisla x,y\n”;
  cin>>x>>y;
  if (sqrt(y) < sqrt(x))
     y=y*5;
  cout<<”\n x=”<<x<<” y=”<<y;
  getch ();
}
```

6. Програама для вычисления значения функции Z(a)

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
  clrscr ();
  float a, z;
  cout<<”vvedite a\n”;
  cin>>a;
  if (a>0)
     z=1;
  else
     if (a==0)
        z=0;
     else
       if (a<0)
          z =-1;
  cout<<”z=”<<z;
getch ();
}
```

7. Вид треуголиника:

```
//u7.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main ()
{
  clrscr ();
  float a, b, c, d;
  cout<<”vvedite a, b, c\n”;
```

```
 cin>>a>>b>>c;
 if ((a<b+c) && (b<a+c) && (c<a+b))
    {
     if (pow (c, 2)==pow(b, 2) + pow(b, 2))
       cout<<”treugolnik prajmougolnij”;
     else
         {
           d=(pow(c,2) + pow(b,2) – pow(a,2)) / (2*c*b);
           if ((d>0) && (d<1))
              cout<<”ugol ostrij”;
           else
              cout<<”ugol tupoj”;
          };
     }
else
  cout<<”treugolnik ne sushestvuet”;
getch ();
}
```

Практическая работа № 7

Упражнение 1

```
//u1.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
int s=0;
for (int i=8; i<21; i++)
s=s+i;
cout<<"summa="<<s;
getch ();
}
```

Упражнение 2

```
//u2.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
```

```
int i, k;
float x, s;
cout<<"vvedite chislo x\n";
cin>>x;
cout<<"vvedite stepen k\n";
cin>>k;
s=x;
for (i=2; i<k+1; i++)
x=x*s;
cout<<"stepen chisla x="<<x;
getch ();
}
```

Упражнение 3

```
//u3.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
int i, k, s;
s=1;
cout<<"vvedite chislo\n";
cin>>k;
if (k==0)
   s=1;
else
{
  for (i=1; i<k+1; i++)
      s=s*i;
}
cout<<"faktorial chisla= "<<s;
getch ();
}
```

Упражнение 4

```
//u4.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
```

```
int i, n, s;
s=0;
cout<<"vvedite chislo\n";
cin>>n;
for (i=1; i<n+1; i++)
if ((i%2==0) && (n%i==0))
   s=s+i;
cout<<"summa chetnih delitelej chisla= "<<s;
getch ();
}
```

Упражнение 5

Значения счетчиков внешнего и внутреннего циклов изменяются следующим образом:

i=1
j=1, j=2
i=2
j=1, j=2
i=3
j=1, j=2
i=4
j=1, j=2

Упражнение 6

Введем следующие обозначения: *s*1 - общая сумма выплат за квартал всем работникам, s2 - зарплата, полученная за квартал каждым работником, *s*4, *s*5, *s*6 - зарплаты, полученные за квартал каждым работником.

```
//u5.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
int i, n, m, s1=0, s2, s4=0, s5=0, s6=0;
for (i=1; i<13; i++)
{
 cout <<"vvedite zarplatu "<<i<<" rabotnika za kvartal \n";
 s2=0;
 for (int j=1; j<4; j++)
  {
```

```
      cin>>n;
      s2=s2+n;
      if (j==1) s4=s4+n;
      if (j==2) s5=s5+n;
      if (j==3) s6=s6+n;
    }
   s1=s1+s2;
  cout<<"zarplata "<<i<<" rabotnika za kvartal="<<s2<<" rub\n";
}
cout<<"obshaja summa="<<s1;
cout<<"\n zarplata za 1 kvartal vseh rabotnikov "<<s4;
cout<<"\n zarplata za 2 kvartal vseh rabotnikov "<<s5;
cout<<"\n zarplata za 3 kvartal vseh rabotnikov "<<s6;
getch ();
}
```

Упражнение 7

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
void main ()
{
clrscr ();
int pow=1;
int numb=1;
while (pow<10000)
{
cout<<setw(2)<<numb;
cout<<setw(5)<<pow<<"\n";
++numb;
pow=numb*numb*numb*numb;
}
getch();
}
```

Упражнение 8

