Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет
Кафедра інформаційних технологій та мехатроніки

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт

“Програмування на мові С++ у середовищі Microsoft Visual Studio 2010”

З дисципліни “Програмування”
(Розділ “Структурне програмування на мові С++”)

Для студентів напряму підготовки 6.050201 “Системна інженерія”
Семестр 1

Розробник доцент кафедри ІТМ Симбірський Г.Д.

Харків, 2016

Лабораторная работа № 1

ИССЛЕДОВАНИЕ ИНТЕРФЕЙСА СРЕДЫ VISUAL STUDIO 2010. РАЗРАБОТКА ПРОГРАММ С ЛИНЕЙНОЙ СТРУКТУРОЙ ДЛЯ VISUAL C++ 2010. ИССЛЕДОВАНИЕ ВВОДА И ВЫВОДА ДАННЫХ.

Цель работы:

1. Исследование интерфейса среды Visual Studio 2010;
2. Приобретение навыков алгоритмизации задач;
3. Исследование процессов ввода и вывода данных в Visual C++ 2010.
4. Изучение приемов создания и отладки консольных проектов и программ с линейной структурой в среде программирования Visual C++ 2010;

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1. Общие сведения о программе Visual Studio 2010

Visual Studio 2010 — это полный набор средств для разработки программ и приложений для компьютеров, мобильных устройств и др. с использованием различных языков программирования. Среда разработки Visual Studio 2010 предназначена для написания программ и приложений на языках .NET, HTML, JavaScript, C# и C++. Возможности этого программного инструмента очень велики. Поэтому он получил название **ИСР - интегрированная среда разработки** (Integrated Development Environment – **IDE**). Не выходя за рамки этой среды, можно решать следующие задачи:

1. Формировать заготовки приложений без написания текстов программ;
2. Просматривать проекты несколькими разными способами;
3. Редактировать файлы заглавий и текстов программ;
4. Формировать визуальный графический интерфейс (меню и диалоговые окна);
5. Компилировать и компоновать программы;
6. Производить отладку разрабатываемых программ и приложений в процессе их выполнения.

Visual C++ 2010 - одна из составных частей программы Visual Studio 2010 и является самодостаточной средой для разработки различных программ и приложений. Visual C++ 2010 включает следующие основные компоненты:

1. **Редактор** – для ввода, просмотра и проверки исходного кода;
2. **Компилятор** – для трансляции начального кода C++ в объектный код;
3. **Компоновщик** – создает выполняемые файлы, объединяя объектный код и библиотечные модули;
4. **Библиотеки** – сборники стандартных программ-модулей, которые можно применять в разрабатываемых программах и приложениях. Одна из самых важных библиотек - Microsoft Foundation Classes (базовые классы Microsoft или MFC), используемая при написании программ, работающих под управлением Microsoft Windows. Кроме этого, стандартные библиотеки C++ поддерживают операции ввода-вывода и другие стандартные возможности языка;

5. Прочие инструментальные средства, включая AppWizard (Мастер приложений), ClassWizard (Мастер классов) и Resource Editor (Редактор ресурсов).

2. Интерфейс программы Visual Studio 2010

Интерфейс Visual Studio 2010 нагляден и удобен для создания, редактирования, просмотра и отладки компонентов разрабатываемых программ и приложений – ресурсов, классов, файлов и др. Основной экран разделен на части с различными функциями, размеры которых можно изменять по своему усмотрению. Для облегчения решения стандартных задач есть контекстные меню, доступ к которым осуществляется с помощью нажатия правой кнопки мыши на различных компонентах изображения на экране.

С помощью Visual C++ 2010 можно работать с разрабатываемым приложением как с проектом.

Проект - это регламентированный (определенный протоколами Visual Studio 2010) набор файлов: заглавий, текстов программ, ресурсов, установок, конфигураций и др. Интерфейс Visual Studio 2010 дает возможность работать со всеми компонентами проекта одновременно, поэтому экран разделен на несколько зон (окон). Каждая разрабатываемая программа (приложение), даже самая простая, является проектом.

Набор папок и файлов, создаваемый средой Visual C++ 2010 при разработке проектов, называется **решением**.

При запуске программы Visual Studio 2010 открывается окно **Начальная страница – Microsoft Visual Studio (Администратор)**, состоящее из трех частей:

1. **Обозреватель решений** – окно в левой части экрана, в котором в дальнейшем будут представлены в виде дерева папки и файлы текущего проекта. В случае необходимости на месте данного окна можно открыть **Окно классов** данного проекта, его **Диспетчер свойств** или **Командный обозреватель**;
2. Окно **Начальная страница** в правой части экрана, в котором предложены на выбор возможные действия пользователя, в частности, **Создать проект** или **Открыть проект**, а также для открытия предложен список последних проектов;
3. Окно **Вывод** в нижней части экрана, в котором при построении решений проектов выводятся сообщения о предупреждениях (**warning**) и ошибках (**error**). На месте этого окна можно открыть **Окно определения кода** или окно **Результаты поиска.**

При создании консольных приложений на языке C++ работа осуществляется в окне <**Имя проекта**> **– Microsoft Visual Studio (Администратор)**, состоящем из трех частей:

1. **Обозреватель решений** (описание см. выше);
2. Окно для ввода программного кода и для работы с программой в правой части экрана;
3. Окно **Вывод** (описание см. выше).

3. Редактирование и отладка программ в Visual C++ 2010

Код (текст) программы вводится в окне редактора с использованием клавиатуры и основных приемов работы с текстом в операционной системе (ОС) Windows. При отображении текста программы используется синтаксическое раскрашивание. По

умолчанию текст программы черного цвета с комментариями зеленого цвета и ключевыми и служебными словами синего цвета.

После того, как набран программный код, следует приступить к отладке программе. Для этого следует открыть меню Построение **и выбрать пункт** Построить решение**. После чего все сообщения о предупреждениях и ошибках отображаются в окне** Вывод**. Ошибки следует исправлять обязательно, т. к. программа с ошибками выполняться не будет. С предупреждениями программа будет выполняться, однако их необходимо проанализировать и, если возможно, исправить или учесть. Описание ошибки находится в строке сообщения об ошибке (для получения подробного описания ошибки следует нажать** <F1>**). Чтобы исправить ошибку необходимо дважды щелкнуть в окне отладчика на сообщении о данной ошибке, после чего в окне редактора появится указатель на строке с ошибкой. После исправления всех ошибок необходимо открыть пункт меню** Отладка**. Программа запускается при выборе пункта меню** Начать отладку **или при нажатии клавиши** <F5>**.**

4. Данные и переменные в языке C++

4.1. Типы данных в языке C++.

Основные или базовые типы данных в языке C++ следующие:

int – целочисленные данные (4 байта), диапазон значений: целые от -2 147 483 647 до 2 147 483 647;

char – символьные данные (1 байт), диапазон значений: от 0 до 255или от -127 до 128;

float – числа с плавающей запятой (4 байта), диапазон значений: от 3.4E-38 до 3.4E+38;

double – число с плавающей запятой двойной точности (8 байт), диапазон значений: от 1.7E-308 до 1.7E+308;

bool – логические переменные (**true** и **false**).

Кроме вышеприведенных основных типов данных, в языке C++ используются еще несколько типов данных.

4.2. Объявление переменных и констант в языке C++

Фрагмент памяти, в котором хранится элемент данных и к которому можно обращаться по некоторому имени, называется **переменной**. Имена переменных могут включать буквы латинского алфавита **A** – **z** (в верхнем или нижнем регистре), цифры от 0 до 9 и знак подчеркивания. Имена переменных должны начинаться либо с буквы, либо со знака подчеркивания. В C++ принято назначать имена переменных с прописных букв, а классов – с заглавных. Компилятор C++ различает прописные и строчные буквы, например, **Sum** и **sum** означают разные переменные.

Объявление переменной с одновременным заданием типа хранимого под ее именем элемента данных осуществляется с использованием следующего синтаксиса:

ТипПеременной ИмяПеременной;

Например, строка **int arg;** объявляет целочисленную переменную с именем **arg**.

В языке C++ есть зарезервированные слова, имеющие специальное значение – **ключевые слова**. Это названия типов данных и некоторых операторов и др. Редактор среды разработки Visual C++ 2010 подсвечивает их **синим** цветом. Имена переменных не должны совпадать с ключевыми словами.

Объявляя переменную, можно сразу присвоить ей начальное значение. Например

int sum=0; или **int sum(0);**
float a=2.7; или **float a(2.7);**

Переменная объявляется **перед** тем местом, где она будет впервые задействована. Подробнее о месте объявления переменных в С++ будет сказано ниже.

Переменную, не меняющую своего значения в программе, можно использовать как константу. В С++ объявление константы выглядит следующим образом:

const Тип ИмяКонстанты = Значение; .

Например, объявить постоянную **π** можно следующим образом:

const float pi = 3.14159; .

5. Выражения и операции в среде Visual C++ 2010

В языке С++ следующим уровнем представления данных после переменных и констант являются выражения.

Выражение – это некоторая допустимая комбинация переменных, констант, функций и знаков операций для вычислений в программах. Выражения в языке С++ записываются в строчку.

Например, формула

$$d = \frac{a + b(c + e)}{c(a + b) + t}$$

в языке С++ запишется следующим образом:

$$d = (a + b * (c + e)) / (c * (a + b) + t)) .$$

В приведенном выше выражении знак “=” обозначает операцию **присваивания**, которая выполняется следующим образом. Вычисляется значение выражения в правой части и присваивается переменной **d.**

В данной лабораторной работе будут использоваться привычные для студентов арифметические операции; в полном объеме операции среды Visual C++ 2010 приведены в лекционном материале по курсу и будут исследованы в следующих работах. Приоритет (очередность выполнения) арифметических операций такой же, как и в математике (см. таб. 1.1).

Математические действия с переменными, константами и функциями в языке С++ записываются только в строчку, при этом для соблюдения необходимой по условию задачи очередности операций используются круглые скобки.

Таблица 1.1. Арифметические операции в языке С++

Операция	Знак в языке

	C++
Сложение	+
Вычитание	-
Умножение	*
Деление	/

6. Стандартные математические функции в среде Visual C++ 2010

В языке C++ в выражения можно вставлять стандартные математические функции, которые вызываются из библиотеки **<math.h>**. Перечень математических функций, которые чаще всего встречаются в вычислениях приведены в таблице 1.2.

Таблица 1.2. Основные стандартные математические функции (библиотека math.h)

Название функции	Что вычисляет	Тип данных функции и аргумента
abs(x)	Абсолютное значение (модуль) аргумента |x|	**int abs(int x)**
exp(x)	Экспонента e^x	**double exp(double x)**
log(x)	Натуральный логарифм ln x	**double log(double x)**
log10(x)	Десятичный логарифм lg x	**double log10(double x)**
pow(x,y)	Возведение в степень x^y	**double pow(double x, double y)**
sqrt(x)	Квадратный корень $\sqrt{x}$.	**double sqrt(double x)**
fmod(x,y)	Остаток от деления x/y	**double fmod(double x, double y)**
sin(x)	Синус (угол задается в радианах)	**double sin(double x)**
asin(x)	Арксинус (угол задается в радианах от –1 до +1)	**double asin(double x)**
cos(x)	Косинус (угол задается в радианах)	**double cos(double x)**
acos(x)	Арккосинус (угол задается в радианах от –1 до +1)	**double acos(double x)**
tan(x)	Тангенс (угол задается в радианах)	**double tan(double x)**
atan(x)	Арктангенс (угол задается в радианах)	**double atan(double x)**

При обращении к этим функциям необходимо придерживаться следующих правил:

1) **x** и **y** должны быть типа **double**;
2) углы (аргументы) в тригонометрических функциях задаются в радианах.
3) вычисляемые функциями данные имеют тип **double**.

Например, выражение $y = \sin^3(x^4)\dfrac{e^x + z^5 - 4.5\cdot 10^2\sqrt{x}}{tg(a)(z^x + b)}$ **на языке С++ будет иметь вид:**

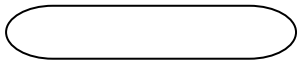
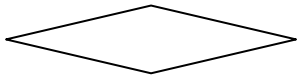
```
y=pow(sin(pow(x,4)),3)*(exp(x)+pow(z, 5) - 4.5*10*10*sqrt(x))/(tan(a)*(pow(z,x)+b))
```

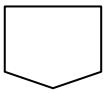
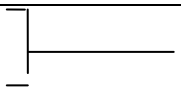
7. Алгоритмизация задач

Алгоритмизация задачи – это представление процесса решения данной задачи в виде последовательности простых операций, необходимых для получения такого решения. Как правило, алгоритм решения задачи представлен в виде блок-схемы, в которой показаны все необходимые для решения задачи операции в строгой последовательности. Состоят блок-схемы алгоритмов из стандартных обозначений.

В таблице 1.3 представлены условные обозначения основных операционных блоков схем алгоритмов.

Таблица 1.3. Основные операционные блоки схем алгоритмов

№ п/п	Условное обозначение	Наименование операции	Описание операции
1		Начало, завершение	Начало и завершение алгоритма
2		Процесс	Вычислительная операция или их совокупность
3		Решение	Проверка условия и выбор дальнейшего направления процесса решения
4		Модификация	Заголовок цикла, проверка условий цикла
5		Данные	Ввод исходных данных, вывод данных и результатов
6		Типовой процесс	Использование ранее созданных алгоритмов, подпрограмм, функций
7		Печать документа	Вывод данных на печать
8		Соединитель внутристранич ный	Разрыв линий потока в пределах одной страницы

9		Соединитель межстраничный	Перенос линий потока на другую страницу
10		Узел	Слияние линий потока
11		Комментарии	Описание операционного блока

В схемах алгоритмов операционные блоки соединяются между собой линиями потока в виде стрелок. Допускается линии потока, которые идут сверху вниз и слева направо изображать без стрелок. При необходимости блок-схему могут сопровождать комментарии.

Алгоритмизацию задач рассмотрим на примере программ с линейной структурой, которые являются самыми простыми и используются, как правило, для реализации вычислений по формулам. В программах с линейной структурой операторы выполняются последовательно один за другим. Алгоритмизация более сложных задач будет исследована при выполнении дальнейших лабораторных работ.

Исследуем процесс разработки алгоритма программы с линейной структурой на следующем примере.

Задача. Необходимо вычислить силу тока I в новогодней гирлянде, состоящей из n=50 электрических лампочек сопротивлением r=20 Ом. Используя закон Ома и формулу для расчета суммарного сопротивления последовательной цепи, составим блок-схему алгоритма (рис. 1.1).

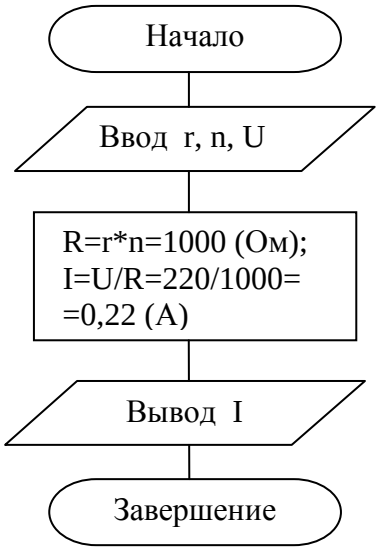


Рис. 1.1. Блок-схема алгоритма определения силы тока в новогодней электрогирлянде

8. Ввод и вывод данных в среде Visual C++ 2010

Лабораторные работы будут выполняться в Консольном приложении Win32 среды программирования Visual C++ 2010 консольный (с использованием

клавиатуры и экрана дисплея) ввод данных производится при помощи оператора **cin**. В С++ этот оператор называется также потоком ввода. Например, для ввода значений трех переменных надо записать:

cin>>a >>b>>c;

где >> - символ операции извлечения данных из потока; **a**, **b** и **c** – переменные, значения которых будут вводится. Вводимые значения должны разделяться пробелами, а ввод завершается нажатием клавиши **<Enter>**. Потоковый ввод и его операции автоматически распознают переменные и данные любого типа.

Консольный (на экран дисплея) вывод данных производится при помощи оператора **cout**. В С++ этот оператор называется также потоком вывода. Например, для вывода значений трех переменных надо записать:

cout<<a<<b<<c;

где << - символ операции вставки данных в поток; **a**, **b** и **c** – переменные, значения которых будут выводится на экран. Потоковый вывод и его операции автоматически распознают переменные и данные любого типа. Вывод в Win32 производится в командную строку окна DOS. Помимо данных можно выводить и текстовую строку, заключив ее в кавычки:

cout<<"Summa a+b+c = "<<d;

Стандартные функции ввода и вывода находятся в библиотечном файле **iostream**. Чтобы связать программу разработчика со стандартной библиотекой ввода-вывода, необходимо в начале программы указать оператор подключения:

 # include <iostream> .

Точка с запятой после операторов **include** не ставится.

Оператор **cout** часто используется с различными опциями (управляющими последовательностями) для расширения его функций.

ПРАКТИЧЕСКАЯ ЧАСТЬ

На диске D создайте папку с названием и номером группы, в которой создайте папку с Вашей фамилией, а в ней – папку с номером лабораторной работы. Например, **D:\РЕ-11\Иванов\Лр1**. Выполнять эти и последующие действия с файлами и папками необходимо в двухоконном файловом менеджере **Unreal Commander**.

Задание 1.1. Разработайте программу, вычисляющую сумму трех целых чисел с использованием операторов консольных ввода и вывода данных. Проанализируйте приведенный ниже код (текст) программы с комментариями, реализующей данную задачу.

Для этого:

1. Самостоятельно разработайте блок-схему алгоритма решения данного задания и аккуратно начертите в отчете (за основу возьмите схему на рис. 1.1).

2. Запустите Visual C++ 2010 (**Пуск/Программы/Microsoft Visual Studio 2010** или ярлык **Microsoft Visual Studio 2010** на рабочем столе).

3. При создании нового проекта в Visual C++ 2010 необходимо выполнить команды основного меню **Файл→Создать→Проект**. В левой части появившегося

диалогового окна выбрать установленный шаблон **Visual C++ →Win32** и далее тип проекта **– Консольное приложение Win32**.

4. Ввести имя проекта **Lr1-1** в текстовое поле **Имя,** удалив перед этим надпись **<Введите_имя>**.

5. Нажмите кнопку **Обзор**, выберите папку **d:\PE-11\Фамилия\Лр1** для размещения создаваемого проекта и нажмите **Ok**.

6. В открывшемся диалоговом окне **Мастер приложений Win32** выберите пункт **Параметры приложения** (слева) и тип приложения **Консольное приложение.** После этого нажмите на кнопку **Готово**.

7. Создать файл программного кода **Lr1-1.cpp**. Для этого в открывшемся окне **Lr1-1 - Microsoft Visual Studio** в центральной части с заголовком **Lr1-1.cpp** (окно редактора программного кода) ввести в уже созданную заготовку код (текст) программы (не нужно дублировать уже имеющиеся строки!).

```
#include "stdafx.h"
#include <conio.h>                      //Обеспечивает задержку окна DOS на
                                        //экране
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])    //Объявление главной функции
{                                       //Начало главной функции
int a, b;                               //Объявление переменных целого типа
int c=2, d=3;                           //Объявление переменных целого типа
                                        //и их инициализация
cout<<"vvedite b"<<endl;                //Вывод на экран надписи  vvedite b .
                                        //Здесь endl – признак перемещения
                                        //курсора на новую строку
cin>>b;                                 //Ввод значения переменной  b
a=b+c+d;                                //Вычисление суммы  b, c, d
cout<<"b+c+d = "<<a;                    //Вывод на экран значения a - суммы
                                        //переменных  b, c и d
getch();                                //Функция задержки окна DOS на
                                        //экране
return 0;
}                                       // Конец главной функции
```

Внимание!! И в данной, и в последующих задачах студентам необходимо разобраться с действием и назначением каждой строки программного кода!!

8. После ввода программного кода в окно редактора тщательно проверьте введенный код на отсутствие ошибок, а затем откройте пункт меню **Построение** и щелкните левой клавишей мыши (ЛК) на команде **Построить решение**. **Visual C++ 2010** проведет анализ Вашей программы с выводом результатов этого анализа в окно **Вывод**. В последней строке выведенных результатов анализа будет указано, есть ли в программе ошибки.

9. Просмотрите текст в окне **Выводы**, где будут указаны характер ошибки и номер строки ее местонахождения. После устранения ошибок в программном коде снова выполните п. 7.

10. Если ошибки отсутствуют, откройте пункт меню **Отладка** и выполните команду **Начать отладку**.

11. В открывшемся окне DOS прочитайте инструкции, введите необходимые данные (подтверждая ввод данных нажатием клавиши **<Enter>)** и изучите полученные результаты. Результат выполнения **Задания 1** (проект **Lr1-1**) имеет следующий вид:

vvedite b
10
b+c+d = 15

12. Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр1**. Поля документа сделайте по 0,5 см.

13. Поместите окно DOS с результатами решения **Задания 1.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr1-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр1**. Над вставленным рисунком проставьте номер задания – **1-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

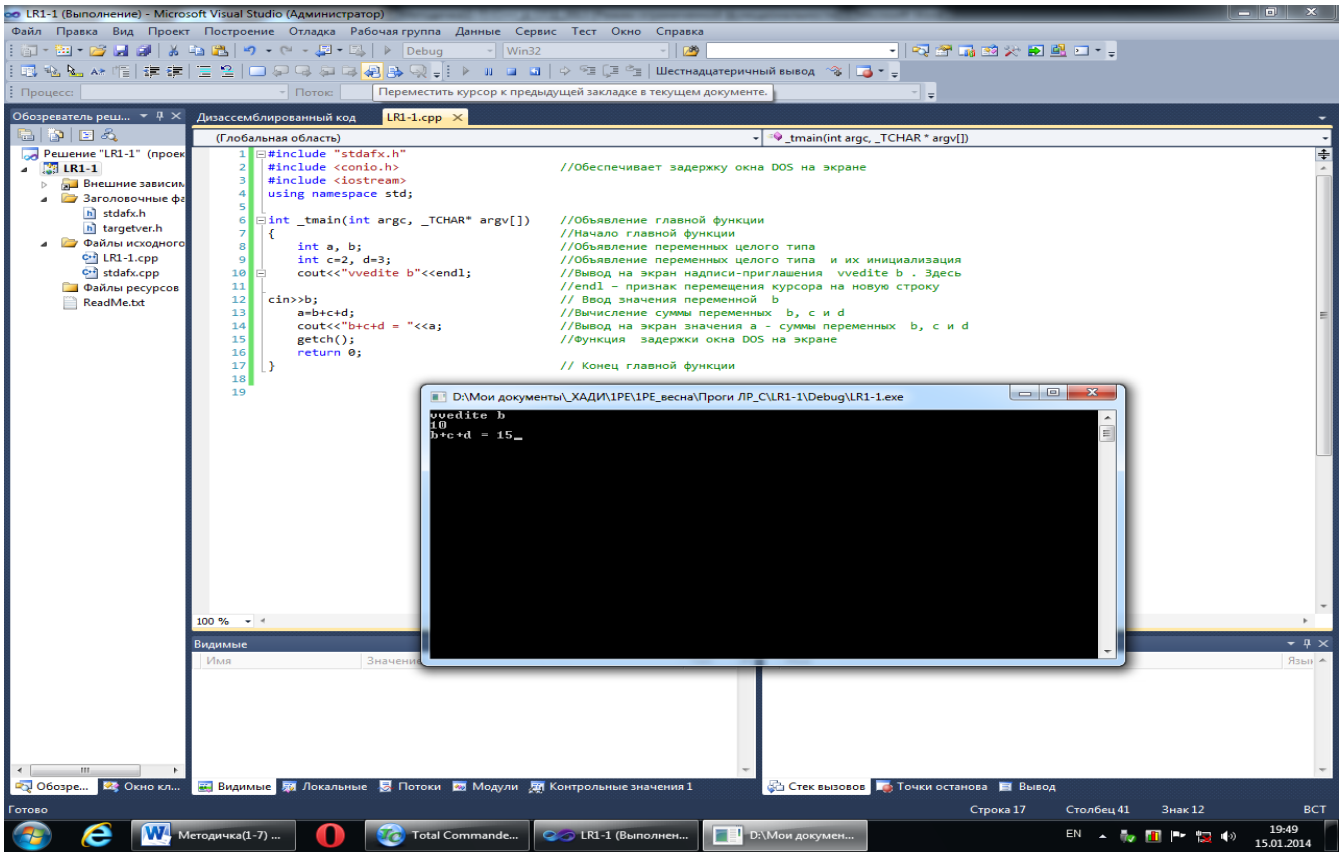


Рис. 1.2. Результаты выполнения **Задания 1**. Копия экрана с результатами выполнения проекта **Lr1-1.**

14. Закройте окно DOS.

15. Откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Следующие задания данной и других лабораторных работ выполняйте строго в соответствии с приведенным выше порядком действий!

Задание 1.2. Создайте проект **Lr1-2** для вычисления переменной **a** по формуле $a = b\frac{c + 2d - cd}{d(5c + 4b)}$ с использованием операторов консольных ввода и вывода данных. Блок-схема алгоритма решения данного задания аналогична блок-схеме на рис. 1.1. Проанализируйте код (текст) программы с комментариями, приведенный ниже.

```
#include "stdafx.h"
#include <conio.h>                  //Файл conio.h обеспечивает задержку окна
                                    //DOS на экране дисплея
#include <iostream>                 //Директива include подключает файл ввода-
                                    //вывода iostream
using namespace std;                //Подключает все имена из пространства
                                    //имен std
int _tmain(int argc, _TCHAR* argv[])    //Объявление главной функции
{                                   //Начало главной функции
double a;                           //Объявление переменной  a
double b, c=2, d=3;                 //Объявление переменных  b, c, d и их
                                    //инициализация
   cout<<"vvedite b"<<endl;         //Вывод на экран надписи vvedite b.  Здесь
                                    //endl – признак перемещения курсора на
                                    //новую строку
   cin>>b;                          //Ввод значения переменной  b
   a=b*(c+2*d-c*d)/(d*(5*c+4*b));   //Вычисление  a по заданной формуле
   cout<<"a = "<<a;                 //Вывод на экран значения a
getch();                            //Функция  задержки окна DOS на экране
return 0;
}
```

Результат выполнения **Задания 2** (проект **Lr1-1**) имеет следующий вид:

```
vvedite b
10
a = 0.133333
```

Сохраните результаты выполнения Задания 2 **в виде копии экрана таким же образом, как и в** Задании 1**.**

Задание 1.3. **Самостоятельно создайте проект** Lr1-3 **для вычисления площади поверхности и объема цилиндра по формулам**

$$S = 2\pi r(h+r),$$

где **S** – площадь поверхности цилиндра; **r** – радиус основания цилиндра; **h** – высота цилиндра;

$$V = \pi r^2 h ,$$

где **V** - объем цилиндра.

Радиус основания цилиндра равен номеру Вашего компьютера, а высота – удвоенному номеру компьютера.

Перед составлением программного кода **аккуратно (!)** начертите в отчете блок-схему алгоритма решения задачи, которая аналогична блок-схеме на рис. 1.1.

Примечание. Поскольку в С++ нет операции возведения в степень, то для вычисления r^2 следует воспользоваться оператором умножения, менее трудоемким, чем функция **pow**.