```
//u7.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
```

```
float a, b, c;
char ch;
do
{
clrscr ();
cout<<"vvedite 1 chislo\n";
cin>>a;
cout<<"vvedite 2 chislo\n";
cin>>b;
cout<<"vvedite znak operazii\n";
cin>>ch;
if (ch=='+') c= a + b;
if (ch=='-') c= a - b;
if (ch=='*') c= a * b;
if (ch=='/') c= a / b;
cout<<"otvet ="<<c<<"\n";
cout<<" (y/n) \n";
cin>>ch;
}
while (ch !='n');
}
```

Вопросы и задания для отчета

1. а) так как *i* и *j* объявлены как переменные целого типа, ошибочным является шаг цикла 0.1, переменная *j* не может быть аргументом функций sin и exp;

б) символ открывающейся скобки необходимо поставить после блока <инициализация>, символ “точка с запятой” после блока <инициализация> не ставится.

2. а) 1 2 3 4; б) 5 5 5; в) $$$
###
###

3. Программа, вычисляющая суму ряда $S=\sum_{k}^{10}\frac{k^2}{2^k}$:

```
//u3.cpp
#include <conio.h>
#include <iostream.h>
#include <math.h>
void main ()
{
clrscr ();
```

```
int k;
float s=0;
for (k=0; k<11; k++)
  {
   s=s + (pow (k, 2) / pow (2, k));
   cout<<"\n"<<s;
  }
      getch ();
      }
```

4. Возможный вариант программы:

```
//u4.cpp
#include <conio.h>
#include <iostream.h>
void main ()
{
clrscr ();
int i,n;
float s1,s2,s3=0,s4=0,s5=0;
for (i=1; i<16; i++)
  {
    cout <<"vvedite balli "<<i<<" sportsmena po vidam programm\n";
    s1=0;
    s2=0;
 for (int j=1; j<4; j++)
   {
     cin>>n;
     s1=s1+n;
     if (j==1) s3=s3+n;
     if (j==2) s4=s4+n;
     if (j==3) s5=s5+n;
    }
 s2 = s1 / 3;
cout<<"srednee kolichestvo ballov "<<i<<" sportsmena za program-
mu="<<s2<<"\n";
  }
s3 = s3 / 15;
cout<<"srednee kolichestvo ballov za objazatelnju programmu "<<s3;
s4=s4 / 15;
cout<<"\n srednee kolichestvo ballov za korotkuju programmu "<<s4;
s5 = s5 / 15;
```

```
cout<<"\n srednee kolichestvo ballov za proizvolnuju programmu "<<s5;
getch ();
}
```

5. Возможный вариант программы:

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
#include <math.h>
void main()
{
clrscr ();
int i, n, a, min=1, max=1, s;
cout<<"vvedite kolichestvo chisel, chisla\n";
cin>>n;
for (i=1; i<n+1; i++)
{
 cin>>a;
 if (i % 2==0)
    if (min>=a) min=a;
 if (i %2 !=0)
    if (max<=a)  max=a;
}
s=min + max;
cout<<"s="<<s;
getch ();
}
```

6. Возможный вариант программы:

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
clrscr ();
int k,s;
do
{
  cout<<"\n vvedite chislo\n";
  cin>>k;
  s=1;
  for (int i=1; i<=k; i++)
```

```
        s=s*i;
cout<<"faktorial="<<s;
}
while (k !=0);
}
```

Практическая работа № 8

Упражнение 2

```
//u2.cpp
#include <iostream.h>
#include <conio.h>
struct phone
{
  int area;                          //код региона (3 цифры)
  int exchange;                      //номер АТС (3 цифры)
  int number;                        //номер абонента (4 цифры)
  };
void main ()
{
clrscr();
phone ph1={212,767,8900};            //инициализация номера
phone ph2;                           //определение номера

//получение данных от пользователя
// вводим полный номер без начальных нулей, так как задан тип
// поля integer
cout<<"vvedite polnij nomer \n";
cin>>ph2.area>>ph2.exchange>>ph2.number;

//вывод номеров
cout<<"\n moj nomer:"<<" ("<<ph1.area<<")";
cout<<ph1.exchange<<"-"<<ph1.number;

cout<<"\n Vash nomer:";
cout<<"("<<ph2.area<<")"<<ph2.exchange<<"-"<<ph2.number;
getch();
}
```

Упражнение 3

```
//time.cpp
#include <iostream.h>
#include <conio.h>
```

```
//объявление структуры
struct time
{
 int hours, minutes, second;
 long totalsecs;
};

void main ()
{
 clrscr ();
 //определение структурной переменной
 time t1;
 cout<<"\nvvedite chislo chasov "; cin>>t1.hours;
 cout<<"\nvvedite chislo minut "; cin>>t1.minutes;
 cout<<"\nvvedite chislo secund ";  cin>>t1.second;

//вычисление количества секунд
t1.totalsecs=t1.hours*3600+t1.minutes*60+t1.second;
cout<<"vremja v sec: "<<t1.totalsecs;
 getch();
 }
```

Упражнение 4

```
//u4.cpp
#include <iostream.h>
#include <conio.h>
struct tovar
{
 char tovarname [15];
 int rubl;
 int kop;
 };

 void main ()
 {
   clrscr ();
   tovar t1={"disk",12,37};
   tovar t2={"doll",623,32};
   tovar t3={"cup",45,78};
 if (t1.rubl>t2.rubl)
    cout<<t1.tovarname<<" doroge "<<t2.tovarname<<"\n";
```

```
 else cout<<t2.tovarname<<" doroge "<<t1.tovarname<<"\n";
 if (t1.rubl>t3.rubl)
   cout<<t1.tovarname<<" doroge "<<t3.tovarname<<"\n";
 else cout<<t3.tovarname<<" doroge "<<t1.tovarname<<"\n";
 if (t2.rubl>t3.rubl)
   cout<<t2.tovarname<<" doroge "<<t3.tovarname<<"\n";
 else cout<<t3.tovarname<<" doroge "<<t2.tovarname<<"\n";
    getch ();
    }
```

Вопросы и задания для отчета

4. Программа, в которой используется структура *point* для интерпретации точки на плоскости:

```
//u4.cpp
#include <iostream.h>
    #include <conio.h>
    struct point
{
    int xCo;                    //координата x
    int yCo;                    //координата y
 };
void main ()
{
  point p1, p2, p3;                    //определяем три точки

//ввести координаты двух точек
cout<<"\n vvedite koordinati p1: ";
cin>>p1.xCo>>p1.yCo;
cout<<"\n vvedite koordinati p2: ";
cin>>p2.xCo>>p2.yCo;

//вычислить сумму координат
p3.xCo= p1.xCo+ p2.xCo;
p3.yCo= p1.yCo+ p2.yCo;

//вывести сумму координат
cout<<"koordinati p1+p2: "<<p3.xCo<<","<<p3.yCo;
getch ();
}
```

5. Возможный вариант программы:

```
//gorod.cpp
#include <iostream.h>
#include <conio.h>
struct gorod
{
  char g[10];
  long int ch;
};
void main ()
{
clrscr();
gorod g1={"b",100000};
gorod g2={"v", 120000};
gorod g3={"S",130000};

if (g1.ch>100000)
  cout<<"\n"<<g1.g;
if (g2.ch>100000)
  cout<<"\n"<<g2.g;
if (g3.ch>100000)
  cout<<"\n"<<g3.g;
getch();
}
```

Практическая работа № 9

Упражнение1

```
//poweroverload.cpp
#include <iostream.h>
#include <conio.h>

//функция вычисляет пятую степень заданного числа
float power (float a)
{
  int i=1, exp=5;
    for (; exp; exp--)
      i=a*i;
    return i;
}
```

```
//функция вычисляет заданную степень заданного числа
float power (float a, int m)
{
  int i=1;
    for (; m; m--)
      i=a*i;
    return i;
}
void main()
{
clrscr ();
float a;
int m;
cout<<"vvedite a, m\n";
cin>>a>>m;
float z=(power (a) + 1/power (a))/(2 * power (a, m));
cout<<"z="<<z;
getch ();
}
```

Упражнение 2

```
#include <iostream.h>
#include <conio.h>
int factorial (int);          //прототип функции factorial ()
void main()
{
clrscr ();
const int N=10;
for (int i=0; i<=N; i++)
  cout<<i<<"!="<<factorial (i)<<"\n";
getch ();
}

//определение функции factorial ()
int factorial (int n)
{
  int s=1;
```

```
  for (int i=1; i<=n; i++)
     s*=i;
  return s;
}
```

Упражнение 3

```
//timefunc.cpp
#include <iostream.h>
#include <conio.h>

long hms (int, int, int);          //прототип функции

void main ()
{
 clrscr ();

 int hours, minutes, second;
 char ch='y';
//организуем циклический запрос у пользователя ввода значений
do
 {
 cout<<"\nvvedite chislo chasov "; cin>>hours;
 cout<<"\nvvedite chislo minut "; cin>>minutes;
 cout<<"\nvvedite chislo secund ";  cin>>second;
 hms (hours, minutes, second);
 cout<<"\ny/n "; cin>>ch;
 }
  while (ch !='n');

 }
//определение функции hms ()
long hms (int hours, int minutes, int second)
{
long totalsecs=hours*3600+minutes*60+second;
cout<<"vremja v sec: "<<totalsecs;
}
```

Вопросы и задания для отчета

5. Программа будет выглядеть следующим образом:

```
//u5.cpp
```

```
#include <iostream.h>
#include <conio.h>
#include <math.h>
float d (float, float, float, float);
 void main ()
 {
   clrscr ();
   float x1, y1, x2, y2, x3, y3;
   cout<<"vvedite x1, y1, x2, y2, x3, y3";
   cin>>x1>>y1>>x2>>y2>>x3>>y3;
   float s1 = d(x1, y1, x2, y2);
   float s2 = d(x2, y2, x3, y3);

   float s = s1 + s2;
   cout<<"s="<<s;
     getch ();
 }
float d (float x1, float y1, float x2, float y2)
{
 float r = sqrt (pow ((x2 - x1),2)+pow((y2 - y1),2));
 return r;
}
```

6. Программа будет выглядеть следующим образом:

а) при использовании глобальной переменной:

```
//global.cpp
#include <iostream.h>
#include <conio.h>
// глобальная переменная k
int k=0;
int func (int k)
{
k++;
return k;
}

 void main ()
 {
   clrscr ();
```

```
  for (int i=1; i<=10; i++)
  {
  k=func (k);
  cout<<k<<" visov"<<"\n";
  }
    getch ();
 }
```

б) при использовании статической локальной переменной:

```
//static.cpp
#include <iostream.h>
#include <conio.h>
int favg (int);
void main ()
{
clrscr ();
int i;
   for (i=1; i<=10; i++)
   cout<<favg (i);
 getch ();
 }
 int favg (int a)
 {
  a=0;
  static int count=0;
  a++;
  count+=a;
  return count;
 }
```

Способ «б» предпочтительнее, так как позволяет использовать значение переменной *count* между вызовами функции *favg* (), и, в отличие от способа «а», ни одна функция в программе не имеет доступа к ней.

Для решения задачи нельзя использовать локальную переменную *count*, т. к. между вызовами функции *favg* () ее значение теряется.

Практическая работа № 10

Упражнение 1

```
//objint.cpp
#include <iostream.h>
#include <conio.h>
class objint                    //определение класса
{
private:
  int a;
public:
//метод класса, инициализирующий поле a значением 0 и выводящий значение поля a на экран
void setdata1 ()
  {
   a=0;
  cout<<"b="<<a;
  }
 //метод класса, инициализирующий поле a ненулевым значением и
// выводящий значение поля a на экран
void setdata2 (int d)
  {
   a=d;
   cout<<"\nc="<<a;
  }
//метод класса, инициализирующий значение поля a суммой двух целых чисел  и выводящий значение поля a на экран
 void plus (int b1, int b2)
 {
   a=b1+b2;
   cout<<"\ns="<<a;
   }
};
void main ()
{
clrscr();
//определение трех объектов класса objint
objint  b, c, s;
int b1 =7;
int b2 =8;
```

```
//вызовы методов класса
b. setdata1 ();
c.setdata2 (45);
s.plus (b1, b2);
getch ();
}
```

Упражнение 2

```
//objpart.cpp
#include <iostream.h>
#include <conio.h>
//определение класса
class part
{
private:
  int modelnumber;       //номер изделия
  int partnumber;        //номер детали
  float cost;            //стоимость детали
   public:
   //установка данных
  void setpart (int mn, int pn,float c)
  {
    modelnumber=mn;
    partnumber=pn;
    cost=c;
  }
  //вывод данных
  void showpart ()
  {
   cout<<"\nmodel "<<modelnumber;
   cout<<"\npartnumber "<<partnumber;
   cout<<"\nstoimist "<<cost;
  }
};
void main ()
{
clrscr();
part part1;                          //определение объекта класса part
part1.setpart (6244, 373, 215.5);
```

```
part1.showpart ();
getch ();
}
```

Упражнение 3

```
//objint.cpp
#include <iostream.h>
#include <conio.h>
class objint
{
private:
  int a;
public:
//конструктор
objint (): a(0)
 {}
void setdata1 ()
 {
 cout<<"b="<<a;
 }
void setdata2 (int d)
  {
   a=d;
   cout<<"\nc="<<a;
  }
 void plus (int b1, int b2)
 {
   a=b1+b2;
   cout<<"\ns="<<a;
   }
};
void main ()
{
clrscr();
objint  b, c, s;
int b1 =7;
int b2 =8;
b.setdata1 ();
c.setdata2 (45);
```

```
s.plus (b1, b2);
getch ();
}
```

Вопросы и задания для отчета

2. Вариант программы:

```
//empl.cpp
#include <iostream.h>
#include <conio.h>
class empl
{
private:
  int number;
  float oklad;
public:
void setdata (int n, float c)
  {
   cout<<"\nvvedite numer, oklad sotrudnikov\n";
   cin>>n>>c;
   number = n;
   oklad = c;
}
void showdata ()
{
 cout<<"\nn="<<number<<" oklad="<<oklad;
   }
};
void main ()
{
clrscr();

empl  s1, s2, s3;
int n;
float c;
s1.setdata (n, c);
s2.setdata (n, c);
s3.setdata (n, c);
s1.showdata ();
```

```
s2.showdata ();
s3.showdata ();
getch ();
}
```

5. Вариант программы:

```
//avto.cpp
#include <iostream.h>
#include <conio.h>
class avto
{
private:
  unsigned int kol;
  double summa;
public:
avto (): kol (0), summa (0)
{ }
void setdata1 ()
  {
   kol++;
   summa+=0.5;
  }
void setdata2 ()
 {
  kol++;
 }
void showdata ()
{
 cout<<"\nkol="<<kol<<" summa="<<summa;
   }
};
void main ()
{
clrscr();
avto s1;
char ch;
ch='y';
while (ch !='q')
  {
```

```
  cout<<"y/n\n";
  cin>>ch;
  if (ch=='y')
    s1. setdata1 ();
  if (ch=='n') s1.setdata2 ();
  }
s1.showdata();
getch ();
}
```

Практическая работа № 11

Упражнение 1

Алгоритм работы программы:

1. Объявляем целочисленный массив A из десяти чисел.
2. Вводим значение элементов массива A с клавиатуры.
3. Присваиваем начальное значение переменной sum, в которой будет храниться значение суммы элементов массива A.
4. Выводим значение суммы на экран.

Упражнение 2

Разница в алгоритме работы программы состоит в том, что в данной программе значения элементов массива задаются в тексте программы.

Упражнение 3

Смысл использования переменной *n* состоит в указании с ее помощью размеров массива *a*. Переменная *n* в этом случае объявлена как типизированная константа.

Упражнение 4

Алгоритм работы программы:

1. Объявляем целочисленную константу *n* со значением 10, указывающую размер массива *a*.
2. Задаем значения элементов массива *a*.
3. Выводим значение элементов массива на экран.

Разница в алгоритме работы программы состоит в том, что в данной программе значения элементов массива задаются с помощью формулы.

Упражнение 5

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int n;
float s=0;
const int n=20;
float a[n];
for (int i=0; i<n; i++)
  {
  cout<<"\n vvedite "<<i<<" chislo  ";
  cin>>a[i];
   }
for (i=0; i<n; i++)
  if (i % 2==0) s=s+a[i];
cout<<"\n s="<<s;
getch ();
}
```

Упражнение 6

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int i;
float a[18], s1=0, s2=0, s3=0;
cout<<"vvedite ocenki sportsmena\n";
for (i=0; i<18; i++)
    cin>>a[i];
```

```
for (i=0; i<18; i++)
  {
   if ((i>=0) && (i<6))
       s1=s1+a[i];
   if ((i>5) && (i<13))
        s2=s2+a[i];
   if (i>12) s3=s3+a[i];
  };
if ((s1>s2) && (s1>s3))
  cout<<"\n objazatelnaj programma";
else
  if ((s2>s1) && (s2>s3))
     cout<<"\n korotkaja programma";
  else cout<<"\n proizvolnaja programma";

getch ();
}
```

Упражнение 7

```
#include <iostream.h>
#include <conio.h>
const int n=6;
int funk ( int [n]);
void main ()
{
 clrscr ();
 int mas [n];
 for (int i=0; i<n; i++)
  {
  cout<<"\nvvedite "<<i<<" element massiva\n";
  cin >>mas[i];
  }
  int k=funk (mas);
  cout<<"index="<<k<<" element="<<mas[k];
  getch ();
  }
  int funk (int mas [n])
{
  int index;
```

```
  int max=mas[0];
  for (int i=1; i<n; i++)
   if (max<mas[i])
    {
     max=mas[i];
     index=i;
     };
   return index;
  }
```

Упражнение 8

```
//array.cpp
#include <iostream.h>
#include <conio.h>
const int size=3;
struct tovar
{
 char tovarname [15];
 int rubl;
 int kop;
 };

 void main ()
 {
   clrscr ();
   int n;
   tovar t[size];
   for (n=0; n<size; n++)
   {
   cout<<"\nvvedite naimenovanie tovara\n" ;
   cin>>t[n]. tovarname;
    cout<<"\nvvedite stoimost tovara v rub\n" ;
   cin>>t[n].rubl;
    cout<<"\nvvedite stoimost tovara v kop\n";
   cin>>t[n].kop;
   };
   for (int i=0; i<size; i++)
   {
```

```
for (n=i+1; n<size; n++)
{
 if (t[i].rubl>t[n].rubl)
   {
    cout<<t[i].tovarname<<" doroge "<<t[n].tovarname<<"\n";
    break;
    }
 else
  { cout<<t[n].tovarname<<" doroge "<<t[i].tovarname<<"\n";
   break;
   }
};
};
  getch ();
  }
```

Вопросы и задания для отчета

5. Варианты программы:

```
//u5.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr ();
 int i;
 const int n=15;
 int a[n];
 for (i=0; i<n; i++)
    {
      cout<<"\n vvedite "<<i<<" element massiva ";
      cin>>a[i];
     }
 for (i=3; i<n; i+=3)
    {
     cout<<" "<<a[i];
     }
 getch ();
}
```

6. Программа:

а) //u6a.cpp

```
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int i, s1=0,s2=0;
const int n=30;
int a[n];
for (i=0; i<n; i++)
  {
    cout<<"\n vvedite kolichestvo osadkov "<<i+1<<"  dnja iyunja";
    cin>>a[i];
    if (i<16)
       s1=s1+a[i];
    else
      s2=s2+a[i];
   }
 if (s1>s2)
    cout<<"\n 1 polovina";
 else
    cout<<"\n 2 polovina";
getch ();
}
```

б) //u6b.cpp

```
#include <iostream.h>
#include <conio.h>
void main()
{
 clrscr ();
 int i, s1=0,s2=0,s3=0;
const int n=30;
int a[n];
for (i=0; i<n; i++)
 {
     cout<<"\n vvedite kolichestvo osadkov "<<i+1<<"  dnja iyunja";
     cin>>a[i];
     if (i<10)
```

```
            s1=s1+a[i];
          else
          if ((i>=10) && (i<20))
            s2=s2+a[i];
          else
            s3=s3+a[i];
        }
       if ((s1>s2)&&(s1>s3))
          cout<<"\n 1 dekada";
       else
          if ((s2>s1)&&(s2>s3))
            cout<<"\n 2 dekada";
          else
            cout<<"3 dekada";
      getch ();
       }
```

7. Возможный вариант программы:

```
//order.cpp
#include <iostream.h>
#include <conio.h>
class order
{
  private:
   enum {MAX=10};
  int st [MAX];
  int head;
  int tail;
  int a;
 public:
 order ()
     {head=0; tail=0;}
 void put (int var)
    {st[head++]=var;}
 int get ()
   {a=st[tail];
    tail++;
    return a;}
 };
 void main ()
```

```
 {
  clrscr ();
  order s1;
  s1.put (11);
 s1.put (22);
  cout<<"1:"<<s1.get()<<endl;
  cout<<"2:"<<s1.get()<<endl;
  s1.put(33);
  s1.put(44);
  s1.put(55);
  s1.put(66);
  cout<<"3:"<<s1.get()<<endl;
  cout<<"4:"<<s1.get()<<endl;
  cout<<"5:"<<s1.get()<<endl;
  cout<<"6:"<<s1.get()<<endl;
  getch();
  }
```

Практическая работа № 12

Упражнение 1

```
//u1.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int a[5]={3, 5, 7, 1, 9};
int i, min=a[0];
for (i=1; i<5; i++)
  if (a[i]<min) min=a[i];
cout<<"min="<<min;
getch ();
}
```

Упражнение 2

В данных фрагментах:

а) вычисляется сумма элементов массива с четными номерами;

б) если найден элемент массива, равный 5, на экран выводится символ «!»;

в) на экран выводится элемент массива, если он больше 0 и меньше 1;

г) каждый предыдущий элемент массива заменяется следующим, таким образом, все элементы массива за исключением нулевого сдвигаются на одну позицию влево.

Упражнение 3

а) 1 шаг – объявление и инициализация целочисленного массива *c*;

2 шаг – к значению первого элемента массива прибавляется значение первого элемента;

3 шаг – к значению первого элемента массива прибавляется значение второго элемента массива;

4 шаг – к значению первого элемента массива прибавляется значение третьего элемента массива;

5 шаг – в результате на экран будет выведено значение первого элемента массива, равное 7;

б) 1 шаг – объявление и инициализация целочисленного массива *c*;

2 шаг – к значению первого элемента массива прибавляется значение третьего элемента;

3 шаг – к значению второго элемента массива прибавляется значение второго элемента массива;

4 шаг – к значению третьего элемента массива прибавляется значение первого элемента массива;

5 шаг – в результате на экран будет выведено значение суммы элементов массива, равное 15;

в) 1 шаг – объявление и инициализация целочисленного массива *i*;

2 шаг – на экран выводится значение первого элемента массива, равное 3;

3 шаг – на экран выводится значение третьего элемента массива, равное 1;

г) 1 шаг – объявление целочисленного массива *S*, определение первого элемента массива, объявление и инициализация целочисленной переменной *f*;

2 шаг – к значению переменной *f* прибавляется значение первого элемента массива;

3 шаг – к значению переменной *f* прибавляется значение второго элемента массива;

4 шаг – к значению переменной *f* прибавляется значение третьего элемента массива;

5 шаг –значение переменной *f* , выводимое в результате работы программы, будет равно 2, так как в массиве *S* определен только первый элемент.

Упражнение 4