Число π задайте как константу, а значения **r** и **h** введите с клавиатуры. Результат вычислений должен содержать необходимые пояснения. Например, для 10-го компьютера **V cilindra = 6 280**.

Здесь и далее сохраните результаты выполнения задания таким же образом, как и в Задании 1**.**

Задание 1.4. Создайте проект **Lr1-4** для вычисления переменной **Y** по формуле $Y = x^5 \dfrac{\sin^4 x^3 + 2e^{3a} - tg(\frac{1-x}{1+x})}{\cos 4(x+a)}$ с использованием операторов консольных ввода и вывода данных. Код (текст) программы с комментариями приведен ниже. Переменные **x** и **a** зададим непосредственно в программном коде. Формулу для определения переменной **Y** запишем в виде, принятом в языке С++:

Y=pow(x,5)*(pow(sin(pow(x,3)),4)+2*exp(3*a)-tan((1-x)/(1+x)))/cos(4*(x+a)) .

Блок-схема алгоритма решения данного задания аналогична блок-схеме на рис. 1.1. Код (текст) программы с комментариями приведен ниже.

```
#include "stdafx.h"
#include <conio.h>                  // Файл conio.h обеспечивает задержку
                                    //окна DOS на экране дисплея
#include <iostream>                 //Директива include подключает файл
                                    //потоковых ввода и вывода  iostream
using namespace std;                //Подключает все имена из пространства
                                    //std
int _tmain(int argc, _TCHAR* argv[])      //Объявление главной функции
{                                   //Начало главной функции
double Y;                           //Объявление переменной Y
double a=3, x=0.3;                  //Объявление переменных a, x и их
                                    //инициализация
Y=pow(x,5)*(pow(sin(pow(x,3)),4)+2*exp(3*a)-tan((1-x)/(1+x)))/cos(4*(x+a));
     cout<<"Y = "<<Y;               //Вывод на экран значения Y
```

```
getch();                    //Функция задержки окна DOS на экране
return 0;
}
```

Результат вычислений: **Y = 48.8651 .**

Задание 1.5. Создайте проект для вычисления функции **Y** по заданной формуле в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 1.4. Ввод исходных данных организуйте непосредственно в программе (см. задание 4).

Перед разработкой программного кода запишите заданную формулу в отчете на языке C++.

Таблица 1.4 Исходные данные и формулы для расчета Y (Задание 1.5)

№ варианта	Формула для расчета **Y**	Значения **x, a, в**
1	$Y = \mathrm{tg}\frac{1-x}{1+x} + \sin^2 5x + e^{5a}$	x=0,5; a=3,5
2	$Y = \frac{\sin\frac{x+1}{4}}{\sin^2 5x + e^{3a}}$	x=0,7; a=1,5
3	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,82; a=2,55
4	$Y = \frac{\mathrm{tg}^3(x+a) - \arccos^2(x+a)}{(x+a)^4}$	x=0,68; a=5,55
5	$Y = \mathrm{ctg}\frac{1-3x}{1+2x} + \cos^2 5x + e^{7a}$	x=0,35; a=4,8
6	$Y = \frac{\cos^3(x+a) - 7(x+a)}{\mathrm{tg}(x+a)^4}$	x=0,62; a=4,55
7	$Y = \frac{\cos\frac{3a+1}{4}}{\sin^3 3x + e^{4a}}$	x=0,43; a=2,6
8	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,74; a=1,55
9	$Y = \frac{\mathrm{tg}\frac{4a^2+1}{4}}{\cos^3 2x + e^{2a}}$	x=0,14; a=2,55

10	$Y = \sin\frac{1-x}{1+x} + tg^4 5x + e^{5a}$	x=0,34; a=4,95
11	$Y = \frac{\sin^3(x+a) - \arccos^2(x+a)}{\cos(x+a)^4}$	x=0,14; a=2,95
12	$Y = \frac{ctg\frac{x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=0,75; a=1,9
13	$Y = \frac{ctg^3(3x+a) - \sin^2(x+7a)}{(5x+a)^3}$	x=0,44; a=2,95
14	$Y = \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 3x + e^{3a}}$	x=0,27; a=1,9
15	$Y = \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 3x + e^{3a}}$	x=0,49; a=3,7
16	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,83; a=4,7
17	$Y = \frac{arcctg\frac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=0,37; a=2,75
18	$Y = \frac{tg^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	x=0,13; a=0,7

Задание 1.6. Создайте проект для вычисления функции **Y** по заданной формуле в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 1.5. Ввод исходных данных организуйте непосредственно в программе (см. задание 4). Перед разработкой программного кода запишите заданную формулу в отчете на языке C++.

Таблица 1.5 Исходные данные и формулы для расчета Y (Задание 1.6)

№ варианта	Формула для расчета **Y**	Значения **x, a, в**
1	$Y = tg\frac{1-x}{1+x} + \sin^2 5x + e^{5a}$	x=0,5; a=3,5

2	$Y=\frac{1+\cos\frac{1-x^2}{1+x^2}}{\sqrt{1+\left(\frac{1-a^3}{1+a^4}\right)^2}}-e^{-\frac{x^2}{2}}$	x=1,573; a=1,775
3	$Y=\ln\left(1+\sqrt{1+x^2}\right)+\frac{1+x^2}{\sqrt{1+\frac{a^4}{2}}}$	x=1,573; a=1,775
4	$Y=\frac{\sin^3\left(a+x^2\right)}{\sqrt{a+\frac{x^3+1}{2}+\left(\frac{x^2+1}{2}\right)^5}}$	x=1,573; a=1,775
5	$Y=\left(a+\frac{a}{a^4+x^2}\right)-\sqrt{1+\frac{a}{a^2+x^2}}$	x=1,573; a=1,775
6	$Y=2a^3+\frac{2x^4}{\sqrt{1+a^2}}-e^{-\frac{a^2+1}{2}}$	x=1,573; a=1,775
7	$Y=\frac{e^{\frac{x^2}{2}}+e^{1+\frac{a^2}{2}}}{1+\frac{a^2}{2}+\left(\frac{x^4}{2}\right)^2}$	x=1,573; a=1,775
8	$Y=\lg\left(1+\sqrt{1+a^5}\right)+\frac{e^a+x^4}{\sqrt[4]{1+\frac{x^3}{2}}}$	x=1,573; a=1,775
9	$Y=\frac{(a+x)^3\cdot\ln\frac{a+x}{2}}{\sqrt[3]{1+\frac{\left(a+x^3\right)^2}{4}}}$	x=1,573; a=1,775
10	$Y=2x^3+\frac{2a^4}{\sqrt[3]{1+x^3}}-e^{-\frac{a^2+1}{2}}$	x=1,573; a=1,775

11	$Y = \sqrt[4]{\frac{1+\mathrm{tg}^2\frac{x+1}{4}}{1+\frac{a^3+1}{4}}} - e^{-\frac{x+1}{4}}$	x=1,573; a=1,775
12	$Y = \frac{e^{\frac{5x^2}{2}} + e^{1+\frac{4a^3}{2}}}{1+\frac{x^2}{2}+\left(\frac{a^3}{2}\right)^2}$	x=1,573; a=1,775
13	$Y = \frac{\lg\left(1+x^4\right)}{\sqrt{1+\frac{x^2+a^3}{2}+\left(\frac{x^2+1}{2}\right)^2}}$	x=1,573; a=1,775
14	$Y = \frac{1+\left(\frac{x}{a}\right)^{2,5} + \sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x^3} + \sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}$	x=1,573; a=1,775
15	$Y = \left(x^3 + \frac{a}{a^2+x^2}\right) - \sqrt{1+\frac{a}{a^2+x^4}}$	x=1,573; a=1,775
16	$Y = \sqrt[4]{\frac{1+\mathrm{tg}^2\frac{a^3+1}{4}}{1+\frac{x+1}{4}}} - e^{-\frac{a^2+1}{4}}$	x=1,573; a=1,775
17	$Y = \frac{1+\left(\frac{x^3}{a}\right)^{2,5} + \sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x} + \sqrt{1+\left(\frac{x}{a^3}\right)^{2,5}}}$	x=1,573; a=1,775
18	$Y = \ln\left(1+\sqrt{1+x^2}\right) + \frac{1+x^2}{\sqrt[3]{1+\frac{a^3}{2}}}$	x=1,573; a=1,775

Контрольные вопросы

1. Что такое проект в программе Visual Studio 2010?

2. Перечислите основные или базовые типы данных в языке С++.

3. Раскройте понятие переменной в языке С++.

4. Раскройте понятие выражения в языке С++.

5. Как записываются математические действия с переменными, константами и функциями в языке С++?
6. Каковы правила обращений к математическим функциям в языке С++?
7. Чем отличаются программы с линейной структурой?
8. Как вывести на экран значение переменной?
9. Как в одном операторе объявить тип переменной и задать ее значение?
10. Перечислите опции оператора **cout** и раскройте их действие.
11. Какие функции выполняет оператор **include**?
12. Какие функции выполняет файл **iostream**?

Лабораторная работа № 2

РАЗРАБОТКА И ИССЛЕДОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ ПРОГРАММ В VISUAL C++ 2010

Цель работы: исследование разветвляющихся программ и получение практических навыков в их разработке.

1. Исследование условного оператора if…else

Любой алгоритм может быть записан на языке программирования с использованием только трех управляющих структур: последовательное выполнение, ветвление и повторение. Последовательное выполнение реализуется в линейных алгоритмах, исследованных в предыдущей лабораторной работе. В данной работе будут исследованы разветвленные алгоритмы, реализующие алгоритмы ветвления.

На рис. 2.1 приведен пример структуры ветвления. Операционный блок **Условие** состоит из выражения, содержащего логическое отношение, т. е. условие. Если условие выполняется, то реализуется **Действие 1** алгоритма, а в случае невыполнения - **Действие 2**.

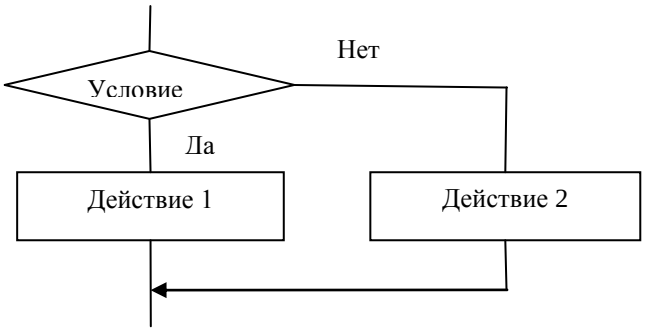


Рис.2.1. Структура ветвления

Структура ветвления описывается в языке C++ с помощью **условного оператора**:

```
if (выражение)
оператор_1;
else
оператор_2;
```

где часть **else оператор_2;** может отсутствовать.

Вначале вычисляется **выражение** в скобках. Если оно истинно, то выполняется **оператор_1**. Если **выражение** ложно, то **оператор_1** пропускается и выполняется **оператор_2**. Вместо единичных операторов могут использоваться группы из нескольких операторов (сложные операторы). При этом они заключаются в фигурные скобки - **{ }**.

Выражение в скобках представляет собой условие, заданное с помощью операций отношений и логических операций. В программировании постоянно приходится сравнивать числовые значения переменных, т. к. на этом построен

процесс принятия решений при работе над различными проектами. Для реализации этого в языке С++ существует шесть базовых операторов для сравнения значений двух переменных:

< меньше; **<= меньше или равно**;
> больше; **>= больше или равно**;
== равно; **!= не равно**.

Сложные операторы. К сложным операторам относят собственно сложные операторы и блоки. В обоих случаях это последовательность операторов, помещенная в фигурные скобки. Блок отличается от сложного оператора наличием **объявлений переменных** в теле блока. Например,

```
{
a=b+c;        // сложный оператор
d=a+b;
}

{
int a,b,c,d;
a=b+c;        // блок
d=a+b;
}
```

Для лучшего понимания действия операторов в настоящей работе следует знать перевод следующих английских слов:

if – если;
then – тогда;
else – иначе;
case – случай;
switch – переключатель;
break – прерывать;
default – не выполнять.

Задание 2.1. Исследовать и выполнить программу для определения переменной **a** по следующему условию:

$$a = \begin{cases} b + c + d, & \text{если} \quad b > 10; \\ b - c - d, & \text{если} \quad b \le 10. \end{cases}$$

Создайте проект **Lr2-1** в папке **d:\PE-11\Фамилия\Лр2** для разработки программы, производящей арифметические действия с тремя переменными **b**, **c** и **d** с использованием условного оператора **if…else** и операторов консольных ввода и вывода данных. Код (текст) программы с комментариями приведен ниже.

Блок-схема алгоритма решения задания 2.1 представлена на рис.2.2. Начертите ее в отчете по лабораторной работе.

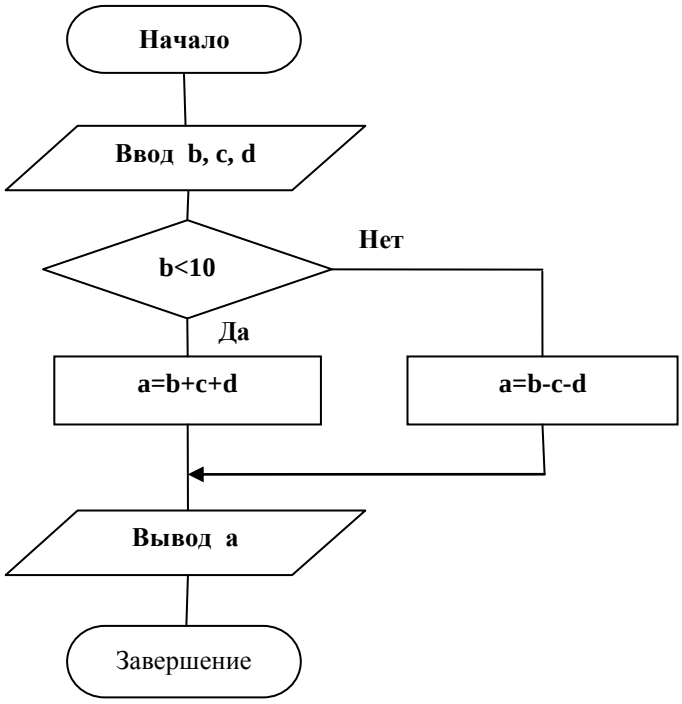


Рис. 2.2. Блок-схема алгоритма определения переменной **a** (простые операторы)

Введите и выполните программный код, реализующий алгоритм:

```
#include "stdafx.h"
#include <conio.h>                  //Файл conio.h обеспечивает задержку окна DOS
                                    //на экране дисплея
#include <iostream>                 //Директива include подключает файл ввода–
                                    //вывода iostream
using namespace std;                //Подключает все имена из пространства std
int _tmain(int argc, _TCHAR* argv[])    //Объявление главной функции _tmain
{                                   //Начало главной функции
int a, b;                           //Объявление переменных целого типа
int c=2, d=3;                       //Объявление переменных целого типа  и их
                                    //инициализация
cout<<"Vvvedite b"<<endl;           //Вывод на экран надписи  Vvedite b
cin>>b;                             //Ввод значения переменной  b
   if (b<10)                        //Условный оператор if …else (1-я часть)
      a=b+c+d;                      //Вычисление переменной a при выполнении
                                    //условия
   else                             //Условный оператор if …else (2-я часть)
      a=b-c-d;                      //Вычисление переменной a при невыполнении
                                    //условия
cout<<"a = "<<a;                    //Вывод на экран значения переменной  a
getch();                            //Функция задержки окна DOS на экране
return 0;
}
```

Проведите вычисления для **b=8** и **b=12**, а результаты сохраните следующим образом.

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр2**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 2.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr2-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr>**, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр2**.

Над вставленным рисунком проставьте номер задания – **2-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

Задание 2.2. Исследовать и выполнить программу для определения переменной **a** по условию **Задания 2.1**. Отличие будет состоять в использовании сложных операторов в условном операторе **if…else**.

Блок-схема алгоритма решения данного задания представлена на рис.2.3. Обратите внимание на отличия данной блок-схемы от схемы, приведенной на рис. 2.2. Объясните их. Начертите блок-схему в отчете по лабораторной работе.

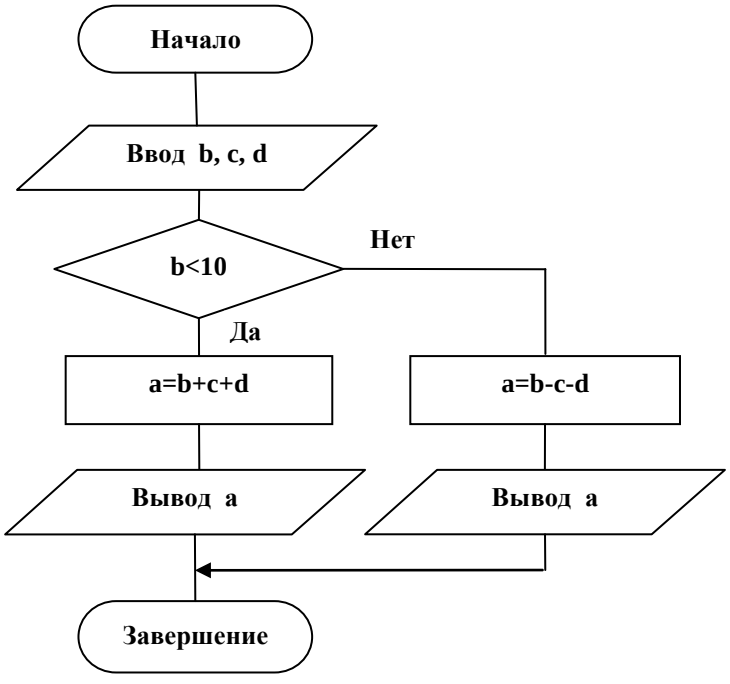


Рис. 2.3. Блок-схема алгоритма определения переменной **a** (сложные операторы)

Введите и выполните программный код, реализующий алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
#include "stdafx.h"
#include <conio.h>                    //Файл conio.h обеспечивает задержку окна
                                      //DOS  на экране дисплея
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])      //Объявление главной функции _tmain
{                                     //Начало главной функции
int a, b;                             //Объявление переменных целого типа
int c=2, d=3;                         //Объявление переменных целого типа  и их
                                      //инициализация
cout<<"Vvvedite b"<<endl;             //Вывод на экран надписи Vvedite b
cin>>b;                               //Ввод значения переменной  b
    if (b<10)                         //Условный оператор if…else (1-я часть)
    {                                 //Начало сложного оператора
       a=b+c+d;                       //Вычисление переменной a при выполнении
                                      //условия
       cout<<"a=b+c+d= "<<a;          //Вывод на экран значения переменной  a
    }                                 //Конец сложного оператора
    else                              //Условный оператор if …else (2-я часть)
    {                                 //Начало сложного оператора
       a=b-c-d;                       //Вычисление a при невыполнении условия
       cout<<"a=b-c-d= "<<a;          //Вывод на экран значения переменной  a
    }                                 //Конец сложного оператора
getch();                              //Функция задержки окна DOS на экране
return 0;
}
```

Проведите вычисления для **b=8** и **b=12**, а результаты сохраните в файле **Результат_Фамилия_Лр2** в папке **d:\РЕ-11\Фамилия\Лр2** в подобном рис.1.2 виде. Над вставленным рисунком проставьте номер задания – **2-2**. При выполнении следующих заданий сохраняйте результаты таким же образом.

Задание 2.3. Самостоятельно разработайте и выполните программу для определения переменной **Y** по следующему условию:

$$\begin{cases} Y = \sqrt[3]{1+\dfrac{\left(1+x^3\right)^2}{4}}, & \text{если} \quad x \ge 5; \\ Y = \sin\dfrac{1-x}{1+x} + tg^4 5x, & \text{если} \quad x < 5. \end{cases}$$

Используйте условный оператор **if…else** и оператор консольных ввода и вывода данных аналогично заданию 2.1. Обратите внимание на знаки отношений (>, >=,<=,…). Блок-схема алгоритма для решения данной задачи аналогична блок-схеме на рис. 2.1. Чертить ее необязательно. Результаты сохраните в файле **Результат_Фамилия_Лр2**.

Задание 2.4. Самостоятельно разработать алгоритм и программу для вычисления переменной **Y** (использовать оператор **if…else** и консольный ввод-вывод переменных). Исходные данные для решения задачи находятся в таблице 2.1. Определите свой номер варианта как номер компьютера. Значения переменной **x** задайте самостоятельно из обоих интервалов неравенства.

Таблица 2.1 Исходные данные и формулы для расчета Y

№ варианта	Формула для расчета **Y**	Значение **a**
1	$\begin{cases} Y = \operatorname{tg}\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}5; \end{cases}$	a=3,5
2	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \ge 0{,}7; \end{cases}$	a=2,55
3	$\begin{cases} Y = \operatorname{tg}\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}5; \end{cases}$	a=1,5
4	$\begin{cases} Y = \operatorname{ctg}\frac{1-x}{1+x} + \cos^2 5x, & \text{если } x < 0{,}3; \\ Y = \sin^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}3; \end{cases}$	a=1,44
5	$\begin{cases} Y = \sin\frac{1-x}{1+x} + \operatorname{ctg}^2 5x, & \text{если } x < 0{,}48; \\ Y = \cos^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}48; \end{cases}$	a=2,4
6	$\begin{cases} Y = \operatorname{ctg}^3(x+a) - \sin^2(x+a), & \text{если } x < 0{,}7; \\ Y = \operatorname{tg}^3(x+a) - \cos^2(x+a), & \text{если } x \ge 0{,}7; \end{cases}$	a=1,1
7	$\begin{cases} Y = \operatorname{tg}\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}5; \end{cases}$	a=2,6
8	$\begin{cases} Y = \operatorname{tg}\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}5; \end{cases}$	a=1,9
9	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \ge 0{,}7; \end{cases}$	a=3,1

10	$\begin{cases} Y = \operatorname{ctg}\dfrac{1-x}{1+x} + \cos^2 5x, & \text{если } x < 0{,}3; \\ Y = \sin^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}3; \end{cases}$	a=2,2
11	$\begin{cases} Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x < 0{,}37; \\ Y = \operatorname{ctg}^2 3x + \operatorname{tg}^3(x+a)e^{3a}, & \text{если } x \ge 0{,}37; \end{cases}$	a=1,1
12	$\begin{cases} Y = \operatorname{ctg}^2 3x + e^{3a}, & \text{если } x < 0{,}8; \\ Y = \sin^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}8; \end{cases}$	a=3,2
13	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \ge 0{,}7; \end{cases}$	a=1,2
14	$\begin{cases} Y = \operatorname{tg}\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}44; \\ Y = \cos^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}44; \end{cases}$	a=2,8
15	$\begin{cases} Y = \operatorname{tg}\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}5; \end{cases}$	a=3,5
16	$\begin{cases} Y = \operatorname{ctg}^2 3x + e^{3a}, & \text{если } x < 0{,}8; \\ Y = \sin^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}8; \end{cases}$	a=1,3
17	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \ge 0{,}7; \end{cases}$	a=1,7
18	$\begin{cases} Y = \operatorname{tg}\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}9; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}9; \end{cases}$	a=2,6
19	$\begin{cases} Y = \operatorname{ctg}\dfrac{1-x}{1+x} + \cos^2 5x, & \text{если } x < 0{,}4; \\ Y = \sin^3(x+a) - \operatorname{tg}^2(x+a), & \text{если } x \ge 0{,}4; \end{cases}$	a=1,9
20	$\begin{cases} Y = \operatorname{tg}\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}2; \\ Y = \operatorname{tg}^3(x+a) - \arccos^2(x+a), & \text{если } x \ge 0{,}2; \end{cases}$	a=2,0

2. Использование условного оператора if…else для 3-х интервалов значений переменой

Выше исследовались случаи применения условного оператора **if…else** для 2-х интервалов значений переменой, т. е. рассматривались простые условия

(логические выражения) типа **x<a.** При этом значение **a** делит числовую ось **x** на два интервала (рис. 2.4).

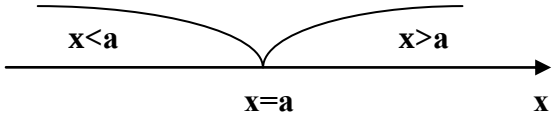


Рис. 2.4. Графическая интерпретация логического выражения для двух интервалов значений **x**

Язык C++ и среда Visual C++2010 позволяют в случае необходимости применять в условном операторе **if…else** и более сложные логические выражения с использованием различных логических операций.

Рассмотрим случай, когда в условии оператора **if…else** указан некоторый интервал значений, с которым сравнивается переменная. Например, переменная **x** должна находиться в интервале значений от **a** до **b** (**a<b**) (рис. 2.5). В этом случае условие запишется следующим образом:

if (x>=a && x<=b) ,

где **&&** - операция логического **И.**

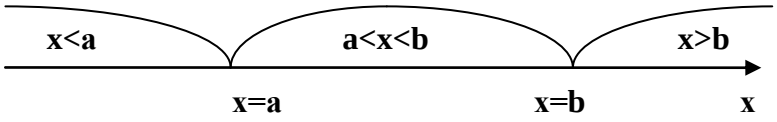


Рис. 2.5. Графическая интерпретация логического выражения для трех интервалов значений **x**

3. Исследование вложенного условного оператора if…else

Во многих случаях использование простого (одинарного) оператора **if…else** не обеспечивает решение поставленной задачи. Очевидно, что данный оператор, например, не позволяет вычислительному процессу принять решение в случае, когда для какой-то переменной существует четыре и более интервала ее значений. В этом случае применяют вложенный условный оператор **if…else**.

На рис. 2.6 приведен пример структуры вложенного условного оператора **if…else** (в качестве базовой использована структура, изображенная на рис. 2.1).

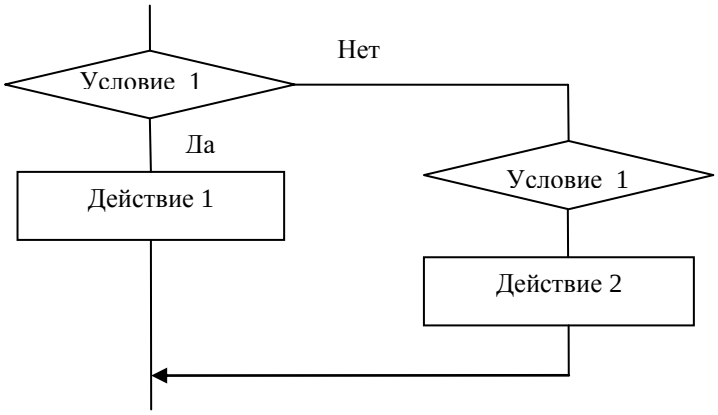


Рис.2.6. Структура вложенного условного оператора **if…else**

4. Оператор выбора switch

Оператор **switch** (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Он обеспечивает выбор из нескольких вариантов на основании некоторого фиксированного параметра. Формат оператора следующий:

switch (выражение)
{
case ***константа_1*****: оператор_1;**
case ***константа_2*****: оператор_2;**
case ***константа_3*****: оператор_3;**
...
case ***константа_n*****: оператор_n;**
[default: оператор;]
}

Блок-схема алгоритма, реализующего работу оператора выбора **switch**, приведена на рис. 2.7.

Выполнение оператора **switch** начинается с вычисления выражения в скобках (оно должно быть целочисленным). Его значение последовательно сравнивается с константами, которые записаны следом за каждым оператором **case**. Затем управление передается тому оператору, который помечен константным выражением (меткой), значение которого совпало с вычисленным выражением в условии. После этого последовательно выполняются все остальные ветви, если выход из переключателя явно не указан.

Все константные выражения должны иметь разные значения, но быть одного и того же целочисленного типа. Несколько меток могут следовать подряд. Если совпадения не произошло, выполняются операторы, расположенные после слова **default** (а при его отсутствии управление передается следующему за **switch** оператору).

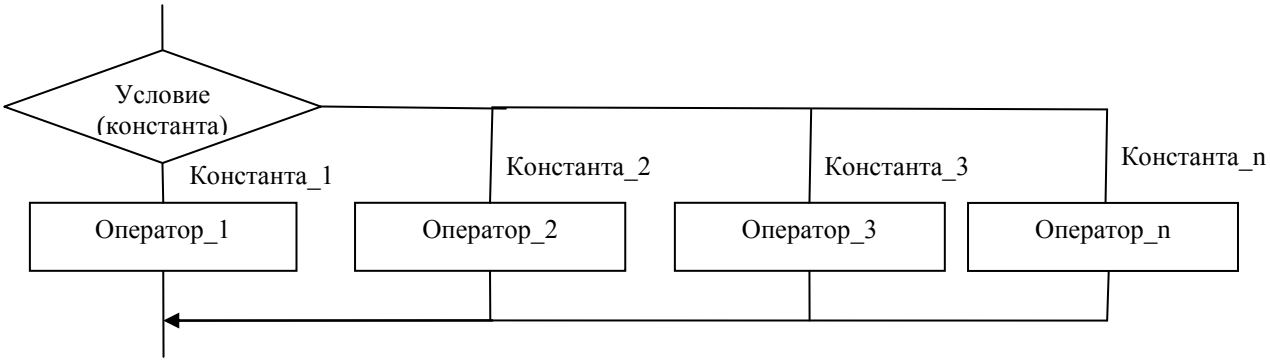


Рис. 2.7. Блок-схема алгоритма работы оператора **switch**

Задание 2.5. Проанализируем работу оператора **switch** на примере программы, реализующей простейший калькулятор на 4 действия. Здесь консольно вводятся две переменные **a** и **b** и знак операции, которую необходимо совершить с этими переменными.

Оператор **switch** сравнивает введенный знак операции со знаком, содержащемся в четырех метках **case**, и производит необходимую арифметическую операцию. Результат выводится на экран. Если знак операции не совпадает с содержащимися в метках, то на экран выводится сообщение о неизвестной операции.

Ниже приведен программный код, реализующий алгоритм решения задачи.

```
#include "stdafx.h"
#include <conio.h>                    //Файл conio.h обеспечивает задержку
                                      //окна DOS на экране дисплея
#include <iostream>                   //Директива include подключает файл
                                      //ввода–вывода iostream
using namespace std;                  //Подключает все имена из пространства
                                      //имен  std
int _tmain(int argc, _TCHAR* argv[])  //Объявление главной функции _tmain
{                                     //Начало главной функции
double a, b, res;                     //Объявление вещественных переменных
char operation;                       //Объявление символьной  переменной
cout << "Vvedite a : "<<endl;         //Вывод на экран надписи Vvedite a :
cin >> a;                             //Ввод значения переменной  a
cout << "Vvedite operation : "<<endl; //Вывод на экран надписи  Vvedite
                                      //symbol of operation :
cin >> operation;                     //Ввод знака операции
cout << "Vvedite b : "<<endl;         //Вывод на экран надписи Vvedite b :
cin >> b;                             //Ввод значения переменной  b
   switch (operation)                 //Условие (с чем будет сравнение) в
                                      //операторе  switch
   {                                  //Начало  составного оператора
      case '+': res = a + b; break;   //Метка ”+” и вычисление переменной
                                      //res для этой метки
```

```
        case '-': res = a - b; break;          //Метка ”-” и вычисление переменной res
                                               //для этой метки
        case '*': res = a * b; break;          //Метка ”*” и вычисление переменной
                                               //res для этой метки
        case '/': res = a / b; break;          //Метка ”/” и вычисление переменной res
                                               //для этой метки
        default : cout <<"Unknown operation"<<endl; //Вывод сообщения о
                                               //неизвестной операции
    }                                          //Конец сложного оператора
    cout << "Result : " << res;                //Вывод на экран результата вычислений
    getch();                                   //Функция задержки окна DOS на экране
    return 0;
    }                                          //Конец главной функции
```

5. Безусловный оператор goto

Переход к любому оператору программы без условия (директивный переход) в среде Visual C++2010 осуществляется с помощью оператора безусловного перехода **go to**. В отличие от оператора **if…else** этот оператор осуществляет переход в нужное место программы без выполнения каких-либо условий. Оператор имеет следующий вид:

goto Метка;

Меткой обозначают какой-либо оператор, на который должен быть осуществлен переход из места установки оператора **goto**. В качестве метки выступает идентификатор с расположенным за ним символом двоеточия:

Метка: оператор;

Как только выполнение программы достигает оператора **goto**, управление передается оператору, помеченному меткой A.

Задание 2.6. Проанализируем работу оператора **goto** на примере программы из задания 2.2. Расположим оператор **goto** с меткой **A** перед условным оператором, а метку **A** – перед концом программы.

Теперь условный оператор не выполнится, т. к. компьютер выполнит оператор **goto A** и перейдет сразу к выводу надписи **"Perehod po metke "**.

Ниже приведен программный код, реализующий алгоритм решения задачи.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{                                              //Начало главной функции
int a, b;                                      //Объявление переменных целого типа
int c=2, d=3;                                  //Объявление переменных целого типа и
                                               //их инициализация
cout<<"Vvvedite b"<<endl;                      //Вывод на экран надписи Vvedite b
cin>>b;                                        //Ввод значения переменной b
    goto A;                                    //Оператор goto
    if (b<10)
```

```
{
a=b+c+d;
cout<<"a=b+c+d= "<<a;
}
else
{
a=b-c-d;
cout<<"a=b-c-d= "<<a;
}
A: cout<<"Perehod po metke ";          //Метка A и оператор вывода
getch();                               //Функция задержки окна DOS на экране
return 0;
}
```

Задание 2.7. Самостоятельно разработайте алгоритм и программу решения задачи (использовать оператор **if…else**). Определите свой номер варианта как номер компьютера.

Варианты заданий

Вариант № 1. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Рекомендованный вид экрана:

Вычисление частного
Введите в одной строке делимое и делитель. Нажмите <Enter>
12 0
Вы ошиблись. Делитель не должен быть равен нулю

Вариант № 2. Написать программу вычисления площади кольца. Программа должна проверять правильность введенных данных. Рекомендованный вид экрана:

Вычисление площади кольца.
Введите начальные данные:
Радиус кольца (см) 3.5
Радиус отверстия (см) 7
Ошибка. Радиус отверстия не может быть больше радиуса кольца.

Вариант № 3. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки более 1000 грн. Рекомендованный вид экрана:

Вычисление стоимости покупки с учетом скидки 10%.
Введите сумму покупки: 1200
Сумма покупки: 1080.00 грн.

Вариант № 4. Написать программу вычисления стоимости покупки с учетом

скидки. Скидка в 3% предоставляется, если сумма покупки более 500 грн., в 5% -- если сумма более 800 грн. Рекомендованный вид экрана:

Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки: 640
Вам предоставляется скидка 3%
Сумма покупки: 620.80 грн.

Вариант №5. Написать программу для проверки знания даты начала второй мировой войны. В случае неправильного ответа, программа должна выводить правильный ответ. Рекомендованный вид экрана

В каком году началась вторая мировая война?
Введите число и нажмите <Enter>
1939
Правильно

Вариант № 6. Написать программу, которая сравнивает два введенных с клавиатуры числа. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Рекомендованный вид экрана:

Введите в одной строке два целых числа: 34 67
34 меньше 67

Вариант № 7. Написать программу, которая проверяет, является ли введенное целое число четное. Рекомендованный вид экрана:

Введите целое число: 23
Число 23 – нечетное.

Вариант № 8. Написать программу, которая проверяет, делится ли на три введенное с клавиатуры целое число. Рекомендованный вид экрана:

Введите целое число: 451
Число 451 нацело на три не делится.

Вариант № 9. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Рекомендованный вид экрана:

Вычисление частного
Введите в одной строке делимое и делитель. Нажмите <Enter>
12 0

Вы ошиблись. Делитель не должен быть равен нулю

Вариант № 10. Написать программу, которая вычисляет оптимальный вес, сравнивает его с реальным и выдает рекомендацию о необходимости пополнеть или похудеть. Оптимальный вес вычисляется по формуле: **рост (см) – 100**. Рекомендованный вид экрана:

Введите в одной строке через пропуск рост (см) и вес (кг): 170 68
Вам нужно пополнеть на 2,00 кг

Вариант № 11. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Рекомендованный вид экрана:

Вычисление частного
Введите в одной строке делимое и делитель. Нажмите <Enter>
12 0
Вы ошиблись. Делитель не должен быть равен нулю

Вариант № 12. Написать программу вычисления площади кольца. Программа должна проверять правильность введенных данных. Рекомендованный вид экрана:

Вычисление площади кольца.
Введите начальные данные:
Радиус кольца (с) 3.5
Радиус отверстия (см) 7
Ошибка. Радиус отверстия не может быть больше радиуса кольца.

Вариант № 13. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 10% предоставляется, если сумма покупки более 1000 грн. Рекомендованный вид экрана:

Вычисление стоимости покупки с учетом скидки 10%.
Введите сумму покупки: 1200
Сумма покупки: 1080.00 грн.

Вариант № 14. Написать программу вычисления стоимости покупки с учетом скидки. Скидка в 3% предоставляется, если сумма покупки более 500 грн., в 5% -- если сумма более 800 грн. Рекомендованный вид экрана:

Вычисление стоимости покупки с учетом скидки.
Введите сумму покупки: 640
Вам предоставляется скидка 3%
Сумма покупки: 620.80 грн.

Вариант № 15. Написать программу для проверки знания даты начала второй

мировой войны. В случае неправильного ответа, программа должна выводить правильный ответ. Рекомендованный вид экрана:

В каком году началась вторая мировая война?
Введите число и нажмите <Enter>
1939
Правильно

Вариант № 16. Написать программу, которая сравнивает два введенных с клавиатуры числа. Программа должна указать, какое число больше, или, если числа равны, вывести соответствующее сообщение. Рекомендованный вид экрана:

Введите в одной строке два целых числа: 34 67
34 меньше 67

Вариант № 17. Написать программу, которая проверяет, является ли введенное целое число четное. Рекомендованный вид экрана:

Введите целое число: 23
Число 23 – нечетное.

Вариант № 18. Написать программу, которая проверяет, делится ли на три введенное с клавиатуры целое число. Рекомендованный вид экрана:

Введите целое число: 451
Число 451 нацело на три не делится.

Вариант № 19. Написать программу, которая вычисляет частное двух чисел. Программа должна проверять правильность введенных данных и, если они неверные (делитель равен нулю), выдавать сообщение об ошибке. Рекомендованный вид экрана:

Вычисление частного
Введите в одной строке делимое и делитель. Нажмите <Enter>
12 0
Вы ошиблись. Делитель не должен быть равен нулю

Вариант № 20. Написать программу, которая вычисляет оптимальный вес, сравнивает его с реальным и выдает рекомендацию о необходимости пополнеть или похудеть. Оптимальный вес вычисляется по формуле: **рост (см) – 100.** Рекомендованный вид экрана:

Введите в одной строке через пропуск рост (см) и вес (кг): 170 68
Вам нужно набрать 2,00 кг

Контрольные вопросы

1. Какие операторы языка С++ используются для реализации разветвляющихся вычислительных процессов?
2. В каких случаях необходимо применение условных операторов?
3. Запишите условный оператор **if…else** в общем виде и опишите порядок его выполнения.
4. Чем отличается блок операторов от составного оператора?
5. Запишите в общем виде оператор **switch** и опишите порядок его выполнения.
6. Для чего необходим оператор **break**?
7. В каких случаях необходимо применение оператора **goto**?
8. Запишите в общем виде оператор **goto** и опишите порядок его выполнения

Лабораторная работа № 3

РАЗРАБОТКА И ИССЛЕДОВАНИЕ ЦИКЛИЧЕСКИХ ПРОГРАММ В VISUAL C++ 2010 (ОПЕРАТОРЫ ЦИКЛА WHILE И DO...WHILE)

Цель работы: получение практических навыков в разработке циклических программ с использованием операторов цикла **while**, **do...while**.

1. Циклические вычислительные процессы

Возможность повторно выполнять некоторые действия очень важна при разработке любых программ и программных приложений. Вычислительный процесс, содержащий многократные вычисления по одним и тем же математическим зависимостям, называется **циклическим.**

Цикл выполняет оператор или группу операторов до тех пор, пока истинно (или ложно) определенное условие относительно некоторой переменной, называемой **параметром цикла.**

Многократно повторяющиеся части такого процесса составляют **тело цикла.**

Алгоритм циклических структур должен содержать (рис. 3.1):

1. **Подготовку к циклу** – присваивание начального значения параметру цикла.
2. **Проверку условия** выполнения тела цикла.
3. **Тело цикла** – действия, которые выполняются в циклической программе для разных значений параметра цикла.
4. **Изменение (модификация) значений параметра цикла.**

На рис. 3.1 изображена блок-схема алгоритма циклического вычислительного процесса, где помимо характеристик операционных блоков в качестве примера приведены реальные операторы.

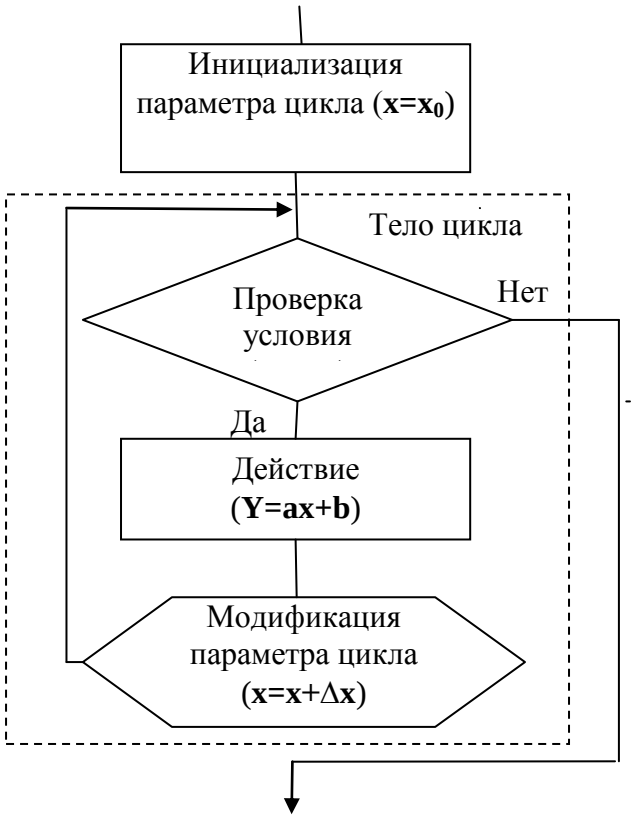


Рис. 3.1. Блок-схема алгоритма циклического вычислительного процесса

В среде Visual C++ 2010 циклические вычислительные процессы реализуются с помощью операторов **while**, **do...while** и **for**, анализ которых будет проведен ниже.

Для лучшего понимания действия операторов языка С++в настоящей работе следует знать перевод следующих английских слов:

do - делать, выполнять;
while - пока;
for - для.

2. Разработка и исследование программ с оператором цикла while в среде Visual C++ 2010

Оператор цикла **while** реализует вычислительную структуру **цикл с предусловием** и имеет следующий вид:

while (логическое выражение)
оператор;

В качестве **логического выражения** используется отношение или логическое выражение относительно параметра цикла в виде какого-то условия. Если оно истинно, то тело цикла (**оператор**) выполняется до тех пор, пока **выражение** не перестанет выполняться, т. е. не станет ложным. **Оператор** может быть простым или составным (из нескольких операторов, заключенных в фигурные скобки).

Блок-схема алгоритма циклического вычислительного процесса с предусловием, реализуемого оператором цикла **while**, приведена на рис. 3.1.

Исследуем простейшую программу с оператором цикла **while**.

Задание 3.1. Исследовать программу вычисления значения функции: **Y=ax + sin(πx)**, если значение x изменяется от $\mathbf{x_0}$ до $\mathbf{x_k}$ с шагом Δx, где $\mathbf{x_0}$=1, $\mathbf{x_k}$=2, а **Δx**=0,2. Блок-схема алгоритма решения данной задачи приведена на рис. 3.2. Необходимо ввести код программы, построить решение и произвести вычисления.

Определим типы и значения исходных данных, которые понадобятся для решения задачи:

a – константа, инициализируем ее, как **a** = 2 и определим ее тип как **int** (целое);

π – константа, ее инициализируем как **pi** = 3.14, тип **double** (действительное двойной точности);

$\mathbf{x_0}$ – переменная, обозначим ее как **xn**, тип **double**;

$\mathbf{x_k}$ – переменная, обозначим ее как **xk**, тип **double**;

Δx – переменная, обозначим ее как **dx**, тип **double**;

x – переменная, обозначим ее как **x**, тип **double**;

y – переменная, обозначим ее как **Y**, тип **double**.

Блок-схема алгоритма решения данного задания представлена на рис.3.2.

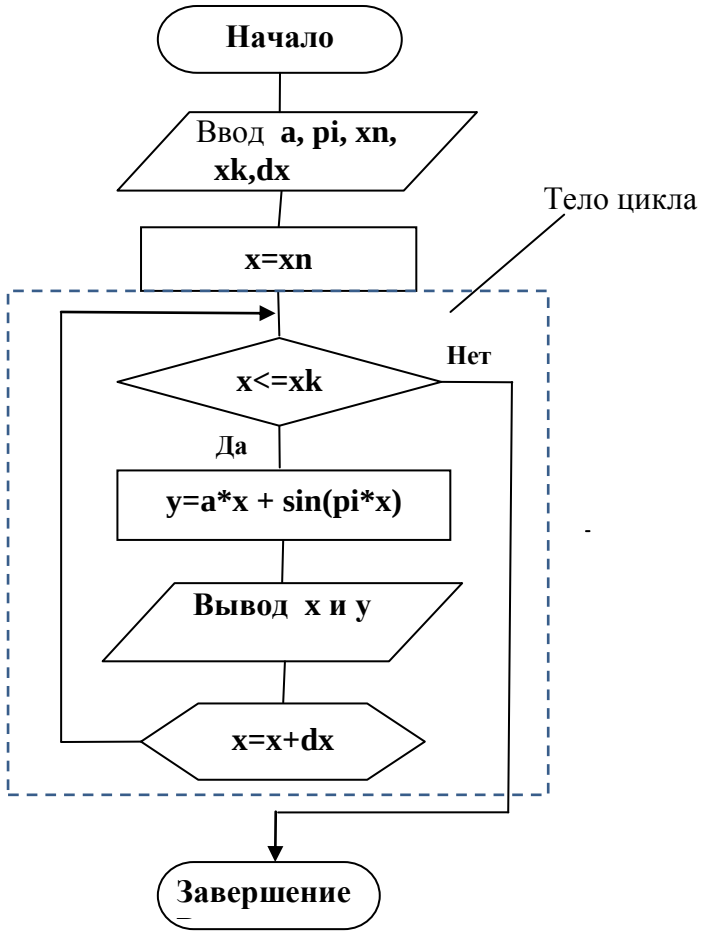


Рис. 3.2. Блок-схема алгоритма решения задания 3.1

Введите и выполните программный код, реализующий данный алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;
_tmain(int argc, _TCHAR* argv[])          //Объявление главной функции _tmain
{
    const int a=3;                        //Инициализация целой константы a
    const  double  pi=3.14;               //Инициализация действительной
    double x, y, xn, xk, dx;              //константы π, переменных и параметра
                                          //цикла
    cout<<" Vvedite xn, xk, dx: "<<endl; //Вопрос для ввода  параметров цикла
    cin>>xn>>xk>>dx;                      //Ввод  исходных данных
    cout<<"xn= "<<xn<<"  xk= "<<xk;       //Вывод исходных данных
    cout<<"  dx= "<<dx<<endl;             //Вывод исходных данных
    x=xn;                                 //Подготовка к циклу – присваивание
                                          //начального значения параметру цикла
```

```
                                        //переменной x
    cout<<" X   "<<"   Y  "<<endl;      //Вывод на экран заголовков "X" и "Y"
    while (x<=xk)                       //Оператор while (условие цикла x ≤ xk)
    {                                   //Начало составного оператора
        y=a*x + sin(pi*x);              //Вычисление действ. переменной y на
                                        //данном шаге
        cout<<x<<"   "<<y<<endl;       //Вывод на экран переменной y
        x=x+dx;                         //Вычисление  следующего значения
                                        //параметра цикла  x
    }                                   //Конец составного оператора
getch();                                //Функция задержки окна DOS на экране
return 0;
}                                       //Конец главной функции
```

В результате выполнения данной программы получим следующие результаты:

X	Y
1.0	3.0015
1.2	3.01376
1.4	3.24963
1.6	3.84816
1.8	4.8099
2.0	5.9968

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр3**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 3.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr3-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр3**.

Над вставленным рисунком проставьте номер задания – **3-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

Задание 3.2. Самостоятельно разработать алгоритм и программу для вычисления переменной **Y** (использовать оператор **while** и консольный ввод-вывод переменных). Определите свой номер варианта как номер компьютера.

Перед (!) разработкой программного кода начертите в отчете блок-схему алгоритма решения и запишите там же на языке С++ формулу для вычисления **Y**.

Таблица 3.1 Исходные данные и формулы для расчета Y (Задание 3.2)

№ варианта	Формула для расчета **Y**	Значения переменных
1	$Y = \operatorname{ctg}\frac{1-3x}{1+2x} + \cos^2 5x + e^{7a}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
2	$Y = \frac{\cos^3(x+a) - 7(x+a)}{\operatorname{tg}(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
3	$Y = \frac{\cos\frac{3a+1}{4}}{\sin^3 3x + e^{4a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
4	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
5	$Y = \frac{\operatorname{tg}\frac{4a^2+1}{4}}{\cos^3 2x + e^{2a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
6	$Y = \sin\frac{1-x}{1+x} + \operatorname{tg}^4 5x + e^{5a}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
7	$Y = \frac{\sin^3(x+a) - \arccos^2(x+a)}{\cos(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
8	$Y = \frac{\operatorname{ctg}\frac{x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
9	$Y = \frac{\operatorname{ctg}^3(3x+a) - \sin^2(x+7a)}{(5x+a)^3}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
10	$Y = \frac{\arcsin^3\frac{4x+1}{4}}{\operatorname{ctg}^2 3x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
11	$Y = \frac{\arccos^3\frac{4x+1}{4}}{\operatorname{tg}^2 3x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
12	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77

13	$Y = \dfrac{\operatorname{arcctg}\dfrac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
14	$Y = \dfrac{\operatorname{tg}^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
15	$Y = \operatorname{tg}\dfrac{1-x}{1+x} + \sin^2 5x + e^{5a}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
16	$Y = \dfrac{\sin\dfrac{x+1}{4}}{\sin^2 5x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
17	$Y = \dfrac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
18	$Y = \dfrac{\operatorname{tg}^3(x+a) - \arccos^2(x+a)}{(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77

3. Исследование оператора цикла do...while и разработка программ с его использованием

Оператор цикла **do...while** реализует вычислительную структуру **цикл с послеусловием** и имеет следующий вид:

do
оператор;
while (логическое выражение);

В качестве **логического выражения** используется отношение или логическое выражение относительно параметра цикла в виде какого-то условия. Если оно истинно, то тело цикла (**оператор**) выполняется до тех пор, пока **выражение** не перестанет выполняться, т. е. не станет ложным. **Оператор** может быть простым или составным (из нескольких операторов, заключенных в фигурные скобки). Основное отличие оператора цикла **do…while** от оператора **while** состоит в том, что здесь условие относительно параметра цикла проверяется после выполнения тела цикла. Поэтому оператор **do…while** выполняется как минимум один раз. Блок-схему вычислительного процесса, реализующего оператор цикла **do…while**, проанализируем на примере решения **Задания 3.3** (рис. 3.3).

Задание 3.3. Исследовать программу вычисления значения функции

$$Y = \sin^3(x-a) - \cos^2(x+a)^3$$

для значений аргумента **x** от $\mathbf{x_0}$=0 до $\mathbf{x_k}$=1 с шагом **Δx**=0,2.

Для решения задачи необходимо использовать оператор **do…while** и консольный ввод-вывод переменных. Блок-схема алгоритма решения данной

задачи приведена на рис. 3.3. Необходимо ввести код программы, построить решение и произвести вычисления.

Определим типы и значения исходных данных, которые понадобятся для решения задачи:

a – константа, инициализируем ее, как **a** = 0,3 и определим ее тип как **double** (действительное двойной точности);

$\mathbf{x_0}$ – переменная, обозначим ее как **xn**, тип **double**;

$\mathbf{x_k}$ – переменная, обозначим ее как **xk**, тип **double**;

Δx – переменная, обозначим ее как **dx**, тип **double**;

x – переменная, обозначим ее как **x**, тип **double**;

y – переменная, обозначим ее как **Y**, тип **double**.

Блок-схема алгоритма решения данного задания представлена на рис.3.3.

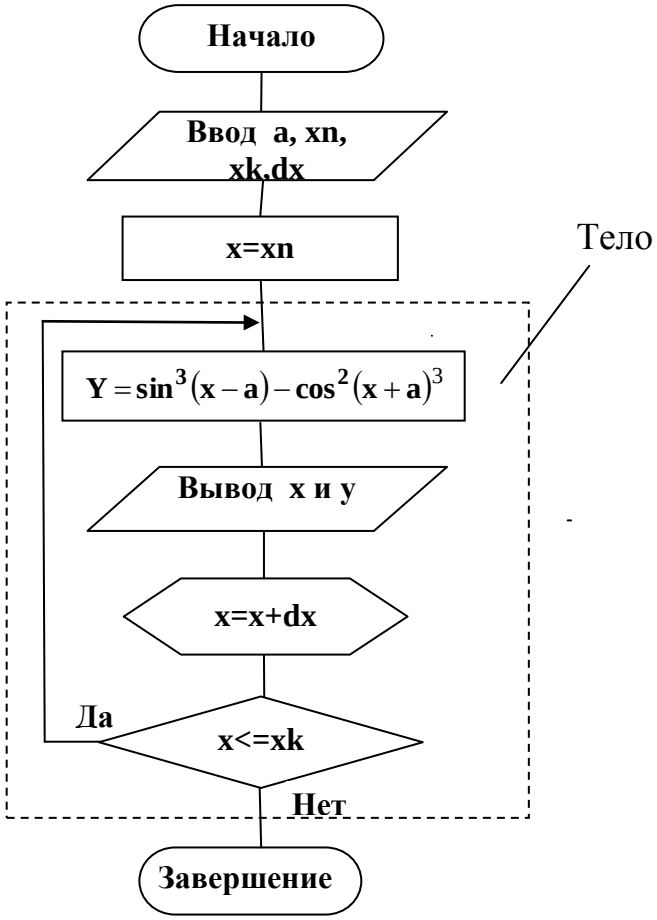


Рис. 3.3. Блок-схема алгоритма решения задания 3.3

Введите и выполните программный код, реализующий данный алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

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