```
//u4.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
float A, B;
cout<<"vvedite A, B\n";
cin>>A>>B;
const int N=5;
int i, k=0;
float x[N];
for (i=0; i<N; i++)
   {
    cout<<"vvedite "<<i<< " element massiva ";
    cin>>x[i];
   }
for (i=0; i<N; i++)
  if ((x[i]>=A) && (x[i]<=B))
    k+=1;
cout<<"k="<<k;
getch ();
}
```

Упражнение 5

```
a) //u5a.cpp
   #include <iostream.h>
   #include <conio.h>
   void main()
   {
   clrscr ();
   int F;
   cout<<"vvedite F\n";
   cin>>F;
   const int N=5;
```

```
    int i, P=0;
    int x[N];
    for (i=0; i<N; i++)
        {
         cout<<"vvedite "<<i<< " element massiva ";
         cin>>x[i];
        }
    for (i=0; i<N; i++)
       if (x[i]==F)
        {
          P=i;
          break;
        }
    cout<<"P="<<P;
    getch ();
   }
```

б)

```
    //u5b.cpp
    #include <iostream.h>
    #include <conio.h>
    void main()
    {
    clrscr ();
    int F;
    cout<<"vvedite F\n";
    cin>>F;
    const int N=5;
    int i, P=0;
    int x[N];
    for (i=0; i<N; i++)
       {
        cout<<"vvedite "<<i<< " element massiva ";
        cin>>x[i];
       }
    for (i=0; i<N; i++)
    if (x[i]==F)   P=i;
    cout<<"P="<<P;
    getch ();
    }
```

Упражнение 6

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
const int N=5;
float A[N]={5, 7, 1, 9, 3};
for (int i=1; i<N; i++)          //упорядочиваем массив
  for (int j=0; j<N - i; j++)
    if (A[j]< A[j+1])
    {
      float tmp=A[j];
      A[j] = A[j+1];
      A[j+1]=tmp;
      }
int m=N / 2;                              //находим медиану
cout<<A[m];
getch ();
}
```

Упражнение 7

```
//u7.cpp
#include <iostream.h>
#include <conio.h>
void main ()
{
clrscr ();
int m, i, j;
const int n = 7;
int a[n]={1, 2, 3, 4, 5, 6, 7};
cout<<"vvedite m";
cin>>m;
m= m % n;
for (i=0; i<m; i++)
{
   int temp=a[n-1];
   for (j=0; j<n-1; j++)
```

```
   {
     a[n-j-1]=a[n-j-2];
    }
   a[0]=temp;
}
for (i=0; i<n; i++)
   cout<<" "<<a[i];
getch ();
}
```

Упражнение 8

```
//u8.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int i, k, j;
const int n=5;
int a[n];
for (i=0; i<n; i++)
  {
  cout<<"vvedite "<<i<<" element massiva";
  cin>>a[i];
  }
for (i=0; i<n; i++)
   for (j=i+1; j<n; j++)
   {
     int k=i;
     if (a[j]<a[k])
       {
          k=j;
          int temp = a[i];
          a[i] = a[k];
          a[k] = temp;
       }
     }
```

```
for (i=0; i<n; i++)
  cout<<" "<<a[i];
getch ();
}
```

Вопросы и задания для отчета

6. Вариант программы:

```
//u6.cpp
#include <iostream.h>
#include <conio.h>
void main()
{
clrscr ();
int i;
const int n=20;
int a[n]={1, 2, 3, 4, 5, 6, 3, 2, 9, 10, 2, 4, 5, 6, 8, 1, 2, 3, 4, 2};
cout<<"menshe 3 pobed u komand s nomerami\n";
for (i=0; i<n; i++)
   if (a[i]<3)
      cout<<" "<<i+1;
getch ();
}
```

7. Варианты программы:

```
shell.cpp
#include <iostream.h>
#include <conio.h>
const int n=5;


 void main ()
 {
   clrscr ();
   int i;
   int a[n]={2,4,1,3,0};
   int k=n/2;
   while (k>0)
   {
    for (int j=n-k; 0; j--)
     {
```

```
i=j;
 while (i<=n-k)
  {
     if (a[i]>a[i+k])
        {
          int c=a[i];
          a[i]=a[i+k];
          a[i+k]=c;
        };
     i=i+k;
     };
  };
  k=k/2;
  };
for (i=0; i<5; i++)
   cout<<a[i]<<" ";
 getch ();
 }
```

ПРИЛОЖЕНИЯ

Приложение 1

Порядок выполнения практических работ

Допуск обучающегося к выполнению практических работ в рамках данного курса возможен только при условии соблюдения им правил техники безопасности и внутренних правил работы с персональным компьютером.