```
int _tmain(int argc, _TCHAR* argv[])
    {
    double a=0.3;
    double x, y, xn, xk, dx;
    cout<<" Vvedite xn, xk, dx: "<<endl;     //Вопрос для ввода  параметров цикла
    cin>>xn>>xk>>dx;                          //Ввод  исходных данных
    cout<<"xn= "<<xn<<"  xk= "<<xk;          //Вывод исходных данных
    cout<<"  dx= "<<dx<<endl;                //Вывод исходных данных
    x=xn;                                    //Присваивание начального
                                             //значения параметру цикла x
    cout<<" X   "<<"   Y  "<<endl;           //Вывод на экран заголовков "X" и
"Y"
       do                                    //Начало оператора do…while
       {                                     //Начало составного оператора
       y=pow(sin(x-a),3)-pow(pow(cos(x+a),3),2);    //Вычисление y на данном шаге
          cout<<x<<"   "<<y<<endl;          //Вывод на экран y
          x=x+dx;                            //Вычисление значения x для
                                             //следующего шага
       }                                     //Конец составного оператора
       while (x<=xk);                        //Оператор while (условие цикла x ≤ xk)
  getch();                                  //Функция задержки окна DOS на экране
  return 0;
  }                                         //Конец главной функции
```

В результате выполнения данной программы получим следующие результаты:

```
X       Y
0       -0.786027
0.2     -0.457797
0.4     -0.19919
0.6     -0.0318825
0.8      0.101485
1        0.266995
```

Задание 3.4. Самостоятельно создайте проект для вычисления функции **Y** по заданной формуле в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 3.2. Ввод исходных данных организуйте непосредственно в программе. Тип исходных данных – **double**.

Перед **(!)** разработкой программного кода начертите в отчете блок-схему алгоритма решения и запишите там же на языке С++ формулу для вычисления Y. Результаты сохраните в файле **Результат_Фамилия_Лр3** в папке **d:\PE-11\Фамилия\Лр3** в подобном рис. 1.2 виде.

Таблица 3.2 Исходные данные и формулы для расчета Y (Задание 3.4)

№ варианта	Формула для расчета **Y**	Значения

		переменных
1	$Y=\left(a+\frac{a}{a^4+x^2}\right)-3\sqrt{1+\frac{a}{a^2+x^2}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
2	$Y=2a^3+\frac{2x^4}{\sqrt{1+a^2}}-xe^{-\frac{a^2+1}{2}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
3	$Y=\frac{e^{\frac{x^2}{2}}+e^{1+\frac{a^2}{2}}}{1+\frac{a^2}{2}+\left(\frac{x^4}{2}\right)^2}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
4	$Y=\lg\left(1+\sqrt{1+a^5}\right)+\frac{e^a+x^4}{\sqrt[4]{1+\frac{x^3}{2}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
5	$Y=\frac{(a+x)^3-\ln\frac{a+x}{2}}{\sqrt[3]{1+\frac{\left(a+x^3\right)^2}{4}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
6	$Y=2x^3+\frac{2a^4}{\sqrt[3]{1+x^3}}\cdot e^{-\frac{a^2+1}{2}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
7	$Y=\sqrt[4]{\frac{1+tg^2\frac{x+1}{4}}{1+\frac{a^3+1}{4}}}\cdot e^{-\frac{x+1}{4}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
8	$Y=\frac{e^{\frac{5x^2}{2}}+e^{1+\frac{4a^3}{2}}}{1+\frac{x^2}{2}+\left(\frac{a^3}{2}\right)^2}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
9	$Y=\frac{\lg\left(1+x^4\right)}{\sqrt{1+\frac{x^2+a^3}{2}+\left(\frac{x^2+1}{2}\right)^2}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77

10	$Y=\frac{1+\left(\frac{x}{a}\right)^{2,5}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x^3}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
11	$Y=\left(x^3+\frac{a}{a^2+x^2}\right)-\sqrt{1+\frac{a}{a^2+x^4}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
12	$Y=\sqrt[4]{\frac{1+tg^2\frac{a^3+1}{4}}{1+\frac{x+1}{4}}}-e^{-\frac{a^2+1}{4}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
13	$Y=\frac{1+\left(\frac{x^3}{a}\right)^{2,5}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x}+\sqrt{1+\left(\frac{x}{a^3}\right)^{2,5}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
14	$Y=\ln\left(1+\sqrt{1+x^2}\right)+\frac{1+x^2}{\sqrt[3]{1+\frac{a^3}{2}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
15	$Y=tg\frac{1-x}{1+x}+\sin^2 5x+e^{5a}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
16	$Y=\frac{1+\cos\frac{1-x^2}{1+x^2}}{\sqrt{1+\left(\frac{1-a^3}{1+a^4}\right)^2}}-e^{-\frac{x^2}{2}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
17	$Y=\ln\left(1+\sqrt{1+x^2}\right)+\frac{1+x^2}{\sqrt{1+\frac{a^4}{2}}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
18	$Y=\frac{\sin^3\left(a+x^2\right)}{\sqrt{a+\frac{x^3+1}{2}+\left(\frac{x^2+1}{2}\right)^5}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77

4. Программирование вложенных циклов в Visual C++ 2010

Циклы можно вкладывать друг в друга. Это следует из записи в общем виде, например, оператора цикла **while**:

while (логическое выражение)
оператор;

где **оператор** – любой оператор, в том числе и сложный, состоящий из нескольких операторов. Т. к. исключений из этого правила в языке C++ нет, то в качестве оператора может быть использован любой из операторов цикла.

Тогда вложенный цикл с оператором **while** будет иметь следующий вид:

while (логическое выражение)
{
while (логическое выражение)
оператор;
},

где **оператор** – любой оператор, в том числе и сложный.

Проанализируем работу программы в Visual C++ 2010, использующей такую вычислительную конструкцию, как вложенный цикл, или цикл в цикле.

Задание 3.5. Исследовать программу вычисления значения функции

$$s = \sum_{n=1}^{4} \sum_{k=1}^{5} \sin^3(knx - a)$$

для некоторой переменной **x** (использовать вложенный цикл с оператором **while** и консольный ввод-вывод переменных).

Блок-схема алгоритма решения данной задачи приведена на рис. 3.4. Необходимо ввести код программы, построить решение и произвести вычисления.

Определим типы и значения исходных данных, которые понадобятся для решения задачи:

a – константа, инициализируем ее, как **a** = 1,4 и определим ее тип как **double** (действительное двойной точности);

x – переменная, обозначим ее как **x**, тип **double**;

y – переменная, обозначим ее как **Y**, тип **double**;

s – переменная, обозначим ее как **S**, тип **double**.

Введите и выполните программный код, реализующий данный алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
```

```
{
const double a=1.4;                  //Инициализация константы a
double x, y, n, k, s;                //Инициализация переменных и
                                     //параметров цикла
cout<<" Vvedite x "<<endl;           //Вопрос для ввода  x
cin>>x;                              //Ввод  переменной x
```

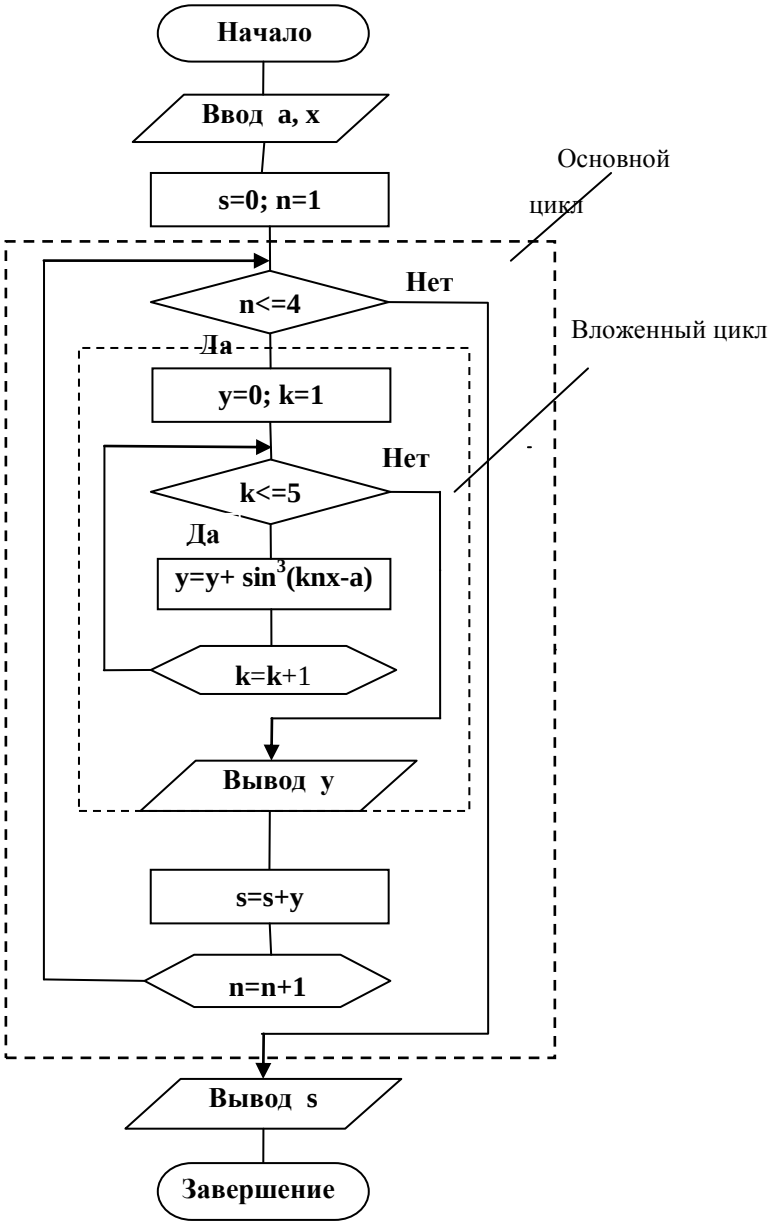


Рис. 3.4. Блок-схема алгоритма решения задания 3.5
с использованием вложенного цикла

```
cout<<"x= "<<x<<"  a= "<<a<<endl; //Вывод исходных данных
n=1;                                 //Присваивание начального значения
                                     //параметру цикла - переменной n
s=0;                                 //Начальное значение переменной s
                                     //для суммирования
while (n<=4)                         //Оператор while (условие цикла n ≤ 4)
```

```
{                                   //Начало составного оператора
   y=0;                             //Начальное значение переменной y
                                    //для суммирования
   k=1;                             //Присваивание начального значения
                                    //параметру цикла - переменной k
   while (k<=5)                     //Вложенный оператор while
   {                                //Начало составного оператора
      y=y+ pow(sin(k*n*x-a),3);     //Суммирование y на k-ом шаге
      cout<<k<<"   y =  "<<y<<endl; //Вывод на экран  y на k-ом шаге
      k=k+1;                        //Вычисление  следующего значения
                                    //параметра цикла  k
   }                                //Конец составного оператора
      s=s+y;                        //Суммирование s на n-ом шаге
      cout<<n<<"   s =  "<<s<<endl; //Вывод на экран  s на k-ом шаге
      n=n+1;                        //Вычисление следующего значения n
   }                                //Конец составного оператора с
                                    //вложенным циклом
getch();                            //Функция задержки окна DOS на экране
return 0;
}                                   //Конец главной функции
```

Результаты вычислений приведены на рис. 3.5. Промежуточные результаты вычислений выводятся на экран для дополнительного контроля правильности работы программы (тестирования программы).

```
 Vvedite x
1.4
x= 1.4   a= 2.3
1    y =   -0.48065
2    y =   -0.370455
3    y =   0.476942
4    y =   0.473016
5    y =   -0.526753
1    s =   -0.526753

1    y =   0.110195
2    y =   0.10627
3    y =   0.100225
4    y =   0.225993
5    y =   -0.21643
2    s =   -0.743183

1    y =   0.847396
2    y =   0.841352
3    y =   0.388923
4    y =   1.20605
5    y =   1.20274
3    s =   0.459557

1    y =   -0.0039253
2    y =   0.121842
3    y =   0.938967
4    y =   1.79397
5    y =   1.94911
4    s =   2.40866
```

Рис. 3.5. Результаты вычислений для **Задания 3.5**

Задание 3.6. Самостоятельно создайте проект для вычисления функции **Y** по

заданной формуле в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 3.3. Ввод исходных данных и вычисления организуйте по аналогии с **Заданием 3.5**.

Перед (!) разработкой программного кода начертите в отчете блок-схему алгоритма решения и запишите формулу для вычислений на языке С++.

Таблица 3.3 Исходные данные и формулы для расчета Y (Задание 3.6)

№ варианта	Формула для расчета **Y**	Значения **x и a**
1	$s=\sum_{n=1}^{6}\sum_{k=1}^{5}\sin^4\left(knx^3-a\right)$	x=0,7; a=1.7
2	$s=\sum_{n=1}^{6}\sum_{k=1}^{3}\sqrt[4]{\left(knx^3-ak\right)}$	x=0,4; a=1.7
3	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^3\left(knx^5-a\right)$	x=0,6; a=1.7
4	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[3]{\left(knx^2-a\right)}$	x=0,8; a=1.7
5	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^5\left(knx^3-an\right)$	x=0,4; a=1.7
6	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{\left(knx^4-ak\right)}$	x=0,5; a=1.7
7	$s=\sum_{n=1}^{6}\sum_{k=1}^{4}tg^3\left(knx^5-ax\right)$	x=0,6; a=1.7
8	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-an\right)}$	x=0,4; a=1.7
9	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}\sin^4\left(knx^6-ax\right)$	x=0,8; a=1.7
10	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}tg^3\left(knx^2-a\right)$	x=0,2; a=1.7
11	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-ax\right)}$	x=0,4; a=1.7
12	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^4\left(knx^3-a\right)$	x=0,3; a=1.7
13	$s=\sum_{n=1}^{3}\sum_{k=1}^{7}\sqrt[3]{\left(knx^5-an\right)}$	x=0,5; a=1.7
14	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-ak\right)}$	x=0,9; a=1.7

15	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^3\left(knx^6-an\right)$	x=0,4; a=1.7
16	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\cos^4\left(knx^3-ak\right)$	x=0,5; a=1.7
17	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{\left(knx^2-an\right)}$	x=0,4; a=1.7
18	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\mathrm{tg}^4\left(knx^3-an\right)$	x=0,6; a=1.7

Контрольные вопросы

1. С помощью каких действий реализуются циклические вычислительные процессы в языке С++?
2. Запишите в общем виде оператор цикла **while** и опишите, как он выполняется.
3. Запишите в общем виде оператор цикла **do... while** и опишите, как он выполняется.
4. Перечислите, что должно быть в конструкции цикла.
7. Сколько раз выполнится оператор цикла

```
int i=1;
while(i>3)
i=i+1;
```
?

8. Сколько раз выполнится оператор цикла

```
int x=1;
do x=x+1;
while(x>3);
```
?

9. Сколько раз выполнится оператор цикла

```
int i=1; while(i<3)  i=i+1;
```
?

10. Сколько раз выполнится оператор цикла

```
int x=3;
while(x>1)
x=x+1;
```
?

11. Сколько раз выполнится оператор цикла

```
int x=1;
while(x<=3)
x=x+1;
```
?

12. Чему будет равняться **x** после выхода из цикла

```
int x=1;
do  x=x+1;
while(x<3);
```
?

Лабораторная работа № 4
РАЗРАБОТКА И ИССЛЕДОВАНИЕ ЦИКЛИЧЕСКИХ ПРОГРАММ В VISUAL C++ 2010 (ОПЕРАТОР ЦИКЛА FOR)

Цель работы: получение практических навыков в разработке циклических программ с использованием оператора цикла **for**.

1. Циклические вычислительные процессы

Возможность повторно выполнять некоторые действия очень важна при разработке любых программ и программных приложений. Вычислительный процесс, содержащий многократные вычисления по одним и тем же математическим зависимостям, называется **циклическим**.

Цикл выполняет оператор или группу операторов до тех пор, пока истинно (или ложно) определенное условие относительно некоторой переменной, называемой **параметром цикла**.

Многократно повторяющиеся части такого процесса составляют **тело цикла**.

Алгоритм циклических структур должен содержать (рис. 4.1):

1. **Подготовку к циклу** – присваивание начального значения параметру цикла.
2. **Проверку условия** выполнения тела цикла.
3. **Тело цикла** – действия, которые выполняются в циклической программе для разных значений параметра цикла.
4. **Изменение (модификация) значений параметра цикла.**

На рис. 4.1 изображена блок-схема алгоритма циклического вычислительного процесса, где помимо характеристик операционных блоков в качестве примера приведены реальные операторы.

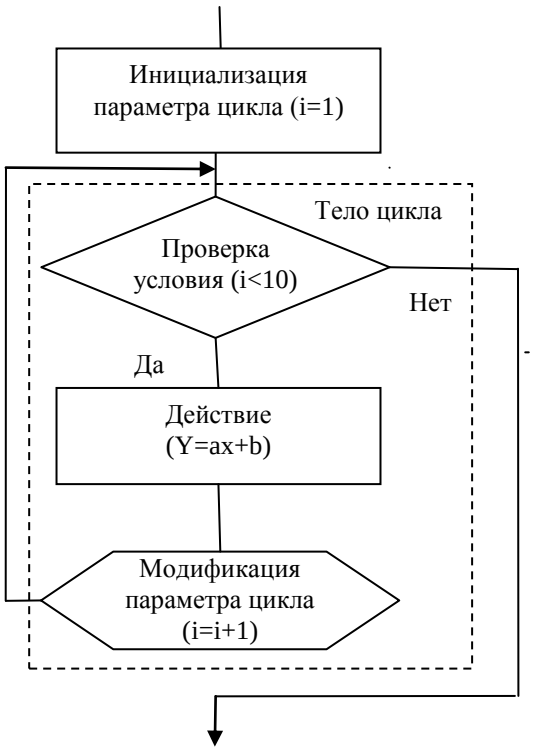


Рис. 4.1. Блок-схема алгоритма циклического вычислительного процесса

В среде Visual C++ 2010 циклические вычислительные процессы реализуются с помощью операторов **while**, **do...while** и **for**. Выше были исследованы операторы **while** и **do...while**. В настоящей работе будут исследованы программы с использованием оператора **for**.

2. Оператор цикла for

Оператор цикла **for** используется, когда количество повторений тела цикла **заранее известно**. Форма записи оператора цикла **for** следующая:

for ([выражение инициализации]; [выражение проверки (условие)];
[выражение модификации])
оператор внутри цикла;

Квадратные скобки показывают, что данная секция в операторе может быть опущена.

На практике это выглядит, например, следующим образом:

```
for(i=1; i<=n; i=i+1)
Y =a*i;
```

где **i** – параметр цикла (в данном случае - номер шага вычислений).

Анализ данной записи показывает, что оператор **for** объединяет в себе три операционных блока из блок-схемы циклического вычислительного процесса (рис. 4.1):

- блок инициализации, т. е. присвоения параметру цикла начального значения (**i=1**);
- блок проверки условия (**i<=n**);
- блок модификации параметра цикла (**i=i+1**).

Это свойство оператора цикла **for** позволяет существенно упростить вычислительные процессы и программные коды при решении различных задач в Visual C++ 2010.

В схемах алгоритмов оператор цикла **for** отражается символом **модификация** (рис. 4.2):

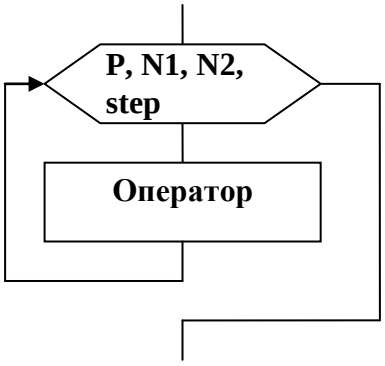


Рис. 4.2. Блок-схема оператора цикла **for**

На схеме алгоритма приведены следующие обозначения:

- **P** – параметр цикла;
- **N1, N2** – границы изменения параметра цикла;

- step – шаг изменения параметра цикла (если шаг не указан, то он равняется 1)
.

Параметр цикла **P**, границы изменения **N1**, **N2** и шаг **step** должны иметь один и тот же тип данных.

В языке C++ принято операцию инкремента (приращения цикла) **i=i+1**, записывать как **i++**, например:

```
for(i=1; i<=n; i++)
f =f*i;
```

Возможности оператора цикла **for** очень велики. Например, вместо любого из трех выражений в записи общей формы можно записать два и более выражения, разделенных запятыми:

```
for(i=1, j=1, z=1; i<=n; i++, j++, z++)
f =f*i*j*z;
```

Рассмотрим простейший пример применения оператора цикла **for**.

Задание 4.1. Исследовать программу для печати чисел от 1 до заданного числа **n**=15 с шагом 1. Использовать оператор **for**. Блок-схема алгоритма выполнения такого задания приведена на рис. 4.3.

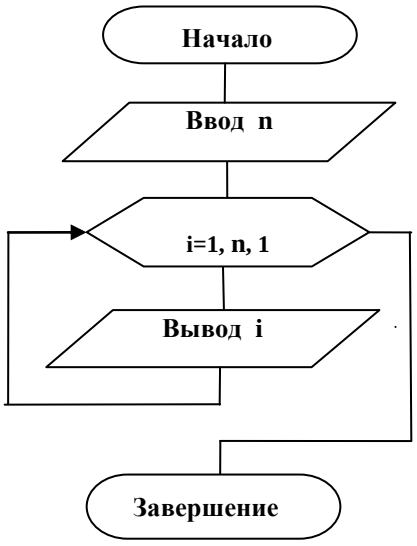


Рис. 4.3. Блок-схема алгоритма для печати в столбец **n** чисел

Ниже приведен программный код, реализующий данный алгоритм:

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

```
int _tmain(int argc, _TCHAR* argv[])
{
int i, n ;
cout<<"Vvedite n "<<endl;
cin>>n;                                  //Ввод значения переменной  n
   for ( i=1; i< =n; i++ )               //Оператор цикла  for
   cout << i<<"  ";                      //Тело цикла – вывод параметра цикла i
getch();
return 0;
}
```

В результате выполнения программы будут напечатаны положительные числа от 1 до 15:

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15.

Анализ результатов показывает, что работа оператора цикла **for** происходит в полном соответствии со схемой вычислительного процесса на рис. 4.1.

В данном примере параметром цикла **for** является переменная **i** (она же – счетчик). В начале цикла счетчик (переменная **i**) инициализируется значением 1. Затем выполняется тело цикла (**cout << i <<endl;**) и проверяется, не достиг ли счетчик значения **n=**15. После каждого выполнения тела цикла счетчик (**i**) увеличивается на единицу. Как только **i** станет равным 15, тело цикла пропускается и управление передается следующему оператору программы.

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр6**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 4.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr4-1.cpp** (см. рис. 1.2) и нажмите клавишу <**Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр4**. Над вставленным рисунком проставьте номер задания – **4-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

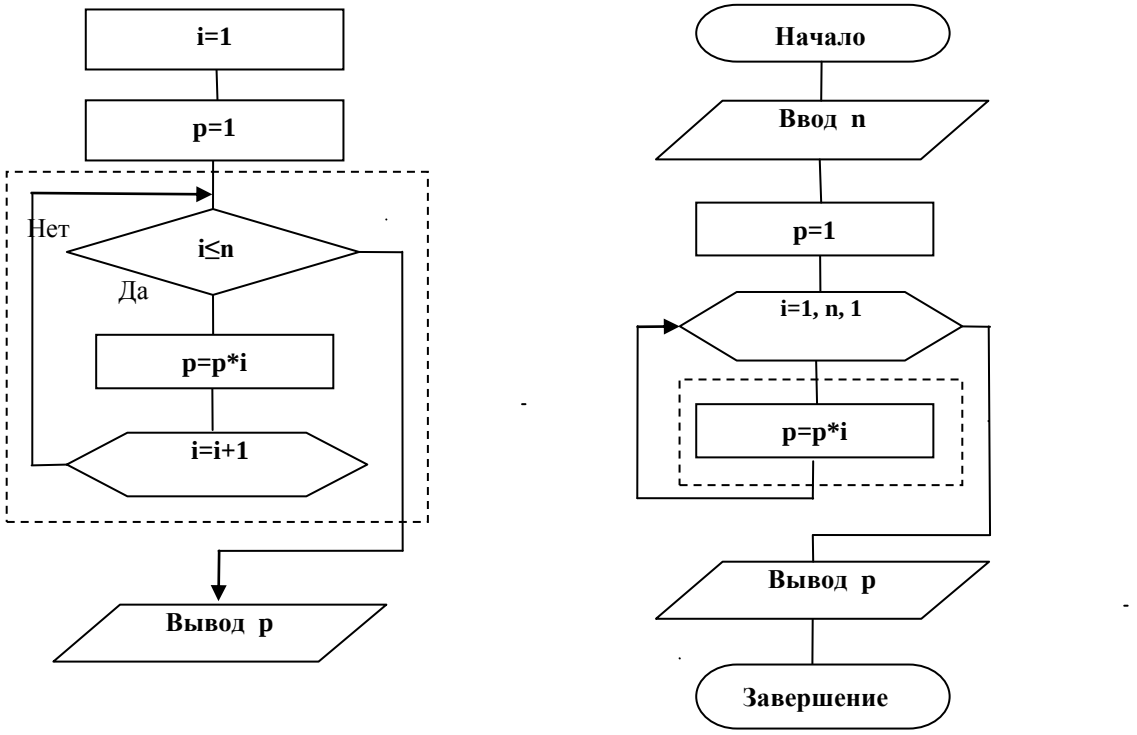
!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

Задание 4.2. Исследовать программу вычисления факториала целого числа **n** с использованием оператора **for**. **Факториал** – это произведение целых чисел от 1 до **n**. Необходимо проанализировать блок-схемы вычислительного процесса и алгоритма решения задания при помощи оператора цикла **for**, ввести код программы, построить решение и произвести вычисления для n=10.

Факториал числа **n** вычисляется по следующей формуле:

$$\mathbf{n! = 1*2*3*4* \ldots *n} = \prod_{i=1}^{n} i .$$

При вычислении факториала начальному значению произведения П необходимо присвоить 1. При каждом выполнении тела цикла $П_{i-1}$ будем умножать на **i**. Примем для обозначения произведения идентификатор **p**. Блок-схемы вычислительного процесса и алгоритма решения задания при помощи оператора цикла **for** имеют следующий вид:



а) б)

Рис. 4.4. Блок-схемы вычисления факториала целого числа **n**:
а) вычислительного процесса; б) алгоритма с использованием оператора **for**

Программу вычисления факториала можно записать в следующем виде:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, n, p;
cout<<"Vvedite N"<<endl;
cin>>n;                                  //Ввод значения переменной  n
p =1;                                    //Начальное значение переменной
                                         //p для произведения
   for (i=1; i<=n; i++)                  //Оператор цикла  for
   p =p*i;                               //Тело цикла – произведение p и  i
cout<<"faktorial " <<n<<"   =  "<<p<<endl; //Печать результата
getch();
return 0;
}
```

После запуска программы на экране появится результат:

Vvedite celoe chislo: 5
factorial 5 = 120

Обратите внимание на необходимость присваивания начального значения (единица) переменной **p** для вычисления последующих произведений.

Задание 4.3. Исследовать программу для вычисления суммы **n** четных чисел, начиная от двух. Использовать оператор **for**. Формула для вычисления такой суммы имеет следующий вид:

$$s = \sum_{i=1}^{n} (2 * i),$$

где **i** – параметр цикла (номер четного числа на данном шаге).

Блок-схема алгоритма решения такой задачи приведена на рис. 4.5. Обратите внимание на необходимость присваивания начального значения (ноль) переменной **s** для вычисления последующих сумм.

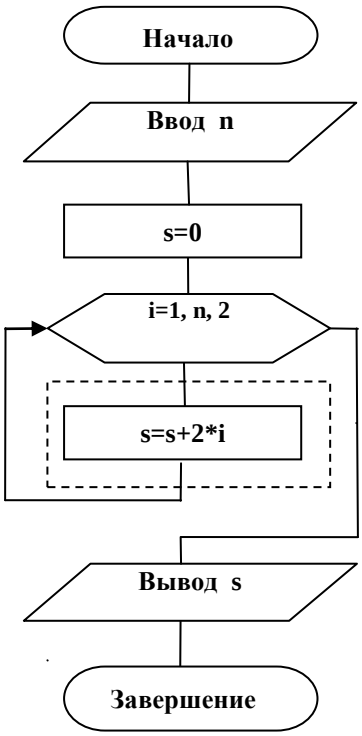


Рис. 4.5. Блок-схема алгоритма для вычисления суммы **n** четных чисел

Программу вычисления суммы **n** четных чисел можно записать в следующем виде

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
```

```
{
int i, n, s;
cout<<"Vvedite N"<<endl;
cin>>n;                                  //Ввод значения переменной  n
s =0;                                    //Начальное значение s длясуммирования
      for (i=1; i<=n; i++)               //Оператор цикла  for
      s=s+2*i;                           //Тело цикла – расчет суммы
cout<<"Summa  " <<2*n<<"   =  "<<s<<endl;     //Печать результата
getch();
return 0;
}
```

После запуска программы на экране появится результат:

```
Vvedite celoe chislo: 5
Summa 10   = 30
```

Задание 4.4. Исследовать программу вычисления функции

$$Y = \sum_{k=1}^{6} \left(\cos^3 kx + \sin \frac{0{,}5x}{\sqrt[3]{k^2 + x^2 + 1}} \right)$$

для **x**=1,32. Блок-схема алгоритма для вычисления данной функции аналогична блок-схеме алгоритма для вычисления суммы **n** четных чисел на рис. 4.5. Необходимо ввести код программы, построить решение и произвести вычисления.

Для решения задачи используется оператор цикла **for**, т. к. известно количество повторений тела цикла (k=10). Введем такие идентификаторы: **x, Y, k** и **S**.

Формула для расчета функции **Y** на каждом шаге вычислений на языке С++ имеет следующий вид:

Y=pow(cos(k*x),3)+sin(0.5*x/pow((k*k+x*x+1),1/3).

Программа для расчета функции **Y** выглядит следующим образом:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int k;
double x=1.32, Y, S;
Y=0;                                     //Начальное значение  переменной Y для
                                         //суммирования
```

```
cout<<"x= "<<x<<endl;                    //Вывод x
    for (k=1; k<=6; k++)                 //Оператор цикла for
    {                                            //Начало составного оператора
                                                 //(тела цикла)
      S=pow(cos(k*x),3)+sin(0.5*x/pow((k*k+x*x+1),1/3));  //Расчет функции на
                                                          //каждом шаге
      Y=Y+S;
      cout<<"k= "<<k<<"    Y = "<<Y<<endl;   //Вывод суммы Y на каждом шаге
    }                                            //Конец тела цикла
getch();
return 0;
}
```

Так как тело цикла содержит более одного оператора, то эти операторы охвачены фигурными скобками.

После запуска программы на экране появится результат:

```
x = 1.32
k = 1    Y = 1.1365
k = 2    Y = 2.27299
k = 3    Y = 3.40949
k = 4    Y = 4.54599
k = 5    Y = 5.68249
k = 6    Y = 6.81898
k = 7    Y = 7.95548
k = 8    Y = 9.09198
k = 9    Y = 10.2285
```

Обратите внимание на то, что результат расчета функции **Y** выводится на каждом шаге вычислений. Это сделано с целью контроля при отладке программы. После устранения возможных ошибок можно выводить только итоговый результат.

Задание 4.5. Самостоятельно разработать алгоритм и программу для вычисления значения функции **Y** с использованием оператора **for** (таб. 4.1). Начертить в отчете блок-схему алгоритма решения задачи и запишите формулу для расчета **Y**. Определите свой номер варианта как номер компьютера.

Таблица 4.1 Исходные данные и формулы для расчета Y (Задание 4.5)

№ варианта	Формула для расчета **Y**	Значения **x** и **a**
1	$Y = \sum_{k=1}^{8} \frac{\sin^3(x+a) - k\cos^2(x+a)}{(x+a)^4}$	x=1,7; a=1,77
2	$Y = \prod_{k=0}^{6} \frac{tg^3(x+a) - \arccos^2(x+a)}{k(x+a)^4}$	x=2,7; a=1,33

3	$Y = \sum_{k=0}^{5} \left(ctg\frac{1-3x}{1+2x} + \cos^2 5x + ke^{3a} \right)$	x=0,7; a=0,46
4	$Y = \prod_{k=0}^{8} \frac{\cos^3(x+a) - k7(x+a)}{tg(x+a)^4}$	x=2,7; a=1,82
5	$Y = \sum_{k=1}^{7} \frac{\cos\left(k\frac{3a+1}{4} \right)}{\sin^3 3x + e^{4a}}$	x=0,45; a=0,82
6	$Y = \prod_{k=1}^{6} \frac{\sin^3(x+a) - \cos^2(x+a)}{k(x+a)^4}$	x=2,1; a=1,47
7	$Y = \sum_{k=0}^{5} \frac{tg\frac{4a^2+1}{4}}{\cos^3 2x + ke^{2a}}$	x=2,1; a=1,34
7	$Y = \prod_{k=0}^{8} \left(\sin\frac{1-x}{1+x} + ktg^4 5x + e^{5a} \right)$	x=0,7; a=1,28
9	$Y = \sum_{k=1}^{5} \frac{\sin^3(x+a) - k\arccos^2(x+a)}{\cos(x+a)^4}$	x=2,2; a=0,66
10	$Y = \prod_{k=0}^{7} \frac{ctg\left(k\frac{x^3+1}{4} \right)}{\cos^2 5x + e^{3a}}$	x=1,45; a=1,12
11	$Y = \sum_{k=1}^{6} \frac{ctg^3(3x+a) - k\sin^2(x+7a)}{(5x+a0)^3}$	x=2,7; a=1,82
12	$Y = \prod_{k=1}^{4} \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 k(3x+e^{3a})}$	x=1,25; a=1,42
13	$Y = \sum_{k=0}^{6} \frac{\sin^3\frac{3x+1}{2}}{tg^2 5x + ke^{3a}}$	x=1,85; a=1,72
14	$Y = \prod_{k=1}^{8} \frac{\sin^3(x+a) - \cos^2(x+a)}{k(x+a)^4}$	x=1,48; a=1,19
14	$Y = \sum_{k=0}^{7} \frac{\operatorname{arcctg}\frac{2kx^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=1,15; a=0,12

16	$Y = \prod_{k=1}^{7} \frac{tg^3(x+a) - 5k(\sin x + a)}{\sin^3(x+a)^4}$	x=2,45; a=2,12
17	$Y = \sum_{k=0}^{6} \left(k \times tg \frac{1-x}{1+x} + k \times \sin^2 5x + e^{5a} \right)$	x=2,25; a=1,88
18	$Y = \prod_{k=1}^{8} \frac{\sin \frac{x+1}{4}}{\sin^2 5x + ke^{3a}}$	x=2,4; a=0,56

Задание 4.6. Исследовать программу вычисления значения функции

$$Y = \sum_{n=0}^{4} n\left(\sum_{k=1}^{5} \sin^3(knx - a) \right)$$

для **x**=1,4 и **a**=2,3 (использовать цикл в цикле с оператором **for**).

Блок-схема алгоритма решения данной задачи приведена на рис. 4.7.

Необходимо ввести код программы, построить решение и произвести вычисления.

Определим исходные данные, которые понадобятся для решения задачи:

a – константа, инициализируем ее, как **a** = 2,3 и определим ее тип как **double** (действительное двойной точности);

x – константа, инициализируем ее, как **x**=1,4, тип **double**;

Y – переменная, тип **double**;

s – промежуточная переменная, обозначим ее как **S**, тип **double**.

Формула для расчета промежуточной суммы (внутри цикла по **k**)

$$s = \sin^3(knx - a)$$

на k-м шаге вычислений на языке C++ имеет следующий вид:

s=pow(sin(k*n*x-a),3) .

Блок-схема алгоритма решения данной задачи приведена на рис. 4.6.

Введите и выполните программный код, реализующий данный алгоритм. Обратите внимание на сдвиг фигурных скобок, ограничивающих сложные операторы, относительно скобок, охватывающих тело главной функции.

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;int _tmain(int argc, _TCHAR* argv[])
{
const double x=1.4, a=2.3;          //Инициализация констант a и x
double Y, n, k, s;                  //Объявление переменных и параметров цикла
cout<<"x= "<<x<<"  a= "<<a<<endl; //Вывод исходных данных
Y=0;                                //Начальное значение Y для суммирования
  for(n=0;n<=4;n++)                 //Внешний оператор for (условие цикла n ≤ 4)
    {                               //Начало составного оператора для внешнего
```

```
                                    //цикла
    s=0;                            //Начальное значение  s для суммирования
      for(k=1;k<=5;k++)             //Вложенный оператор for
      {
      s=s+ pow(sin(k*n*x-a),3);     //Суммирование s на k-ом шаге
      cout<<k<<"   s =  "<<s<<endl;//Вывод на экран переменной y на k-ом шаге
      }
    Y=Y+n*s;                        //Суммирование Y на n-ом шаге
    cout<<n<<"  Y =  "<<Y<<endl;   //Вывод на экран переменной s на k-ом шаге
    }                               //Конец составного оператора с вложенным
                                    //циклом
getch();                            //Функция задержки окна DOS на экране
return 0;
}                                   //Конец главной функции
```

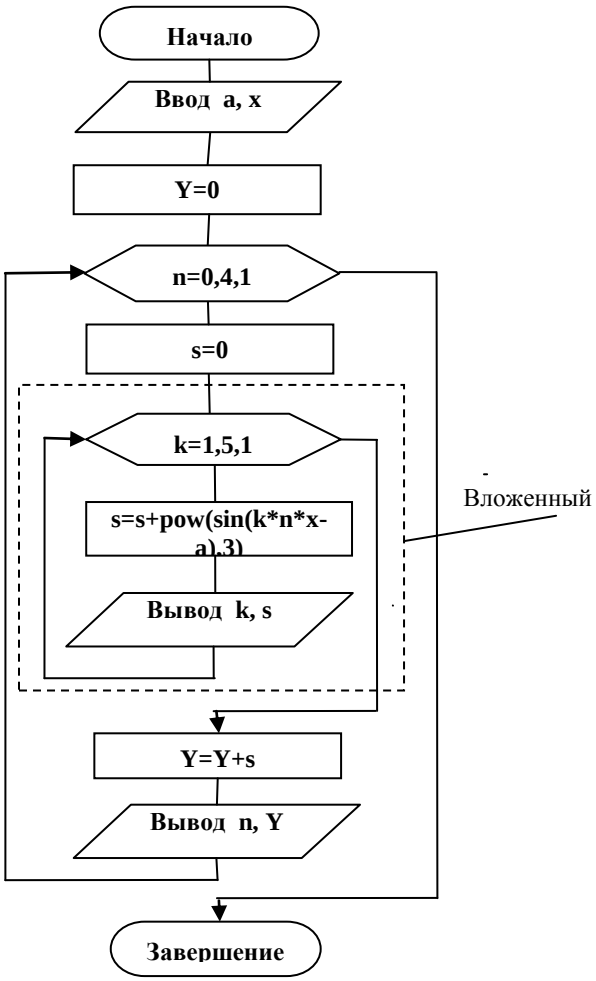


Рис. 4.6. Блок-схема алгоритма решения задания 4.6
с использованием вложенного цикла **for**

Результаты вычислений приведены ниже. Промежуточные результаты вычислений выводятся на экран для дополнительного контроля правильности работы программы (тестирования программы).

```
x= 1.4  a= 2.3
  k = 1    s =  -0.414669
  k = 2    s =  -0.829338
  k = 3    s =  -1.24401
  k = 4    s =  -1.65868
  k = 5    s =  -2.07334

     n = 0  Y =  0
  k = 1    s =  -0.48065
  k = 2    s =  -0.370455
  k = 3    s =  0.476942
  k = 4    s =  0.473016
  k = 5    s =  -0.526753

     n = 1  Y =  -0.526753
  k = 1    s =  0.110195
  k = 2    s =  0.10627
  k = 3    s =  0.100225
  k = 4    s =  0.225993
  k = 5    s =  -0.21643

     n = 2  Y =  -0.959613
  k = 1    s =  0.847396
  k = 2    s =  0.841352
  k = 3    s =  0.388923
  k = 4    s =  1.20605
  k = 5    s =  1.20274

     n = 3  Y =  2.64861
  k = 1    s =  -0.0039253
  k = 2    s =  0.121842
  k = 3    s =  0.938967
  k = 4    s =  1.79397
  k = 5    s =  1.94911

     n = 4  Y =  10.445
```

Задание 4.7. Самостоятельно создайте проект для вычисления функции **Y** по заданной формуле в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 4.2. Ввод исходных данных и вычисления организуйте по аналогии с Заданием 4.6.

Перед **(!)** разработкой программного кода начертите в отчете блок-схему алгоритма решения и запишите формулу для вычислений на языке C++.

Таблица 4.2. Исходные данные и формулы для расчета Y (Задание 4.7)

№ варианта	Формула для расчета **Y**	Значения **a** и **x**
1	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{\left(knx^2-an\right)}$	**a=1,2; x=1,32**
2	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}tg^4\left(knx^3-an\right)$	**a=1,3; x=1,46**
3	$s=\sum_{n=1}^{6}\sum_{k=1}^{5}\sin^4\left(knx^3-a\right)$	a=1,4; x=1,7
4	$s=\sum_{n=1}^{6}\sum_{k=1}^{3}\sqrt[4]{\left(knx^3-ak\right)}$	a=1,5; x=1,4
5	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^3\left(knx^5-a\right)$	a=1,1; x=1,6
6	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[3]{\left(knx^2-a\right)}$	a=1,2; x=1,8
7	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^5\left(knx^3-an\right)$	a=1,3; x=1,4

8	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{\left(knx^4-ak\right)}$	a=1,4; x=1,5
9	$s=\sum_{n=1}^{6}\sum_{k=1}^{4}tg^3\left(knx^5-ax\right)$	a=1,5; x=1,6
10	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-an\right)}$	a=1,1; x=1,4
11	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}\sin^4\left(knx^6-ax\right)$	a=1,2; x=1,8
120	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}tg^3\left(knx^2-a\right)$	a=1,3; x=1,2
13	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-ax\right)}$	a=1,4; x=1,4
14	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^4\left(knx^3-a\right)$	a=1,5; x=1,3
15	$s=\sum_{n=1}^{3}\sum_{k=1}^{7}\sqrt[3]{\left(knx^5-an\right)}$	a=1,1; x=1,5
16	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{\left(knx^3-ak\right)}$	a=1,2; x=1,9
17	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^3\left(knx^6-an\right)$	a=1,3; x=1,4
18	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\cos^4\left(knx^3-ak\right)$	a=1,4; x=1,5

Контрольные вопросы

1. Перечислите, что должно быть в конструкции цикла.

2. С помощью каких действий реализуются циклические вычислительные процессы в Visual C++ 2010?

3. Запишите в общем виде оператор цикла **for** и опишите, как он выполняется.

4. Чем оператор цикла **for** отличается от операторов **while** и **do...while**?

5. В каких случаях используется оператор цикла **for**?

6. Запишите и дайте краткую характеристику операции инкремента?

7. Сколько раз выполнится оператор цикла **int N=3; for (int i=1; i<=N; i++) N=N-1;**?

8. Сколько раз выполнится в программе оператор цикла **for (i = 0; i<1; i++) cout <<i;**?

9. Записан оператор **for (i = 2; i <10; i+=2) cout << i ;**. Что будет выведено на экран дисплея?

Лабораторная работа № 5
ИССЛЕДОВАНИЕ ОПЕРАЦИЙ С ОДНОМЕРНЫМИ МАССИВАМИ ДАННЫХ В VISUAL C++ 2010

Цель работы: получение навыков разработки программ для одномерных массивов данных в Visual C++ 2010.

1. Одномерные массивы данных и их инициализация в Visual C++ 2010

Массив – это конечная именованная последовательность однотипных величин.

Массив в информатике – это некоторое множество мест в памяти компьютера, называемых **элементами массива**, к которым можно обратиться по одному имени переменной. Каждый из **элементов** хранит единицу данных определенного типа (тип данных одинаков для всех элементов массива).

Каждый элемент массива определяется именем массива и его порядковым номером в массиве (индексом). Индекс элемента массива – всегда целое число.

Массивы бывают одномерными и многомерными. В данной работе исследуем работу с **одномерными массивами в Visual C++ 2010**. Одномерные массивы еще называют **векторами**.

Для использования массива в программе его необходимо **объявить**, т. е. зарезервировать под массив определенное количество ячеек памяти.

При объявлении массива указывается тип элементов массива, имя массива и его размер:

Тип Имя массива[размер]; .

Например, оператор

int A[8];

описывает целый одномерный массив по имени **A** из 8-и целых чисел. В памяти будет зарезервировано место для 8-и целочисленных элементов массива (таб. 5.1).

Таблица 5.1. Значения и расположение в памяти элементов одномерного массива A**[8]**

Элемент массива	**A[0]**	**A[1]**	**A[2]**	**A[3]**	**A[4]**	**A[5]**	**A[6]**	**A[7]**
Индекс элемента, i	0	1	2	3	4	5	6	7
Значение элемента	15	22	34	57	11	29	89	47

Обращение к элементам массива осуществляется по имени массива с указанием индекса (номера элемента массива) в квадратных скобках:

A[7]= 47;

x = A[3];
в = A[5];

Если объявлен массив

int B[100]; ,

то элементы массива будут иметь следующие индексы:

B[0], B[1], … B[99].

Тогда оператор

x=B[13];

означает, что переменной **x** будет присвоено значение 14-го элемента массива B.

Массив, как и переменную, можно инициализировать при объявлении. Значения для последовательных элементов массива отделяются один от другого запятыми и помещаются в фигурные скобки. Например,

int C[6]= {2, 4, 7, 11, 12, 13}; .

Если в списке инициализации значений элементов указано меньше, чем размер массива, то имеет место частичная инициализация. При таком объявлении в выражении инициализации после последнего значения для наглядности ставят запятую:

int C[6]= {2, 4, 7,}; .

При этом элементам **C[0]**, **C[1]** и **C[2]** будут присвоены значения 2, 4 и 7, а оставшиеся элементы массива инициализации не получат.

Кроме этого, инициализация массивов возможна в процессе выполнения программы – путем записи данных в отведенные для массивов ячейки памяти.

При работе с массивами, в т. ч. одномерными, целесообразно использовать оператор цикла **for**, т.к. известен размер обрабатываемого массива (число элементов массива), т. е. число повторений цикла.

В языке C++ не проверяется выход индекса за пределы массива. Если массив **m[100]** целочисленный массив:

int m[100]; ,

а в программе указано

x=m[200]; ,

то сообщение об ошибке не будет, а переменной **x** будет присвоено произвольное значение.

При обработке массивов в Visual C++ 2010 все действия в программе выполняются над элементами массива (!), а не над массивом в целом. При этом индекс элемента может быть задан либо его значением, либо выражением:

A[4], F[i+k+1]; .

Над массивами можно выполнять следующие действия:

1. Вводить массивы в память компьютера.
2. Выводить массивы на экран дисплея, на другое устройство или в файл.
3. Присваивать определенные значения элементам массивов.
4. Копировать массивы.
5. Переставлять элементы массивов.
6. Сортировать элементы массивов.

2. Консольный ввод и вывод одномерных массивов в среде Visual C++ 2010

Т. к. все действия необходимо выполнять над элементами массива, то для ввода массива в память компьютера необходимо организовать его поэлементный ввод посредством оператора цикла **for** (т. к. известен размер массива).

Задание 5.1. Исследовать способы консольного ввода и вывода массива **A** из **N** целых чисел. Необходимо ввести с клавиатуры массив **A** в память, определить его размер и вывести эту информацию на дисплей.

Блок-схема алгоритма решения такой задачи приведена на рис. 5.1:

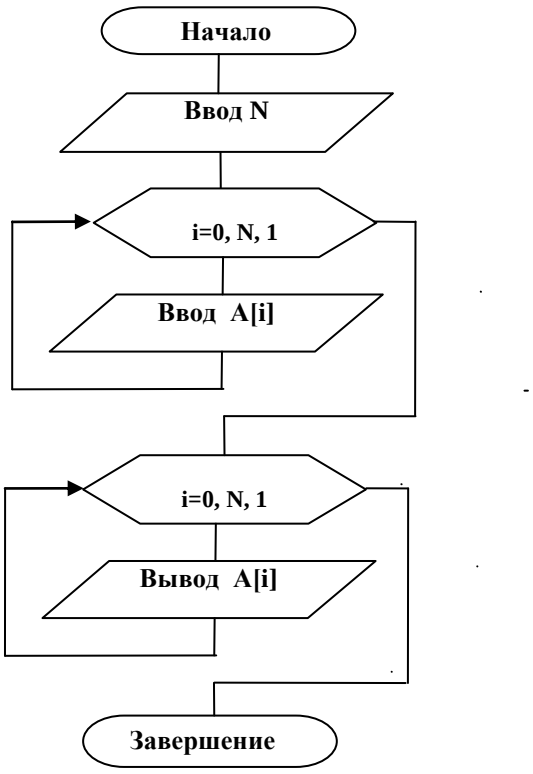


Рис. 5.1. Блок-схема алгоритма ввода и вывода элементов вектора с использованием цикла **for**

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int const N=8;                          //Инициализация размерности массива
int i;                                  //Объявление параметра цикла i
int A[N];                               //Объявление одномерного массива  A
cout<<endl<<"Vvedite massiv:"<<endl;
    for(i=0; i<N; i++)                  //Цикл по i для ввода массива  A
```

```
    cin>>A[i];                          //Ввод i-го элемента массива  A
cout<<endl;
       for (i=0; i<N; i++)              //Цикл по i для вывода массива  A
       cout<<"   A["<<i<<"]= "<<A[i];   //Вывод i-го элемента массива  A
cout<<endl<<"size= "<<i;                //Вывод размерности массива  A  на
экран
getch();
return 0;
}
```

После запуска программы и ввода элементов массива **A** вид экрана будет следующий:

Vvedite massiv:
1 2 3 4 5 6 7 8

A[0]= 1 A[1]= 2 A[2]= 3 A[3]= 4 A[4]= 5 A[5]= 6 A[6]= 7 A[7]= 8
size= 8

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр6**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 5.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr5-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр5**. Над вставленным рисунком проставьте номер задания – **5-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

3. Исследование операций с одномерными массивами в среде Visual C++

Элементам массива могут быть **присвоены** значения выражений. При этом элементы массива и значения выражений должны иметь один и тот же тип.

Например, объявлен массив

double A[3];

тогда возможна запись

A[0]= 3.5;
A[1]= 0;
A[2]= a*x + b;

Копирование – это присваивание значений элементов одного массива элементам другого массива. При копировании оба массива должны иметь одинаковый размер и тип элементов. Копирование массива **A** в **B** будет иметь вид:

for (i=0; i<N; i++) B[i]= A[i];

Задание 5.2. Задать массив **X** действительных чисел из 8 элементов непосредственным присваиванием в программном коде. Вычислить массив **Y** по заданной формуле:

$$Y_i = 2X_i + \sqrt[3]{X_i^5}\ .$$

Массив **Y** вывести на экран.

Решение. Запишем формулу для расчета элементов массива **Y** на языке С++:

Y [i]=2*X[i]+pow(X[i], 5/3).

Блок-схема алгоритма решения этой задачи аналогична блок-схеме, приведенной на рис. 5.1. После блока **Вывод A[i]** необходимо включить операционный блок для расчета

Y [i]=2*X[i]+pow(X[i], 5/3).

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i;
double X[8]={1.2,3.4,12,5.12,23.1,3,0,35.2}; //Инициализация элементов массива X
double Y [8];
   for (i=0; i<8; i++)                    //Цикл для вычисления и вывода элемента  Y[i]
   {                                      //Начало составного оператора
      Y [i]=2*X[i]+pow(X[i], 5/3);        //Вычисление элементов массива Y[8]
      cout<<"  X["<<i<<"]="<<X[i];        //Вывод элементов массива X[8]
      cout<<"  Y["<<i<<"]="<<Y[i];        //Вывод элементов массива Y[8]
      cout<<endl;                         //Переход на новую строку
   }                                      //Конец составного оператора
getch();
return 0;
}
```

В результате выполнения программы получим:

X[0]=1.2 Y[0]=3.6

X[1]=3.4 Y[1]=10.2

X[2]=12 Y[2]=36

X[3]=5.12 Y[3]=15.36

X[4]=23.1 Y[4]=69.3

X[5]=3 Y[5]=9

X[6]=0 Y[6]=0

X[7]=35.2 Y[7]=105.6

Задание 5.3. Самостоятельно разработать программу для выполнения следующих действий:

1. Задать массив **X** действительных чисел из 8 элементов непосредственным присваиванием в программном коде:

X[8]={3.9 2.5 3.6 6.2 5.0 3.3 2.7 4.6}.

2. Вычислить массив **Y** по заданной формуле, согласно своему варианту (таблица 5.1).

!!! Определить свой вариант как номер компьютера.

3. Массивы **X** и **Y** вывести на экран.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++.