Практические занятия, предусмотренные при изучении каждой темы, предполагают самостоятельный анализ задания, разработку, отладку и тестирование соответствующей программы.

Алгоритм выполнения практической работы:

1. Предварительно изучить теоретический материал соответствующей темы.
2. Письменные упражнения выполнять в тетради для практических занятий.
3. Упражнения, требующие написания программы, выполнять следующим образом:

1) разработать и реализовать программу;

2) протестировать ее работу при различных входных параметрах;

3) показать преподавателю на примерах правильность результатов, полученных с помощью разработанной программы;

4) по требованию преподавателя кратко сформулировать алгоритм работы программы и выделить в программе основные блоки, ответственные за выполнение определенных действий (например, ввод исходных данных, объявление переменных, вывод на экран и т.д.);

5) по требованию преподавателя объяснить работу всех использованных в программе языковых конструкций;

6) по требованию преподавателя объяснить назначение и смысл всех использованных констант и переменных.

Приложение 2

Образец оформления отчета

Пмерные теоретические вопросы:

1. Консольный ввод-вывод (кратко о библиотеке, основных операторах и т.д.).
2. Потоковый ввод-вывод …. (кратко о библиотеке, основных операторах и т.д.).
3. Арифметические операции (кратко о бинарных операциях с примерами).

Практические вопросы:

1. Задание

Напишите программу, вычисляющую произведение двух заданных с клавиатуры чисел.

2. Листинг программы:

```
#include <iostream.h>
#include <conio.h>
void main ()
{
  clrscr ();
  float a,b;
  cout<< “vvedite a=”;
  cin >>a;
  cout<< “vvedite b=”;
  cin >>b;
  float c=a*b;
  cout <<”Resultat: “<<c;
 }
```

3. Входные данные и результаты:

```
vvedite a = 2
vvedite b = 3
Resultat: 6
```

Практическая работа № 1	Студент	ФИО, группа
	Практические вопросы	Дата Подпись преподавателя
	Теоретические вопросы	Дата Подпись преподавателя

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Березин, С. Б.* Начальный курс С и С++ / Б.И. Березин, С.Б. Березин. – М.: ДИАЛОГ-МИФИ. 2008. – 288 с. ISBN 5-86404-075-4.

2. *Гуденко, Д. А.* Сборник задач по программированию / Д.А. Гуденко, Д.В. Петроченко. – СПб.: Питер, 2003. – 475 с. – ISBN 5-318-00782-1.

3. *Лафоре, Р.* Объектно-ориентированное программирование в С++ / Р. Лафоре. – СПб.: Питер, 2008. – 928 с. – ISBN 978-5-94723-302-5.

4. *Эллис, М.* Справочное руководство по языку программирования С++ с комментариями: пер. с анг. / М. Эллис, В. Строуструп. – М. : Мир, 1992. – 445 с. – ISBN 5-03-002868-4.

5. *Монахов, М. Ю.* Информатика. Кн. 4. Программные и аппаратные средства: учеб. пособие / М. Ю. Монахов, Ю. А. Иларионов; Владим. гос. ун-т. – Владимир : ВлГУ, 2002. –92 с. – ISBN 5-89368-306-4.

6. *Подбельский, В. В.* Язык С++ / В.В. Подбельский. – 5-е изд. – М.: Финансы и статистика, 2003. – 560 с. – ISBN 5-279-02204-7.

7. *Феськов, С. В.* Информатика. 10 – 11 классы. Программирование на языке С++ (материалы к занятиям) / С.В. Феськов. – Волгоград: Учитель, 2009. – 133 с. – ISBN 978-5-7057-1674-6.

8. *Шилдт, Г.* С++: Руководство для начинающих : пер. с англ. / Г. Шилдт. – 2-е изд. – М.: Вильямс, 2005. – 672 с. – ISBN 8459-0840-х.

9. *Воронин, А. А.* Методы программирования и прикладные алгоритмы: задачник / А.А. Воронин, Д.А. Кузнецов, М.Ф. Кузнецова; Владим. гос. ун-т. – Владимир: Ред.- издат. комплекс ВлГУ, 2005. – 128 с. – ISBN 5-89368-577-6.

ОГЛАВЛЕНИЕ

Учебное издание

АРТЮШИНА Лариса Андреевна
МОНАХОВ Юрий Михайлович
ВОРОНИН Алексей Александрович

ИНФОРМАТИКА. ПРОГРАММИРОВАНИЕ: C++

Учебное пособие

В двух частях

Часть 1. Введение в язык C++

Подписано в печать 23.05.11.
Формат 60x84/16. Усл. печ. л. 7,67. Тираж 150 экз.
Заказ
Издательство
Владимирского государственного университета
600000, Владимир, ул. Горького, 87