Таблица 5.1 Исходные данные и формулы для расчета Y (Задание 5.3)

№ варианта	Формула для расчета **Y**	Значение **a**
1	$Y_i = \operatorname{tg}\frac{1-x_i}{1+x_i} + \sin^2 5x_i + e^{5a}$	a=1,88
2	$Y_i = \frac{\sin\frac{x_i+1}{4}}{\sin^2 5x_i + e^{3a}}$	a=0,56
3	$Y_i = \frac{\sin^3(x_i+a)-\cos^2(x_i+a)}{(x_i+a)^4}$	a=1,77
4	$Y_i = \frac{\operatorname{tg}^3(x_i+a)-\arccos^2(x_i+a)}{(x_i+a)^4}$	a=1,33
5	$Y_i = \operatorname{ctg}\frac{1-3x_i}{1+2x_i} + \cos^2 5x_i + e^{3a}$	a=0,46
6	$Y_i = \frac{\cos^3(x_i+a)-7(x_i+a)}{\operatorname{tg}(x_i+a)^4}$	a=1,82

7	$Y_i = \frac{\cos\left(\frac{3a + x_i}{4}\right)}{\sin^3 3x_i + e^{4a}}$	a=0,82
8	$Y_i = \frac{\sin^3(x_i + a) - \cos^2(x_i + a)}{(x_i + a)^4}$	a=1,47
9	$Y_i = \frac{\operatorname{tg}\frac{4a^2 + 1}{4}}{\cos^3 2x_i + e^{2a}}$	a=1,34
10	$Y_i = \sin\frac{1 - x_i}{1 + x_i} + \operatorname{tg}^4 5x_i + e^{5a}$	a=1,28
11	$Y_i = \frac{\sin^3(x_i + a) - \arccos^2(x_i + a)}{\cos(x_i + a)^4}$	a=0,66
12	$Y_i = \frac{\operatorname{ctg}\left(\frac{x_i^3 + 1}{4}\right)}{\cos^2 5x_i + e^{3a}}$	a=1,12
13	$Y_i = \frac{\operatorname{ctg}^3(3x_i + a) - \sin^2(x_i + 7a)}{(5x_i + a)^3}$	a=1,82
14	$Y_i = \frac{\arcsin^3\frac{4x_i + 1}{4}}{\operatorname{ctg}^2(3x_i + e^{3a})}$	a=1,42
15	$Y_i = \frac{\sin^3\frac{3x_i + 1}{2}}{\operatorname{tg}^2 5x_i + e^{3a}}$	a=1,72
16	$Y_i = \frac{\sin^3(x_i + a) - \cos^2(x_i + a)}{(x_i + a)^4}$	a=1,19
17	$Y_i = \frac{\operatorname{arcctg}\frac{x_i^3 + 1}{4}}{\cos^2 5x_i + e^{3a}}$	a=0,12
18	$Y_i = \frac{\operatorname{tg}^3(x_i + a) - 5(\sin x_i + a)}{\sin^3(x_i + a)^4}$	a=2,12

Рассмотрим применение оператора цикла **for** для расчета, например, суммы элементов массива.

Задание 5.4. Задан массив **A** из **N** произвольных чисел. Массив необходимо инициализировать непосредственно в программе. Необходимо вычислить сумму элементов массива **A**. На экран вывести исходный массив **A,** сумму элементов массива **A** и число его элементов.

В качестве исходного можно использовать массив **X[8]** из задания 5.2.

Определим типы и структуры данных, которые будут использоваться в программе.

N – размер массива. Зададим его константой, равной, например, 8;

S – переменная для накопления суммы элементов массива;

i – параметр цикла;

Блок-схема алгоритма решения этой задачи приведена на рис. 5.2:

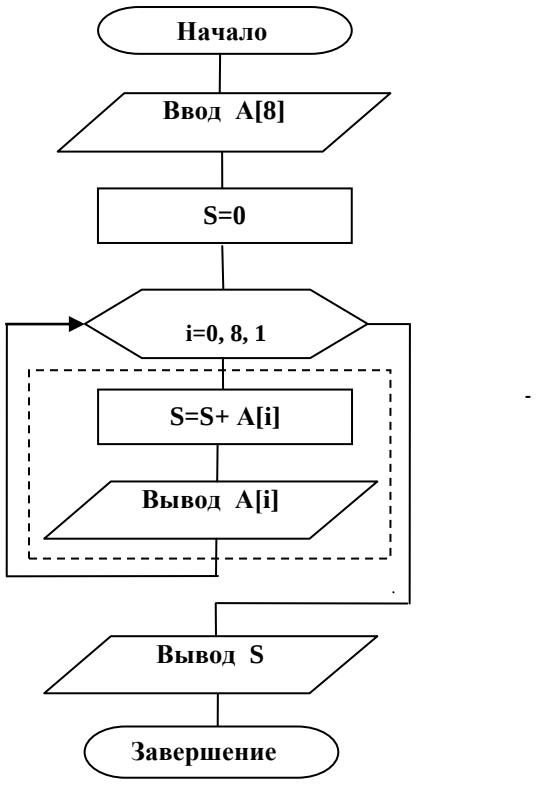


Рис. 5.2. Блок-схема алгоритма вычисления суммы элементов одномерного массива

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i;
```

```
double A[8]={1.2,3.4,12,5.12,23.1,3,0,35.2}; //Инициализация массива A[8]
double S=0;                                   //Начальное значение S для суммы
   for (i=0; i<8; i++)                        //Цикл для вычисления суммы
                                              //элементов массива A[8] и вывода его
                                              //элементов  на экран
   {                                          //Начало составного оператора
      S=S+A[i];                               //Вычисление суммы элементов
                                              //массива A[8]
      cout<<"  A["<<i<<"]="<<A[i];            //Вывод элементов массива A[8]
   }                                          //Конец составного оператора
cout<<endl<<"  S=  "<<S<<"  i= "<<i;         //Вывод суммы и количества
                                              //элементов массива A на экран
getch();
return 0;
}
```

В результате выполнения программы получим:

A[0]=1.2 A[1]=3.4 A[2]=12 A[3]=5.12 A[4]=23.1 A[5]=3 A[6]=0 A[7]=35.2

S=83.02 i=8

Задание 5.5. Самостоятельно разработать программу для выполнения следующих действий:

1. Задать массив **X** действительных чисел из 8 элементов непосредственным присваиванием в коде:

X[8]={3.9 2.5 3.6 6.2 5.0 3.3 2.7 4.6}.

2. Вычислить массив **Y** по заданной формуле, согласно своему варианту (таблица 5.1).

3. Вычислить сумму элементов массива **Y.**

Массив **X** необходимо инициализировать непосредственно в программном коде. На экран вывести исходный массив **X,** массив **Y,** сумму элементов массива **Y** и число его элементов.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++.

Задание 5.6. Составить программу ввода массива **X** произвольной длины от 1 до 20 элементов (0<**k**<20). Вычислить массив **Y** по следующей формуле:

$$y_i = \begin{cases} x_i^2 + \sin^3 3x_i, & \text{если } x > 4; \\ \sqrt{\ln x_i} - tgx_i, & \text{если } x \le 4. \end{cases}.$$

Текущую длину массива **k**=10 и значения элементов массива **X**(10)=1, 2, 3, 4, 5, 6, 7, 8,9,10 ввести с клавиатуры.

Блок-схема алгоритма решения этой задачи приведена на рис. 5.3.

Запишем формулы для вычисления массива Y на языке C++:

Y[i]=X[i]*X[i]+pow(sin(3*X[i]),3) для **X[i]>4** и

Y[i]=sqrt(log(X[i]))-tan(X[i]) для **X[i]<=4**.

Тогда программа имеет вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int imax=20;                          //Максимальный размер массива
                                            //(задается с запасом)
int i,  k;                                  //Счетчик и текущая длина массива X
float    X[imax], Y[imax];                  //Объявление массивов X и Y
cout<<"Vvedite razmer massiva ot 1 do 20: ";
cin>>k;                                     //Ввод длины массива X
   for (i=0; i<k; i++)                      //Цикл для ввода элементов массива X
      {                                     //Начало составного оператора
      cout <<" Vvedite  " <<i<<" element:   ";
      cin>>X[i];                            //Ввод элементов массива X
      }                                     //Конец составного оператора
   for (i=0; i<k; i++)                      //Цикл для вычисления массива Y
      {
      if(X[i]>4)                            //Начало условного оператора
      Y[i]=X[i]*X[i]+pow(sin(3*X[i]),3);    //Вычисление массива Y для X[i]>4
      else
      Y[i]=sqrt(log(X[i]))-tan(X[i]);       //Вычисление массива Y для X[i]<=0
      cout<<"  X["<<i<<"]="<<X[i];          //Вывод элементов массива X на экран
      cout<<"  Y["<<i<<"]="<<Y[i]<<endl;//Вывод элементов массива Y
      }                                     //Конец составного оператора
getch();
return 0;
}
```

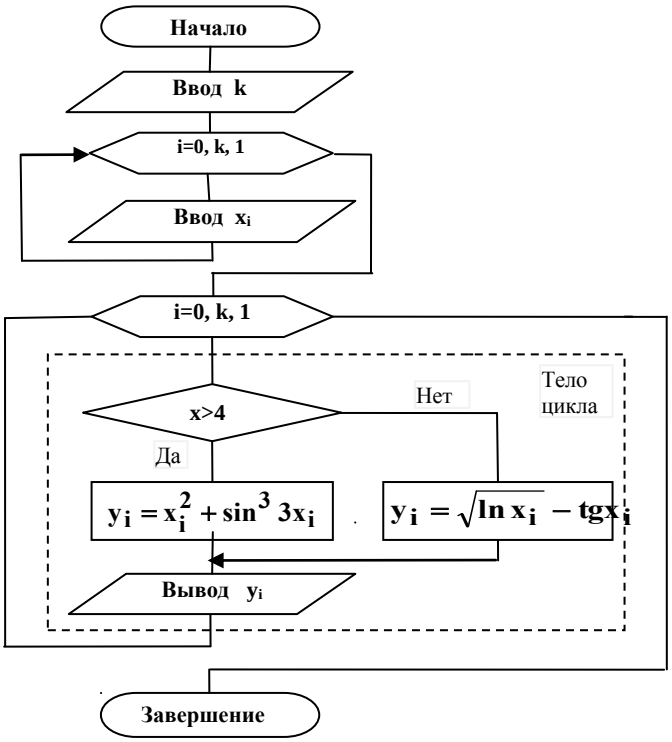


Рис. 5.3. Блок-схема алгоритма вычисления элементов одномерного массива в задании 5.6

В результате выполнения программы получим:

Vvedite razmer massiva ot 1 do 20: 8

Vvedite 0 element: 1
Vvedite 1 element: 2
Vvedite 2 element: 3
Vvedite 3 element: 4
Vvedite 4 element: 5
Vvedite 5 element: 6
Vvedite 6 element: 7
Vvedite 7 element: 8

X[0]=1 Y[0]=-1.55741
X[1]=2 Y[1]=3.01759
X[2]=3 Y[2]=1.19069
X[3]=4 Y[3]=0.0195887
X[4]=5 Y[4]=25.275
X[5]=6 Y[5]=35.5765
X[6]=7 Y[6]=49.5857
X[7]=8 Y[7]=63.2574

Задание 5.7. Самостоятельно разработать программу для ввода массива **X** переменной длины от 1 до 20 элементов (**0<k<20**). Текущую длину массива **k**=10 и элементы массива **X** (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) ввести с клавиатуры.

Массив **Y** рассчитать по формуле в соответствии со своим вариантом (таб. 5.2). Определите свой номер варианта как номер компьютера.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулы для расчета на языке С++.

Задание выполните для двух произвольных значений **x** из двух интервалов.

Таблица 5.2. Исходные данные и формулы для расчета Y (Задание 5.7)

№ вар.	Y=f(x)	№ вар.	Y=f(x)
1	$y_i = \begin{cases} 5x_i^2 + \ln x_i, & \text{если } x < 0; \\ \lg 24^4 + 3x_i^6, & \text{если } x \ge 0; \end{cases}$	10	$y_i = \begin{cases} 23.4^2 + 2\sqrt[3]{12.1x_i}, & \text{если } x < 1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$
2	$y_i = \begin{cases} 23.4^2 + 2\sqrt[3]{12.1x_i}, & \text{если } x < 1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$	11	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 2.5^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$
3	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 2.5^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$	12	$y_i = \begin{cases} \log_3 5x_i^2 + \sqrt{x_i^7}, & \text{если } x < 0; \\ e^{2x_i} + 3x_i^6, & \text{если } x \ge 0; \end{cases}$
4	$y_i = \begin{cases} \log_3 5x_i^2 + \sqrt{x_i^7}, & \text{если } x < 0; \\ e^{2x_i} + 3x_i^6, & \text{если } x \ge 0; \end{cases}$	13	$y_i = \begin{cases} 7.56 + \sin^2 x_i, & \text{если } x < 1 \\ 1.09^3 - x_i, & \text{если } x \ge 1 \end{cases}$
5	$y_i = \begin{cases} 7.56 + \sin^2 x_i, & \text{если } x < 1; \\ 1.09^3 - x_i, & \text{если } x \ge 1; \end{cases}$	14	$y_i = \begin{cases} 1.2^2 + 2\sqrt[3]{3}x_i, & \text{если } x < 1 \\ tg^3 x_i, & \text{если } x \ge 1 \end{cases}$
6	$y_i = \begin{cases} 1.2^2 + 2\sqrt[3]{3}x_i, & \text{если } x < 1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$	15	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 3.2^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$
7	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 3.2^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$	16	$y_i = \begin{cases} 2x_i^3 + 4\sin x_i, & \text{если } x < 0 \\ tg^2 x_i^3, & \text{если } x \ge 0 \end{cases}$

8	$y_i = \begin{cases} 2x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ tg^2 x_i^3, & \text{если } x \ge 0; \end{cases}$	17	$y_i = \begin{cases} 2\cos x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ x_i^6, & \text{если } x \ge 0; \end{cases}$
9	$y_i = \begin{cases} 2\cos x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ x_i^6, & \text{если } x \ge 0; \end{cases}$	18	$y_i = \begin{cases} 5x_i^2 + \ln x_i, & \text{если } x < 0; \\ 6.8^4 - \sqrt{x_i}, & \text{если } x \ge 0; \end{cases}$

4. Поиск элемента в одномерных массивах в среде Visual C++ 2010

Поиск элемента в массиве заключается в выделении из массива отдельных его элементов. Поиск может проводиться по образцу или по правилу.

Поиск по образцу заключается в следующем. Задается значение некоторой переменной (образец) и все элементы массива (или часть элементов) сравниваются со значением этой переменной (образцом).

Поиск по правилу проводится на основе проверки некоторых условий, которым должны отвечать либо элемент массива, либо группа элементов.

Задание 5.8. Исследуем программу для поиска элемента одномерного массива по образцу.

Задан массив **A** из 8 произвольных чисел. Необходимо определить количество элементов, которые больше заданного числа **q** и их порядковые номера. На экран вывести исходный массив, элементы, большие **q** и их порядковые номера.

Перед решением задачи определим типы и структуры данных, которые будут использоваться в программе:

q – число, по которому осуществляется поиск элементов исходного массива **A**;

i – номер элемента исходного массива **A** (параметр цикла);

A[8] – массив из **N** произвольных чисел;

X[8] – массив для хранения номеров элементов исходного массива **A** , больших **q.**

j – номер элемента массива **X** для хранения номеров (параметр цикла);

Блок-схема алгоритма решения задачи поиска элементов одномерного массива приведена на рис. 5.4.

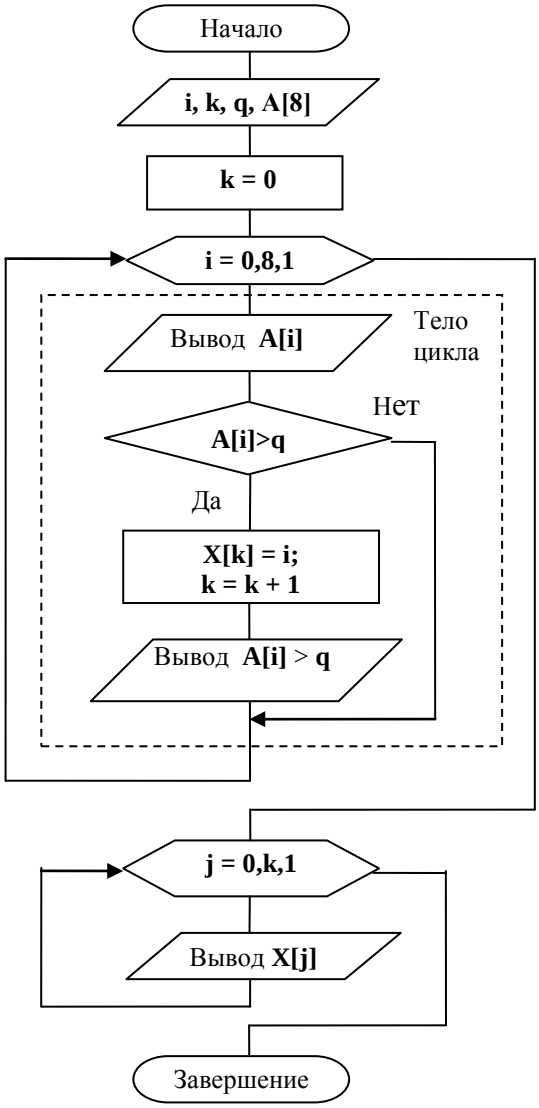


Рис. 5.4. Блок-схема алгоритма поиска элементов
одномерного массива по образцу

Тогда программа имеет вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j, k;
double q=3.7;                    //Объявление переменной  q
double A[8]={1,2,3,4,5,6,7,8};   //Инициализация исходного массива A[8]
int X[8];                        //Объявление массива X[k] номеров элементов
                                 //массива  A[8]
k=0;
```

```
    for (i=0; i<8; i++)                  // Цикл для перебора элементов массива A[8]
    {                                    //Начало составного оператора в цикле
    cout<<"  A["<<i<<"] = "<<A[i];      //Вывод всех элементов массива  A[8]
         if(A[i]>q)                      //Условный оператор для сравнения
                                         //элементов массива A[8] с q
         {                               //Начало составного оператора в условном
         X[k]=i;                         //Записываются номера элементов, больших
                                         //q,  в массив X[k]
         k=k+1;
       cout<<" >q     ";                //Вывод обозначения  >q , если элемент
                                         //массива A[8] больше, чем q
         }                               //Конец составного оператора в условном
    }                                    //Конец составного оператора в цикле
  cout<<endl<<"Nomera elementov  >  q: ";
    for (j=0;j<k; j++)                   //Цикл для вывода номеров элем-в массива
                                         //A[8], больших q
  cout<<X[j]<<" ";                       //Вывод номеров элем-в массива A[8],
                                         //больших q                    getch();
  return 0;
  }
```

В результате выполнения программы получим:

A[0]=1 A[1]=2 A[2]=3 A[3]=4 >q A[4]=5 >q A[5]=6 >q A[6]=7 >q A[7]=8 >q

Nomera elementov > q: 3 4 5 6 7

Задание 5.9. Исследуем программу для поиска элемента одномерного массива по правилу.

Задан массив **A** из 8 произвольных чисел. Найти максимальный элемент массива **A** и его номер. На экран вывести исходный массив **A**, максимальный элемент массива **A** и его номер.

Решение. Определим типы и структуры данных, которые будут использоваться в программе:

i – номер элемента исходного массива **A** (параметр цикла);

A[8] – массив из 8 произвольных чисел;

B – вспомогательная переменная для хранения максимального элемента массива для i-го шага;

C – вспомогательная переменная для хранения номера (индекса) максимального элемента массива.

Блок-схема алгоритма поиска максимального элемента одномерного массива приведена на рис. 5.5.

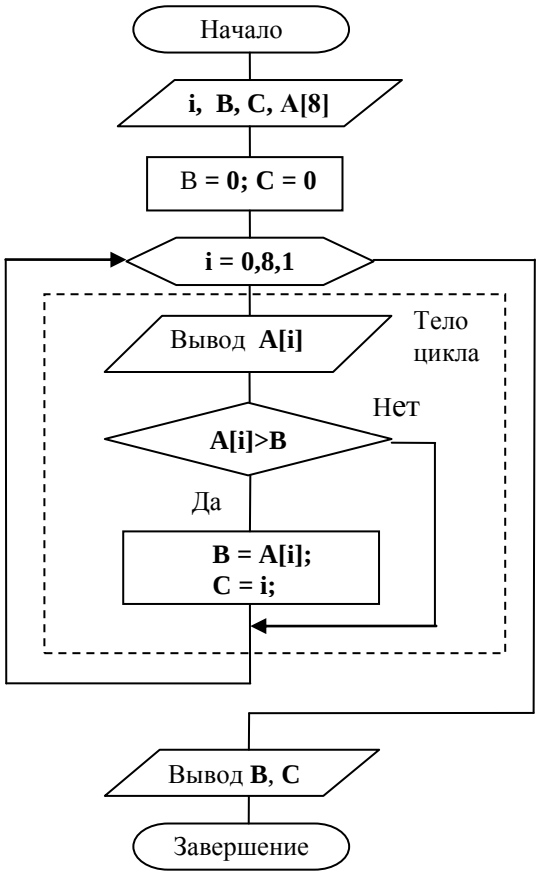


Рис. 5.5. Блок-схема алгоритма поиска
максимального элемента одномерного массива

Программа поиска максимального элемента одномерного массива имеет вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, C;                          //Объявление вспомогательной переменной C
double B;                          //Объявление вспомогательной переменной B
double A[8]={1, 2, 3, 4, 5, 6, 7, 1};    //Инициализация элементов исходного
массива A[8]
B=A[0];
C=0;
   for (i=0; i<8; i++)             //Цикл для перебора элементов массива A[8]
   {                               //Начало составного оператора в цикле
   cout<<"  A["<<i<<"] = "<<A[i];  //Вывод всех элементов массива  A[8]
      if(A[i]>B)                   //Условный оператор для сравнения
элементов массива A[8]
      {                            //Начало составного оператора в условном
```

```
        B=A[i];                 //Присваивание переменной B значения
                                //максимального элемента
        C=i;                    //Присваивание переменной C номера
                                //максимального элемента
        }                       //Конец составного оператора в условном
операторе
    }                           //Конец составного оператора в цикле
cout<<endl<<"Max element:  "<<B;
cout<<endl<<"N max elementa:  "<<C;
getch();
return 0;
}
```

В результате выполнения программы получим:

A[0] = 1 A[1] = 2 A[2] = 3 A[3] = 4 A[4] = 5 A[5] = 6 A[6] = 7 A[7] = 1
Max element: 7
N max elementa: 6

Задание 5.10. Самостоятельно разработать программу для поиска максимального элемента одномерного массива **Y**. Массив **Y** рассчитать из исходного массива **X** по формуле в соответствии со своим вариантом (таб. 5.3).

Массив **X** (1, 2, 3, 4, 5, 6, 7, 8, 9, 10) ввести в программном коде. Определите свой номер варианта как номер компьютера.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулы для расчета на языке С++.

Задание выполните для двух произвольных значений **x** из двух интервалов.

Таблица 5.3. Исходные данные для расчета массива Y (Задание 5.10)

№ вар.	Y=f(x)	№ вар.	Y=f(x)
1	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 3.2^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$	10	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 3.2^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$
2	$y_i = \begin{cases} 2x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ tg^2 x_i^3, & \text{если } x \ge 0; \end{cases}$	11	$y_i = \begin{cases} 2x_i^3 + 4\sin x_i, & \text{если } x < 0 \\ tg^2 x_i^3, & \text{если } x \ge 0 \end{cases}$

3	$y_i = \begin{cases} 5x_i^2 + \ln x_i, & \text{если } x < 0; \\ \lg 24^4 + 3x_i^6, & \text{если } x \ge 0; \end{cases}$	12	$y_i = \begin{cases} 23.4^2 + 2\sqrt[3]{12.1x_i}, & \text{если } x < 1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$
4	$y_i = \begin{cases} 23.4^2 + 2\sqrt[3]{12.1x_i}, & \text{если } x < 1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$	13	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 2.5^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$
5	$y_i = \begin{cases} \cos^2 x_i + 2\sin x_i, & \text{если } x < 1; \\ 2.5^3 - 3x_i, & \text{если } x \ge 1; \end{cases}$	14	$y_i = \begin{cases} \log_3 5x_i^2 + \sqrt{x_i^7}, & \text{если } x < 0; \\ e^{2x_i} + 3x_i^6, & \text{если } x \ge 0; \end{cases}$
6	$y_i = \begin{cases} \log_3 5x_i^2 + \sqrt{x_i^7}, & \text{если } x < 0; \\ e^{2x_i} + 3x_i^6, & \text{если } x \ge 0; \end{cases}$	15	$y_i = \begin{cases} 7.56 + \sin^2 x_i, & \text{если } x < 1 \\ 1.09^3 - x_i, & \text{если } x \ge 1 \end{cases}$
7	$y_i = \begin{cases} 7.56 + \sin^2 x_i, & \text{если } x < 1; \\ 1.09^3 - x_i, & \text{если } x \ge 1; \end{cases}$	16	$y_i = \begin{cases} 1.2^2 + 2\sqrt[3]{3}x_i, & \text{если } x < 1 \\ tg^3 x_i, & \text{если } x \ge 1 \end{cases}$
8	$y_i = \begin{cases} 2x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ tg^2 x_i^3, & \text{если } x \ge 0; \end{cases}$	17	$y_i = \begin{cases} 2\cos x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ x_i^6, & \text{если } x \ge 0; \end{cases}$
9	$y_i = \begin{cases} 2\cos x_i^3 + 4\sin x_i, & \text{если } x < 0; \\ x_i^6, & \text{если } x \ge 0; \end{cases}$	18	$y_i = \begin{cases} 5x_i^2 + \ln x_i, & \text{если } x < 0; \\ 6.8^4 - \sqrt{x_i}, & \text{если } x \ge 0; \end{cases}$

Контрольные вопросы

1. Приведите понятие массива.
2. Приведите понятие одномерного массива.
3. Перечислите основные свойства массивов.
4. Как объявляются одномерные массивы?
5. Как обращаются к элементам одномерных массивов?
6. Как и какие можно выполнять действия над одномерными массивами?

7. С помощью какого оператора рациональнее выполнять действия над одномерными массивами?

8. Каким оператором можно ввести с клавиатуры **n** элементов массива **X**?

1) **for (i=0; i<=n; i++) cin>>X[i]**

2) **for (i=0; i< n; i++) cin>>X[i]**

3) **for (i=1; i<=n; i++) cin>>X[i]**

9. Как в программе на С++ будет выведен на экран массив **A[k]**оператором **for(i=0; i<k; i++) cout<<A[i]<<endl;**?

1) В виде строки 2) В виде столбца 3) В виде матрицы

10. Какой оператор копирует массив **A[N]** в массив **B[N]**?

1) **for (i=0; i<N; i++) B[i]=A[i];**

2) **for (i=0; i<K; i++) B[i]=A[i];**

3) **for (i=0; i<N; i++) A[i]=B[i];**

11. Задан массив **X = {X1, X2 ..., XN}**. Каким оператором можно вывести этот массив на экран?

1) **cout<<X;**

2) **for (i=0; i<n; i++) cout<<X;**

3) **for (i=0; i<n; i++) cout<<X[i];**

Лабораторная работа №6
ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ СРЕДЫ VISUAL C++ 2010 ДЛЯ ОПЕРАЦИЙ С МНОГОМЕРНЫМИ МАССИВАМИ ДАННЫХ

Цель работы: получение навыков разработки программ для многомерных массивов данных в Visual C++ 2010.

1. Двухмерные массивы данных и их инициализация в Visual C++ 2010

Под размерностью массива понимают число индексов, которое необходимо указать для получения доступа к отдельному элементу массива. Массивы, рассмотренные в лабораторной работе № 5, например, были одномерными и требовали только одного индекса. Массивы с более чем одной размерностью, называются многомерными.

Самым простым многомерным массивом является двухмерный массив (матрица).

При объявлении массива указывается тип элементов массива, имя массива и его размер:

Тип ИмяМассива [Размер 1] [Размер 2];

Например, оператор

int A[3][8];

описывает целый двухмерный массив по имени **A** из 24-х целых чисел. В памяти будет зарезервировано место для 24-х целочисленных элементов массива (рис. 9.1).

В памяти компьютера двухмерный массив располагается непрерывно по строкам, то есть

A[0][0], A[0][1], A[0][2], A[0][3] … A[0][8], A[1][0], A[1][1], A[1][2], A[1][8] … A[0][8].

На рис. 9.1 приведена схема размещения элементов массива **A** из 24-х целых чисел **A** размерностью 3×8. В памяти будет зарезервировано место для 24-х целочисленных элементов массива, которые располагаются в **непрерывном (!!!)** блоке памяти.

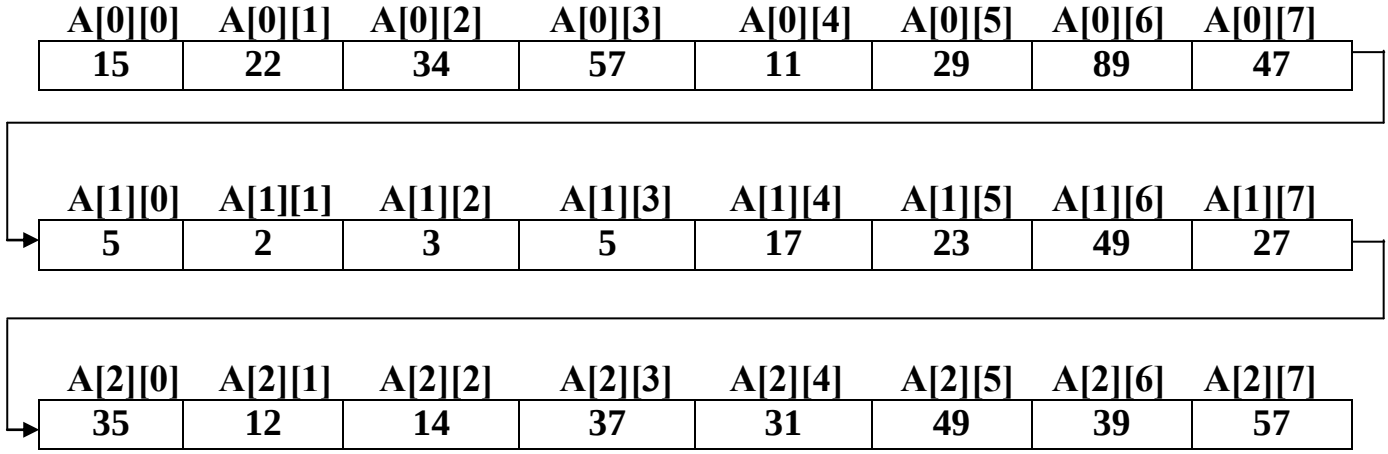


Рис. 6.1. Значения и расположение в памяти элементов массива **A[3][8]**

Массивы хранятся в памяти компьютера так, что самый правый индекс изменяется быстрее всего.

Возможна инициализация массива непосредственно в программном коде. В этом случае в фигурных скобках приводятся значения элементов массива (рис. 6.1) в следующем виде:

int A[2][3]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

или

int A[2][3]={{1, 2, 3, 4} {5, 6, 7, 8} {9, 10, 11, 12}};

Количество инициализаторов не обязано совпадать с количеством элементов массива. Если инициализаторов меньше, то оставшиеся элементы не определены.

Двухмерный массив, например, **int A[3][4]** можно представить в виде следующей матрицы:

Первый индекс – это номер строки в массиве, второй индекс – номер столбца.

В памяти компьютера элементы такой матрицы разместятся в таком порядке:

A_{00}, A_{01}, A_{02}, A_{03}, A_{10}, A_{11}, A_{12}, A_{13}, A_{20}, A_{21}, A_{22}, A_{23}.

Кроме этого, инициализация массивов возможна в процессе выполнения программы – путем записи данных в отведенные для массивов ячейки памяти.

При работе с массивами, в т. ч. одномерными, целесообразно использовать оператор цикла **for**, т.к. известен размер обрабатываемого массива (число элементов массива), т. е. число повторений цикла.

В языке C++ не проверяется выход индекса за пределы массива. Если массив **m[100]** целочисленный массив

int m[100]; ,

а в программе указано

x=m[200]; ,

то сообщение об ошибке не будет, а переменной **x** будет присвоено произвольное значение.

При обработке массивов в Visual C++ 2010 все действия в программе выполняются над элементами массива (!), а не над массивом в целом. При этом индекс элемента может быть задан либо его значением, либо выражением:

A[4], F[i+k+1]; .

Над массивами можно выполнять следующие действия:

1. Вводить массивы в память компьютера.
2. Выводить массивы на экран дисплея, в файл и др.
3. Присваивать определенные значения элементам массивов.
4. Копировать массивы.
5. Переставлять элементы массивов.
6. Сортировать элементы массивов.

2. Консольный ввод и вывод двухмерных

Т. к. все действия необходимо выполнять над элементами двухмерных массивов, то для ввода двухмерного массива в память компьютера необходимо организовать его поэлементный ввод посредством оператора цикла **for** (т. к. известен размер массива). Понадобится внешний цикл, например, по строкам, и внутренний, соответственно, по столбцам.

Для консольного вывода двухмерного массива также надо с помощью внешнего и вложенного операторов цикла **for** организовать поэлементный вывод исследуемого массива.

Задание 6.1. Исследовать способы консольного ввода и вывода двухмерных массивов. Необходимо ввести с клавиатуры матрицу **A** размерностью **M**×**N** в память компьютера и вывести эту информацию на дисплей.

Решение. Для придания программе большей универсальности зададим размерность исходной матрицы с запасом, а реальная размерность будет вводиться в каждом конкретном случае. Число строк обозначим **m**, а столбцов - **n.**

Поэлементный ввод матрицы **A** организован при помощи двух операторов цикла **for** – внешнего и внутреннего (вложенного). Внешний цикл организует перебор элементов матрицы **A** по строкам, а вложенный – по столбцам.

Поэлементный вывод исходной матрицы на экран будет производиться аналогичным образом.

Блок-схема алгоритма решения данной задачи приведена на рис. 6.1.

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int M=10,N=10;                //Число строк и столбцов матрицы A с запасом
int i, j, m, n, A[M][N];            //Объявление переменных и матрицы A
cout<<"Vvedite chislo strok i stolbcov:"<<endl;
cin>>m>>n;                          //Ввод числа строк и столбцов матрицы A
cout<<" Vvedite postrochno elementu:"<<endl;
for(i=0; i<m; i++)                  //Внешний цикл ввода элементов матрицы A
for(j=0; j<n; j++)                  //Вложенный цикл ввода элементов матрицы A
cin>>A[i][j];                       //Ввод элемента матрицы A
cout<<endl;
cout<<"Matrica A"<<endl;
   for(i=0; i<m; i++)               //Внешний цикл вывода элементов матрицы A
   {                                //Начало составного оператора для внешнего
                                    //цикла for
      for(j=0; j<n; j++)            //Вложенный цикл для вывода элементов
                                    //матрицы A
            cout<<A[i][j]<<"   ";   //Вывод элемента матрицы A
```

```
   cout<<endl;
        }                                  //Конец составного оператора для внешнего
                                           //цикла for
getch();
return 0;
}
```

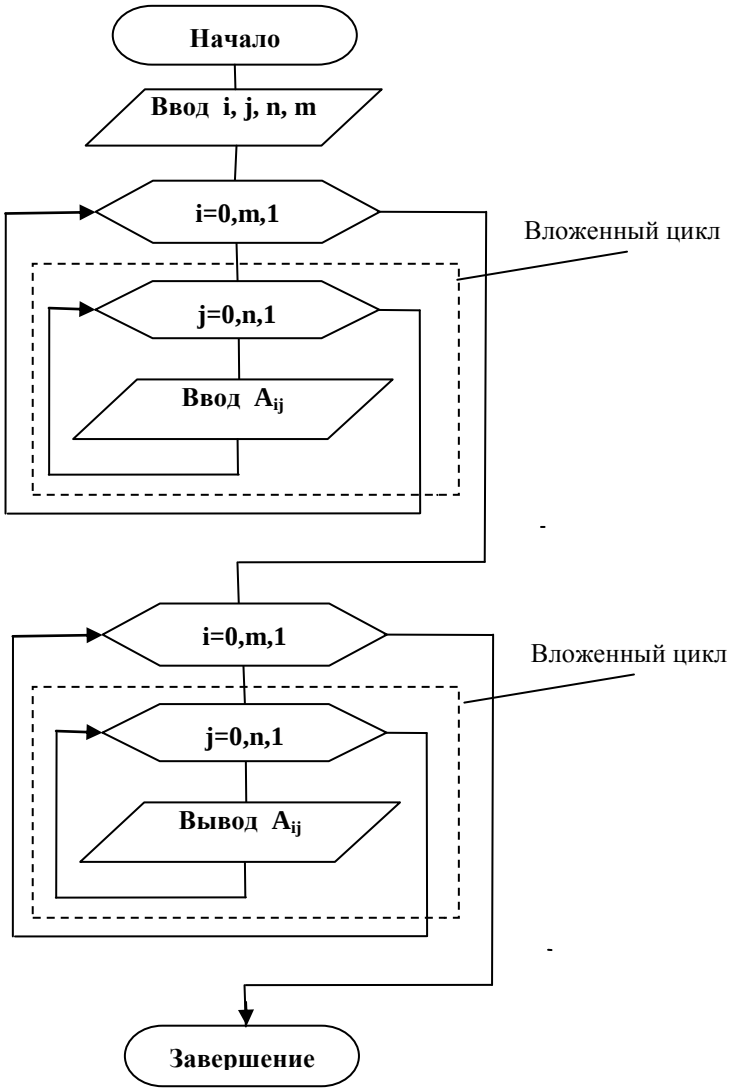


Рис. 6.1. Блок-схема алгоритма поэлементного
ввода и вывода матрицы с использованием цикла **for**

После запуска программы и ввода данных экран дисплея должен иметь вид:

```
Vvedite chislo strok i stolbcov matricu:
3  4
Vvedite postrochno elementu matricu:
1 2 3 4 5 6 7 8 9 10 11 12
Matrica A
1    2    3     4
5    6    7     8
9    10   11    12
```

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр6**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 6.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr6-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr>**, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр6**. Над вставленным рисунком проставьте номер задания – **6-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

3. Исследование различных операций в двухмерных массивах в среде Visual C++ 2010

Элементам массива могут быть **присвоены** значения выражений. При этом элементы массива и значения выражений должны иметь один и тот же тип.

Например, объявлен массив

```
double A[3][4];
```

тогда возможна запись

```
A[0][0]=3.5;
A[1][3]=0;
A[2][1]=a*x+b;
```

Копирование – это присваивание значений элементов одного массива элементам другого массива. При копировании оба массива должны иметь одинаковый размер и тип элементов. Копирование массива **A** в **B** будет иметь вид:

```
for (i=0; i<N; i++) B[i][j]= A[i][j];
```

Задание 6.2. Исследовать операцию присваивания двухмерных массивов, а также операцию транспонирования матрицы. Задана матрица целых чисел **A** размерностью 3×4. Необходимо транспонировать матрицу **A**, т. е. поменять местами ее строки и столбцы, а затем рассчитать матрицу **B**, элементы которой определяются по формуле

$$B_{ij} = \sqrt[3]{A_{ij}^2} + \sin^2(A^3) .$$

Элементы исходной матрицы **A** ввести непосредственно в программном коде. Исходную матрицу **A**, транспонированную **AT** и полученную матрицу **B** вывести на экран. Для наглядности примем

$$A(3,4) = \begin{vmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{vmatrix} .$$

Блок-схема алгоритма решения данной задачи приведена на рис. 6.2.

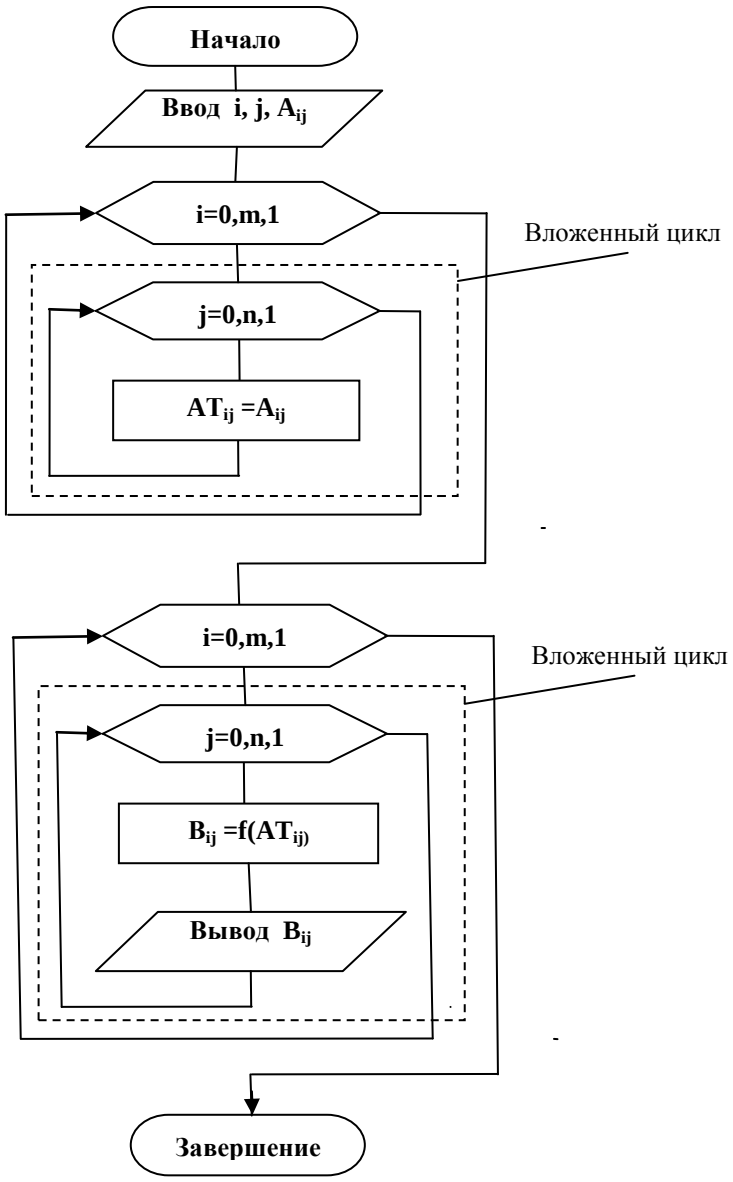


Рис. 6.2. Блок-схема алгоритма транспонирования и присваивания матрицы

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j;
double AT[4][3], B[4][3];                  //Объявление матриц AT и B
double A[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};  //Инициализация исходной матрицы A
cout<<"Matrica A"<<endl;                   //Вывод имени матрицы
   for(i=0; i<3; i++)                      //Внешний цикл вывода элементов
```

```
                                        //матрицы  A
   {
      for(j=0; j<4; j++)                //Вложенный цикл для вывода элементов
матрицы A
         cout<<A[i][j]<<"   ";          //Вывод элементов исходной матрицы A
   cout<<endl;
      }
//Транспонирование исходной матрицы
   for(i=0; i<3; i++)                   //Внешний цикл для транспонирования
      for(j=0; j<4; j++)                // Вложенный цикл для транспонирования
         AT[j][i]=A[i][j];              //Операция транспонирования   матрицы A
cout<<"Matrica AT = A transponirovannaya"<<endl;
   for(i=0; i<4; i++)                   //Начало цикла вывода матрицы AT
   {
      for(j=0; j<3; j++)                //Вложенный цикл вывода матрицы AT
         cout<<AT[i][j]<<"  ";          //Вывод элементов матрицы AT
   cout<<endl;
   }
cout<<"Matrica B = f(AT)"<<endl;
//Вычисление матрицы B
   for(i=0; i<4; i++)                   //Внешний цикл для поэлементного
                                        //вычисления матрицы B
   {
      for(j=0; j<3; j++)                // Вложенный цикл для поэлементного
                                        //вычисления матрицы B
      {
      B[i][j]=5*pow(sin(pow(AT[i][j],3)),2)+pow((AT[i][j]*AT[i][j]),1./3.);
      cout<<B[i][j]<<"  ";              //Вывод ij - го элемента матрицы B
      }
         cout<<endl;
   }
getch();
return 0;
}
```

После запуска программы на выполнение экран дисплея должен иметь вид:

```
Matrica A
1   2   3   4
5   6   7   8
9   10   11   12
Matrica AT = A transponirovannaya
1  5  9
2  6  10
3  7  11
4  8  12
Matrica B = f(AT)
4.54037  4.82155  4.43915
6.48155  5.72441  8.06024
6.65336  5.09899  8.64416
6.75208  4.03162  5.31802
```

Исследуем примеры программ, выполняющих различные действия над двухмерными массивами.

Задание 6.3. Самостоятельно разработать программу для вычисления матрицы **Y** размера **M**×**N** по формуле для своего варианта (таб. 6.1). Определите свой номер варианта как номер компьютера.

По аналогии с заданием 6.1 задать с запасом **M**=10 и **N**=10. Размер матрицы (3×4) и ее элементы (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) ввести с клавиатуры. Вывести на экран исходную и вычисленную матрицы.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++.

Задание выполните для двух произвольных значений **x** из двух интервалов.

Таблица6.1. Исходные данные и формулы для расчета Y (Задание 6.3)

№ вар.	[Y]=f([X])	№ вар.	[Y]=f([X])
1	$y_{ij}=\begin{cases}\cos^2 x_{ij}+2\sin x_{ij}, & \text{если } x<1;\\ 0{,}25x_{ij}^{\,3}-3x_{ij}, & \text{если } x\ge 1;\end{cases}$	10	$y_{ij}=\begin{cases}\log_3 5x_{ij}^2+\sqrt{x_{ij}^3}, & \text{если } x<0;\\ e^{2x_{ij}}+0{,}3x_{ij}^3, & \text{если } x\ge 0;\end{cases}$
2	$y_{ij}=\begin{cases}\log_3 5x_{ij}^2+\sqrt{x_{ij}^3}, & \text{если } x<0;\\ e^{2x_{ij}}+0{,}23x_{ij}^6, & \text{если } x\ge 0;\end{cases}$	11	$y_{ij}=\begin{cases}1{,}56+\sin^2 x_{ij}, & \text{если } x<1\\ 0{,}19x_{ij}^{\,3}-x_{ij}, & \text{если } x\ge 1\end{cases}$
3	$y_{ij}=\begin{cases}1{,}56+\sin^2 x_{ij}, & \text{если } x<1;\\ 0{,}23x_{ij}^{\,3}-x_{ij}, & \text{если } x\ge 1;\end{cases}$	12	$y_{ij}=\begin{cases}1{,}2x_{ij}^{\,2}+2\sqrt[3]{3}x_{ij}, & \text{если } x<1\\ \cos^3 x_{ij}, & \text{если } x\ge 1\end{cases}$
4	$y_{ij}=\begin{cases}0{,}12x_{ij}^{\,2}+2\sqrt[3]{3}x_{ij}, & \text{если } x<1;\\ \sin^3 x_{ij}, & \text{если } x\ge 1;\end{cases}$	13	$y_{ij}=\begin{cases}\cos^2 x_{ij}+2\sin x_{ij}, & \text{если } x<1;\\ 0{,}32x_{ij}^{\,2}-3x_{ij}, & \text{если } x\ge 1;\end{cases}$
5	$y_{ij}=\begin{cases}\cos^2 x_{ij}+2\sin x_{ij}, & \text{если } x<1;\\ 0{,}14x_{ij}^{\,3}-3x_{ij}, & \text{если } x\ge 1;\end{cases}$	14	$y_{ij}=\begin{cases}2x_{ij}^3+4\sin x_{ij}, & \text{если } x<0\\ \sin^2 x_{ij}^3, & \text{если } x\ge 0\end{cases}$

6	$y_{ij}=\begin{cases}2x_{ij}^3+4\sin x_{ij}, & \text{если } x<0;\\ \cos^2 x_{ij}^3, & \text{если } x\ge 0;\end{cases}$	15	$y_{ij}=\begin{cases}2\cos x_{ij}^3+4\sin x_{ij}, & \text{если } x<0;\\ x_{ij}^6, & \text{если } x\ge 0;\end{cases}$
7	$y_{ij}=\begin{cases}2\cos x_{ij}^3+4\sin x_{ij}, & \text{если } x<0;\\ x_{ij}^6, & \text{если } x\ge 0;\end{cases}$	16	$y_{ij}=\begin{cases}5x_{ij}^2+\ln x_{ij}, & \text{если } x<0;\\ 6.8^4-\sqrt{x_i}, & \text{если } x\ge 0;\end{cases}$
8	$y_{ij}=\begin{cases}5x_{ij}^2+\ln x_{ij}, & \text{если } x<0;\\ \lg x_{ij}^4+3x_{ij}^6, & \text{если } x\ge 0;\end{cases}$	17	$y_{ij}=\begin{cases}0{,}34\sin x_{ij}^2+2\sqrt[3]{12{,}1x_{ij}}, & \text{если } x<1\\ \cos^3 x_{i,}, & \text{если } x\ge 1;\end{cases}$
9	$y_{ij}=\begin{cases}0{,}234x_{ij}^2+2\sqrt[3]{12.1x_{ij}}, & \text{если } x<1;\\ \sin^3 x_{ij}, & \text{если } x\ge 1;\end{cases}$	18	$y_{ij}=\begin{cases}\cos^2 x_{ij}+2\sin x_{ij}, & \text{если } x<1;\\ 0{,}25x_{ij}^3-3x_{ij}, & \text{если } x\ge 1;\end{cases}$

Задание 6.4. Исследовать операцию поиска максимального элемента матрицы. Задана матрица **A** из 20 произвольных чисел размера 4×5. Необходимо найти максимальный элемент матрицы **A** и номера его строки и столбца (индексы элемента матрицы). Элементы матрицы **A** ввести в программе прямым образом. На экран вывести исходную матрицу **A**, максимальный элемент и его индексы.

Блок-схема алгоритма решения данной задачи приведена на рис. 6.2.

Необходимо напомнить, что при решении каких-либо задач с использованием различных языков программирования прежде всего необходимо разработать блок-схему алгоритма решения, а уже после этого – программный код, реализующий данный алгоритм. В данном случае (и в некоторых других) блок-схема алгоритма решения задачи находится после соответствующего программного кода из-за особенностей компоновки текста Методических указаний. Студенты же при решении самостоятельных заданий в первую очередь обязаны разработать и начертить в отчете блок-схему алгоритма.

Для решения задачи необходимо ввести и отладить следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;ij
int _tmain(int argc, _TCHAR* argv[])
{
int i, j, m, n;
```

```
int A[4][5]={1,2,18,3,20,4,5,6,7,8,9,10,19,11,12,13,14,15,16,17};   //Инициализация
                                                                    //матрицы A
int B;
cout<<"Matrica A"<<endl;
   for(i=0; i<4; i++)                    //Внешний цикл вывода  матрицы A
   {
      for(j=0; j<5; j++)                 //Внутренний цикл вывода  матрицы A
      cout<<A[i][j]<<"\t";               //Вывод  элемента  матрицы A
      cout<<endl;
      }
B=A[0][0];                               //Начальное значение B для поиска
   for(i=0; i<4; i++)                    //Внешний  цикл по перебору элементов
матрицы A
      for(j=0; j<5; j++)                 //Внутренний  цикл по перебору элементов
матрицы A
         if(A[i][j]>B)                   //Выбор большего элемента на j-ом шаге
         {
            B=A[i][j];                   //Запоминание  в B большего элемента A_ij
            n=i;                         //Запоминание в  n номера строки i
            m=j;                         //Запоминание в m номера столбца j
         }
cout<<endl<<"Max= "<<B;                  //Вывод  максимального элемента
cout<<"   Stroka  "<<n+1;                //Вывод  номера строки
cout<<"   Stolbec  "<<m+1<<endl;         //Вывод   номера столбца
getch();
return 0;
}
```

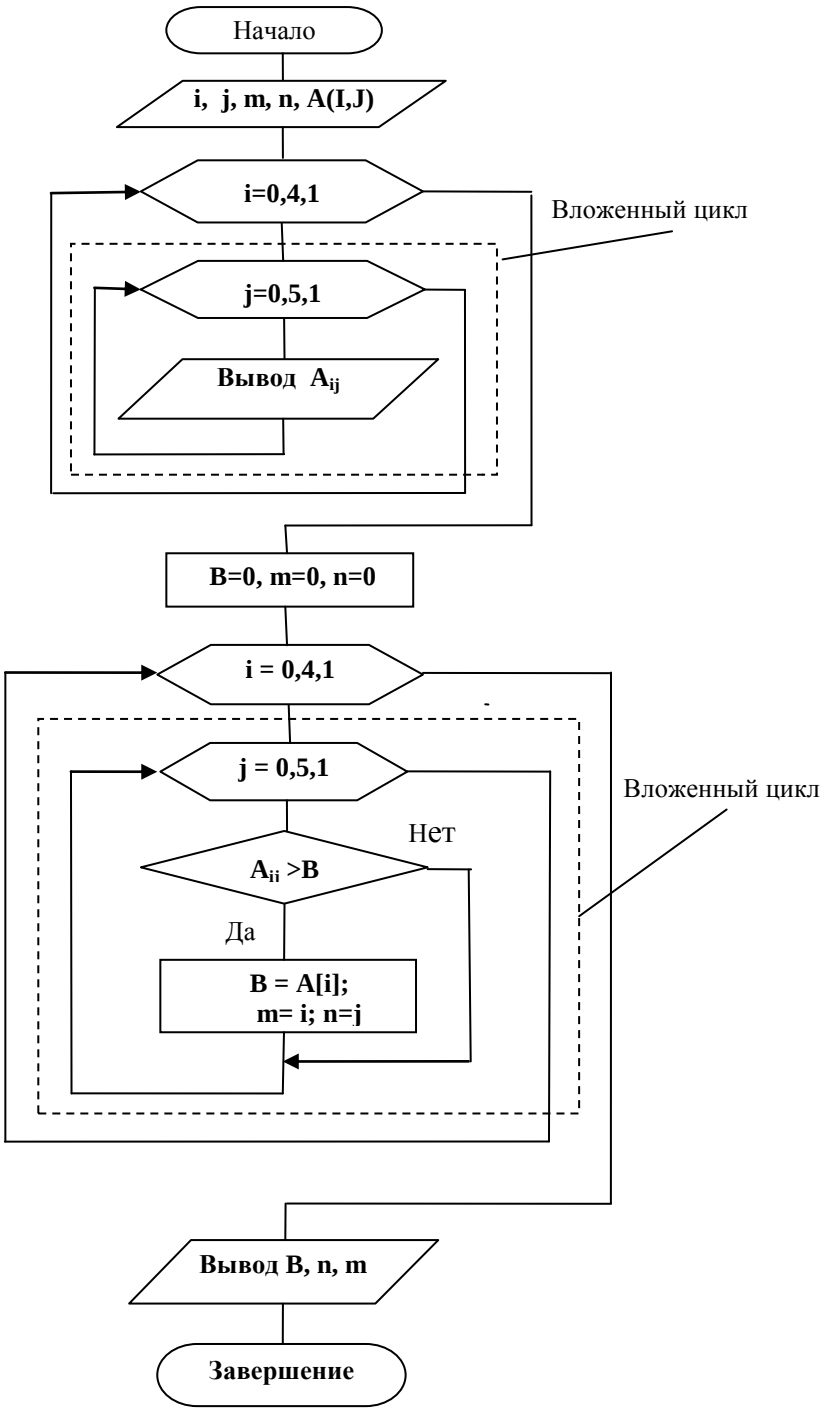


Рис. 6.3. Блок-схема алгоритма поиска максимального элемента матрицы

Экран после выполнения программы должен иметь следующий вид:

Matrica A
1 2 18 3 20
4 5 6 7 8
9 10 19 11 12
13 14 15 16 17
Max= 20 Stroka 1 Stolbec 5

Задание 6.5. Исследовать способы ввода и вывода двухмерных массивов, а также следующие действия с матрицей: найти минимальный элемент каждой строки матрицы **A**, составить из них вектор **R** (одномерный массив) и определить суммы его четных и нечетных элементов. Элементы заданной матрицы из 20 произвольных чисел размера 4×5 (см. в результатах) ввести с клавиатуры. На экран вывести исходную матрицу **A**, вектор **R** и вычисленные суммы.

Блок-схема алгоритма решения данной задачи не приводится, т. к. она аналогична блок-схеме на рис. 6.3.

Для решения необходимо ввести и отладить следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int M=10, N=10;                  //Возможное число строк и столбцов матрицы
int A[M][N], R[M];                     //Декларирование  матрицы A и вектора R
int i, j, k, m, S1, S2;                // Декларирование индексов и вспомогатель-
                                       //ных переменных
cout<<"Vvedite chislo strok  i stolbcov matricu  A :"<<endl;
cin>>m>>k;                             //Ввод реальных размеров исходной матрицы
cout<<"Vvedite postrochno elementu matricu  A :"<<endl;
   for(i=0; i<m; i++)                  //Внешний цикл ввода элементов матрицы A
      for(j=0; j<k; j++)               //Вложенный цикл ввода элементов матрицы
           cin>>A[i][j];               //Ввод ij–го элемента исходной матрицы A
//Поиск минимального элемента каждой строки исходной матрицы A
      for(i=0; i<m; i++)
      {                                //Начало составного оператора в цикле
         R[i]=A[i][0];                 //Задание начального значения минимального
                                       //элемента строки
         for(j=0; j<k; j++)            //Определение минимального элемента
            if(A[i][j]<R[i])           //каждой строки исходной матрицы A
               R[i]=A[i][j];           //Запись его в массив R
      }                                //Конец составного оператора внешнего цикл
S1=0;                                  //Задание начальных значений сумм четных
S2=0;                                  //и нечетных элементов  вектора R
//Операции по определению сумм четных и нечетных элементов массива R
   for(i=0; i<m; i++)
   {                                   //Начало составного оператора в цикле
      if(R[i]%2==0)                    //Условие, есть ли остаток от деления на 2
         S2=S2+R[i];                   //Определение  суммы четных элементов
      else S1=S1+R[i];                 //Определение суммы нечетных элементов
   }                                   //Конец составного оператора в цикле
cout<<"Matrica A"<<endl;
```

```
   for(i=0; i<m; i++)                  //Внешний  цикл вывода элементов матрицы A
   {                                   //Начало составного оператора в цикле
      for(j=0; j<k; j++)
         cout<<A[i][j]<<"    ";        //Вывод элементов матрицы A
   cout<<endl;
   }                                   //Конец составного оператора в цикле
cout<<endl<<"Vektor R"<<endl;
   for(i=0; i<m; i++)
      cout<<R[i]<<"    ";             //Вывод i–го элемента вектора R
cout<<endl;
cout<<"S1= "<<S1<<endl;                //Вывод суммы S1 четных элементов
cout<<"S2= "<<S2<<endl;                //Вывод суммы S2 нечетных элементов
getch();
return 0;
}
```

В результате выполнения вышеприведенной программы получим следующие результаты:

```
Vvedite chislo strok  i stolbcov matricu  A :
4 5
Vvedite postrochno elementu matricu  A :
1 2 18 3 20
4 5 6 7 8
9 10 19 11 12
13 14 17 15 16
Matrica A
1       2       18      3       20
4       5       6       7       8
9       10      19      11      12
13      14      17      15      16

Vektor R
1    4    9    13
S1= 23
S2= 4
```

Задание 6.6. Задана квадратная матрица 4х4 (элементы – числа подряд от 1 до 16). Самостоятельно разработать программу в соответствии со своим вариантом (таб. 6.2). Номер варианта определяется по номеру компьютера.

Ввод исходной матрицы запрограммировать с клавиатуры. Исходную матрицу, полученную матрицу и другие результаты вывести на экран.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++.

Таблица 6.2 Задание для самостоятельного программирования

№	Условие задания
1	Заменить нолями все элементы главной диагонали заданной матрицы и нижележащие элементы
2	Найти сумму всех ее элементов и заменить ею элементы главной диагонали заданной матрицы

3	Найти сумму и произведение всех отрицательных элементов заданной матрицы
4	Найти сумму минимальных элементов каждого столбца заданной матрицы
5	Найти среднее арифметическое максимального и минимального значений элементов заданной матрицы
6	Найти среднее арифметическое каждого из столбцов заданной матрицы
7	Получить новую матрицу путем деления всех элементов заданной матрицы на ее максимальный элемент
8	Получить новую матрицу путем деления всех элементов заданной матрицы на ее минимальный элемент
9	Найти сумму и произведение всех негативных элементов заданной матрицы
10	Найти сумму первых четырех элементов заданной матрицы и заменить ею элементы первой строки.
11	Найти сумму максимальных элементов каждого столбца заданной матрицы
12	Получить новую матрицу вычитанием из каждого элемента заданной матрицы ее минимального элемента
13	Заменить элементы первой строки заданной матрицы элементами ее второго столбца
14	Найти сумму элементов четных столбцов заданной матрицы

Задание 6.7. Задана квадратная матрица 4x4 (элементы – числа подряд от 1 до 16). Составить программу в соответствии со своим вариантом (таб. 6.1). Номер варианта определяется по номеру компьютера плюс два. Ввод исходной матрицы осуществить в программе. Исходную матрицу, полученную матрицу и другие результаты вывести на экран.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++.

Контрольные вопросы

1. Каким образом в программе на С++ объявляется двухмерный массив?
    1. **double A[0..2][0..3];**
    2. **double A[2, 3];**
    3. **double A[2][3];**
2. В памяти компьютера двухмерный массив располагается…
    1. непрерывно строками;
    2. непрерывно столбцами;
    3. в столбец.
3. Возможно ли следующее объявление массивов в С++:

int i, j, k, m ;
int A[i][j], B[k][m];?

    1. Возможно;

2. Не возможно;
3. если **i>j**, а **k>m**.

4. Как в программе на С++ будет выведен на экран двухмерный массив **A[k][k]** оператором

```
for(i=0; i<k; i++)
   for(j=0; j<k; j++)
   cout<<A[i][j]<<endl;
```
?

1. В виде строки;
2. В виде столбца;
3. В виде матрицы.

5. Что нужно изменить в программе

```
for(i=0; i<k; i++)
   for(j=0; j<k; j++)
   cout<<A[i][j]<<endl;
```

чтобы массив **A[k][k]** был выведен на экран в виде строки?

1. Вместо **i** подставить **k**;
2. Убрать **<<endl**;
3. Вместо **j** подставить **k.**

6. Как будет выведен на экран двухмерный массив в программе на С++?

```
int  i, j, k, m;
for(i=0; i<k; i++)
   for ( j=0; j<m; j++)
   cout<<A[i][j]<<”  “;
```

1. В виде строки;
2. В виде столбца;
3. В виде матрицы.

7. Как будет выведен на экран двухмерный массив в программе на С++?

```
int  i, j, k, m
for(i=0; i<k; i++)
{
   for(j=0; j<m; j++)
   cout<<A[i][j];
   cout<<endl;
}
```

1. В виде строки;
2. В виде столбца;
3. В виде матрицы.

8. Что будет выведено на экран в фрагменте программы на С++?

```
   for(i=0; i<n; i++)
   for (j=0; j<n; j++)
   cout<<A[0][j];
```

1. Первая строка матрицы, повторенная **n** раз;
2. Первый столбец матрицы, повторенный **n** раз;
3. Полная матрица.

9. Что нужно изменить в программе

```
int  i, j, k, m;
for(i=0; i<k; i++)
   for (j=0; j<m; j++)
      cout<<A[i][j]<<”  “;
```

для вывода массива **A[k][m]** на экран в виде матрицы?

1. Вместо **i** подставить **k**;
2. Добавить **cout<<endl;**
3. Вместо **j** подставить **m**.

10. Почему при работе с массивами в С++ целесообразно использовать оператор цикла **for**?

1. Оператор цикла **for** безошибочен;
2. Известен размер массива, т. е. число повторений цикла;
3. Оператор цикла **for** быстродействующий.

11. Какое действие выполняет оператор

```
for (i=0; i<N; i++)
B[i][j]= A[i][j];
```

?

1. Выводит на экран матрицу **B[i][j]**;
2. Выводит на экран матрицу **A[i][j]**;
3. Копирует матрицу **A[i][j]** в матрицу **B[i][j]**.

Лабораторная работа № 7
РАЗРАБОТКА И ИССЛЕДОВАНИЕ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ В VISUAL C++ 2010

Цель работы**: получение навыков работы с функциями в языке C++, изучение способов передачи аргументов**

1. Назначение и объявление функций в среде Visual C++ 2010

Как правило, основную часть программного кода в C++ составляют функции. Функции позволяют разбивать программу на отдельные автономные блоки. Любая программа содержит, по крайней мере, одну функцию (главную) - **tmain( ).**

Для создания правильного кода компилятору необходимо сообщить в начале программы имя функции, тип возвращаемого результата, а также количество и типы аргументов. Для этой цели в C++ используется так называемый **прототип функции.**

Прототип функции задается следующим образом:

ТипРезультата ИмяФункции (ТипПараметра1 [ИмяПараметра1], …);

Использование прототипа функции является **объявлением функции**. Чаще всего прототип функции совпадает с заглавием функции. В отличие от заглавия функции в ее описании, прототип заканчивается (!) **точкой с запятой.**

Имена формальных параметров функции при ее объявлении не играют роли. Поэтому прототип функции может выглядеть следующим образом:

```
int function(int a, float b, float c);
```

или

```
int function(int, float, float);
```

Два этих объявления функции **function** равносильны.

2. Описание функций в Visual C++ 2010

Основная форма описания или **программный код** функции имеет следующий вид:

```
Тип ИмяФункции (ТипПараметра1 ИмяПараметра1, …)
{
Тело функции
}
```

Описание функци состоит из заглавия функции и тела функции. Все исследованные нами выше программы имели по умолчанию такое описание главной функции:

```
int _tmain(int argc, _TCHAR* argv[])
{
Тело функции
}
```

В заглавии **Тип** перед именем функции определяет тип значения, которое возвращает функция. Если тип не указан, то по умолчанию предусматривается, что функция возвращает целое значение (тип **int**).

Список параметров состоит из перечня типов и имен параметров, разделенных запятыми. Функция может не иметь параметров, но круглые скобки необходимы всегда.

В списке параметров для каждого параметра должен быть указан тип. Например,

function (int x, int в, float z) - правильный список параметров;

function (int x, в, float z) - неправильный список параметров.

В теле функции обязательно должен присутствовать оператор **return** с параметром того же типа, что и возвращаемое значение.

Оператор **return** имеет два варианта использования.

1. Вызывает немедленный выход из функции и возвращение в программу, которая ее вызвала.

2. Используется для возвращения значения функции.

Если возвращаемое значение не используется в дальнейшем в программе, то оператор **return** следует без параметра или **вообще может быть опущен**. В этом случае возвращение в программу осуществляется после достижения закрывающейся скобки "**}**".

В случае, когда оператора **return** в теле функции нет или за ним нет значения, то значение, возвращаемое функцией, неизвестно (не определено). Если функция должна возвращать значение, но не делает этого, компилятор выдает предупреждение. Все функции, которые возвращают значение, могут использоваться в выражениях языка C++.

Функция может вызывать другие функции (одну или несколько). А те, в свою очередь, проводить вызов третьих и т.д. Кроме того, функция может вызывать саму себя. Это явление в программировании называется **рекурсией.**

Любая программа в среде Visual C++ 2010 обязательно включает главную функцию **tmain()**. С этой функции начинается выполнение программы.

3. Вызов функций в Visual C++ 2010

Для того чтобы функция выполняла определенные действия в программе, она должна быть вызвана. Функция выполняется только при обращении к ней. По окончании работы функция возвращает в основную программу в качестве результата значение некоторой переменной и т. п.

Вызов функции осуществляется путем указания в программе ее имени (идентификатора), за которым в круглых скобках следует список аргументов, разделенных запятыми:

ИмяФункции(аргумент 1, аргумент 2, … аргумент N) .

Каждый аргумент функции является **переменной, выражением или константой**. Они передаются в тело функции для последующего использования в вычислительном процессе. Список аргументов может быть пустым.

4. Передача аргументов функции в Visual C++ 2010

Существует два способа передачи аргументов функции в C++: по значению и по ссылке.

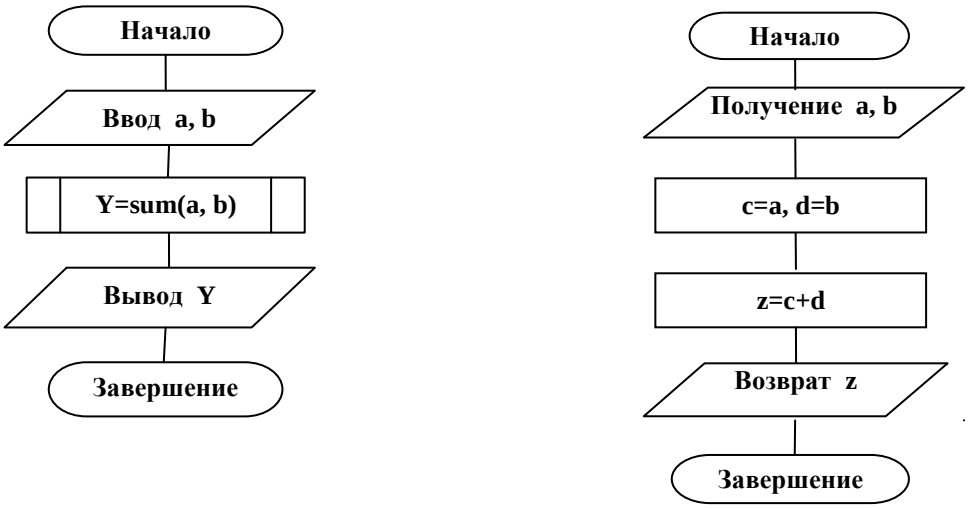
Когда происходит передача переменной-аргумента по значению, в функции создается локальная переменная с именем аргумента, в которую записывается его значение. Внутри функции может измениться значение этой переменной, но не самого аргумента.

Тип каждого фактического параметра (константы или переменной) в инструкции вызова функции должен совпадать с типом соответствующего формального параметра, указанного в объявлении функции.

Если параметр функции используется для возвращения результата, то в объявлении функции этот параметр должен быть ссылкой, а в инструкции вызова функции как фактический параметр должен быть указан адрес переменной (передача аргументов по ссылке).

Задание 7.1. Исследовать программу, вычисляющую **Y** по формуле **Y=a+b**, в которой расчет **Y** оформлен в виде функции.

Блок-схема алгоритма решения данной задачи приведена на рис. 7.1.



а) б)

Рис. 7.1. Блок-схема алгоритма вычисления **Y**
по формуле **Y=a+b** с использованием функции:
а) основная программа; б) функция **sum(double c, double d)**

Программа, использующая функцию для вычисления **Y** по формуле **Y=a+b**, будет иметь следующий вид:

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

double sum(double c, double d);        //Объявление (прототип) ф-ии вычисления
                                       //суммы двух чисел
int _tmain(int argc, _TCHAR* argv[])
```

```
{
double Y, a, b;
cout<<"Vvedite a,  b :"<<endl;
cin>>a>>b;                          //Ввод чисел a и b
Y=sum(a, b);                        //Вызов ф-ии, вычисляющей сумму двух
чисел
cout<<endl<<"Y = "<<Y<<endl;        //Вывод значения Y
getch();
return 0;
}
double sum(double c, double d)      //Описание функции вычисления z=c+d
{
   double z;
   z=c+d;                           //Вычисление  z=c+d
   return z;                        //Возврат значения  z  в основную программу
}
```

Результат выполнения программы следующий:

Vvedite a, b :
5.4 4.3
Y=9.8

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр7**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 7.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr7-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр8**. Над вставленным рисунком проставьте номер задания – **7-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

Задание 7.2. Самостоятельно создайте проект для вычисления **Y** по заданной формуле с использованием функции в соответствии со своим вариантом (номер компьютера). Варианты заданий находятся в таблице 7.1.

Перед разработкой программного кода запишите заданную формулу в отчете по лабораторной работе на языке С++ и начертите блок-схему алгоритма решения поставленной задачи.

Таблица 7.1 Исходные данные и формулы для расчета Y

№ варианта	Формула для расчета **Y**	Значения **x** и **a**

1	$Y = \frac{\sin\frac{x+1}{4}}{\sin^2 5x + e^{3a}}$	x=0,7; a=1,5
2	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,82; a=2,55
3	$Y = \frac{tg^3(x+a) - \arccos^2(x+a)}{(x+a)^4}$	x=0,68; a=5,55
4	$Y = ctg\frac{1-3x}{1+2x} + \cos^2 5x + e^{7a}$	x=0,35; a=4,8
5	$Y = \frac{\cos^3(x+a) - 7(x+a)}{tg(x+a)^4}$	x=0,62; a=4,55
6	$Y = \frac{\cos\frac{3a+1}{4}}{\sin^3 3x + e^{4a}}$	x=0,43; a=2,6
7	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,74; a=1,55
8	$Y = \frac{tg\frac{4a^2+1}{4}}{\cos^3 2x + e^{2a}}$	x=0,14; a=2,55
9	$Y = \sin\frac{1-x}{1+x} + tg^4 5x + e^{5a}$	x=0,34; a=4,95
10	$Y = \frac{\sin^3(x+a) - \arccos^2(x+a)}{\cos(x+a)^4}$	x=0,14; a=2,95
11	$Y = \frac{ctg\frac{x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=0,75; a=1,9
12	$Y = \frac{ctg^3(3x+a) - \sin^2(x+7a)}{(5x+a)^3}$	x=0,44; a=2,95
13	$Y = \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 3x + e^{3a}}$	x=0,27; a=1,9
14	$Y = \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 3x + e^{3a}}$	x=0,49; a=3,7

15	$Y = \frac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x=0,83; a=4,7
16	$Y = \frac{\operatorname{arcctg}\frac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=0,37; a=2,75
17	$Y = \frac{tg^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	x=0,13; a=0,7
18	$Y = tg\frac{1-x}{1+x} + \sin^2 5x + e^{5a}$	x=0,5; a=3,5

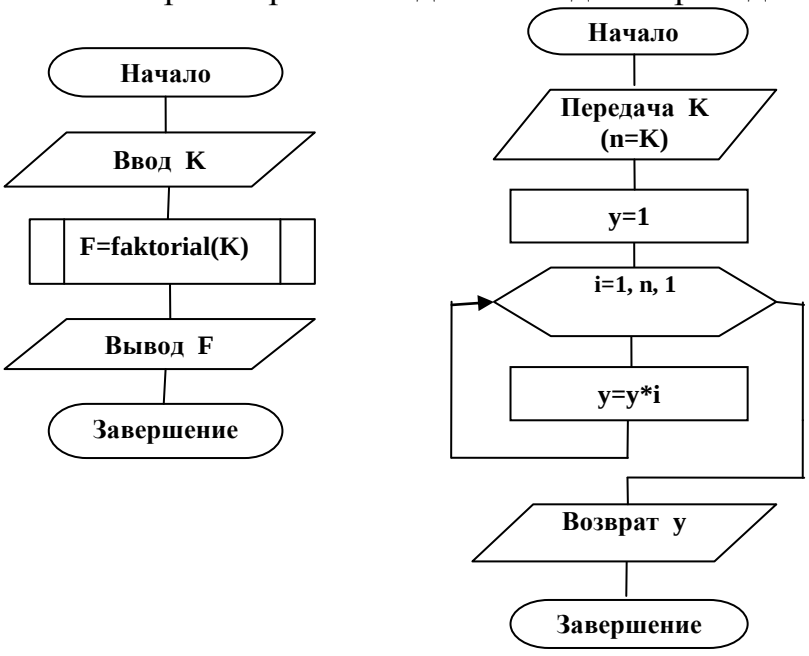
Задание 7.3. Исследовать программу, использующую функцию вычисления факториала числа:

F = n! .

Для составления программы данное выражение запишем следующим образом:

$$\mathbf{F = n! = 1*2*3*4* \ldots *n} = \prod_{j=1}^{n} j;$$

Блок-схема алгоритма решения данной задачи приведена на рис. 7.2.



а)

б)

Рис. 7.2. Блок-схема алгоритма вычисления
факториала числа с использованием функции:
а) основная программа; б) функция **faktorial(int n)**

Программа для функции, вычисляющей факториал числа **n**, будет иметь следующий вид:

```
#include "stdafx.h"
```

```
#include <conio.h>
#include "iostream"
using namespace std;
int faktorial(int  n);                          //Объявление (прототип) функции
                                                //вычисления факториала
int _tmain(int argc, _TCHAR* argv[])
{
int K;
cout<<"Vvedite K :"<<endl;
cin>>K;                                         //Ввод числа K
   F=faktorial(K);                              //Вызов функции, вычисляющей
                                                //факториал числа
cout<<endl<<"Faktorial K= "<<K<<endl;           //Вывод значения факториала числа
getch();
return 0;
}
int faktorial(int  n)                           //Описание функции вычисления
                                                //факториала числа
{
int j, y;
y=1;
   for(j=1; j<=n; j++)                          //Цикл для вычисления факториала
   y=y*j;
return y;                                       //Возврат значения  y  в основную
                                                //программу
}
```

Результат выполнения программы следующий:

Vvedite K:
5
Faktorial K= 120

Задание 7.4. Исследовать программу для вычисления числа соединений **R** из **N** элементов по **M** по формуле

$$\mathbf{R} = C_N^M = \frac{N!}{M!(N-M)!};$$

где расчет **R** производится с использованием функции.

Воспользуемся созданной нами и исследованной в **Задании 7.3** функцией **factorial(int n)** для вычисления факториала числа и будем обращаться к этой функции непосредственно в формуле для расчета **R**.

Блок-схема алгоритма решения задачи аналогична приведенной на рис. 7.2.

Программный код для вычисления **R** следующий:

```
#include "stdafx.h"
```

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

int faktorial(int  n);                          //Объявление (прототип)  функции
                                                //вычисления факториала
int _tmain(int argc, _TCHAR* argv[])
{
int N,M,R;
cout<<"Vvedite N i M (N>M) :"<<endl;
cin>>N>>M;                                      //Ввод значений  N и M
R=faktorial(N)/(faktorial(M)*faktorial(N-M)); //Формула для R (с трехкрат-
                                                //ным вызовом функции faktorial)
cout<<endl<<"R= "<<R<<endl;                     //Вывод значения R
getch();
return 0;
}
int faktorial(int n)                            //Описание функции вычисления
                                                //факториала числа
{
   int j, y;
   y=1;
      for(j=1; j<=n; j++)
      y=y*j;
   return y;                                    //Возвращение в основную
                                                //программу значения  y
}
```

Результат выполнения программы следующий:

```
Vvedite N и M:
8  4
R= 70
```

Задание 7.5. Самостоятельно разработать алгоритм и программу для вычисления значения **Y** с использованием оператора **for** (таб. 7.2). Вычисление **Y** оформить как функцию. Перед разработкой проекта начертить в отчете блок-схему алгоритма решения задачи и записать формулу для расчета **Y**. Определить свой номер варианта как номер компьютера.

Таблица 7.2 Исходные данные и формулы для расчета Y (Задание 7.5)

№ варианта	Формула для расчета **Y**	Значения **x, a**

1	$Y = \prod_{k=0}^{6} \frac{tg^3(x+a) - \arccos^2(x+a)}{k(x+a)^4}$	x=2,7; a=1,33
2	$Y = \sum_{k=0}^{5} \left(ctg\frac{1-3x}{1+2x} + \cos^2 5x + ke^{3a} \right)$	x=0,7; a=0,46
3	$Y = \prod_{k=0}^{8} \frac{\cos^3(x+a) - k7(x+a)}{tg(x+a)^4}$	x=2,7; a=1,82
4	$Y = \sum_{k=1}^{7} \frac{\cos\left(k\frac{3a+1}{4}\right)}{\sin^3 3x + e^{4a}}$	x=0,45; a=0,82
5	$Y = \prod_{k=1}^{6} \frac{\sin^3(x+a) - \cos^2(x+a)}{k(x+a)^4}$	x=2,1; a=1,47
6	$Y = \sum_{k=0}^{5} \frac{tg\frac{4a^2+1}{4}}{\cos^3 2x + ke^{2a}}$	x=2,1; a=1,34
7	$Y = \prod_{k=0}^{8} \left(\sin\frac{1-x}{1+x} + ktg^4 5x + e^{5a} \right)$	x=0,7; a=1,28
8	$Y = \sum_{k=1}^{5} \frac{\sin^3(x+a) - k\arccos^2(x+a)}{\cos(x+a)^4}$	x=2,2; a=0,66
9	$Y = \prod_{k=0}^{7} \frac{ctg\left(k\frac{x^3+1}{4}\right)}{\cos^2 5x + e^{3a}}$	x=1,45; a=1,12
10	$Y = \sum_{k=1}^{6} \frac{ctg^3(3x+a) - k\sin^2(x+7a)}{(5x+a0)^3}$	x=2,7; a=1,82
11	$Y = \prod_{k=1}^{4} \frac{\arcsin^3\frac{4x+1}{4}}{ctg^2 k(3x+e^{3a})}$	x=1,25; a=1,42
12	$Y = \sum_{k=0}^{6} \frac{\sin^3\frac{3x+1}{2}}{tg^2 5x + ke^{3a}}$	x=1,85; a=1,72
13	$Y = \prod_{k=1}^{8} \frac{\sin^3(x+a) - \cos^2(x+a)}{k(x+a)^4}$	x=1,48; a=1,19

14	$Y = \sum_{k=0}^{7} \frac{\operatorname{arcctg}\frac{2kx^3+1}{4}}{\cos^2 5x + e^{3a}}$	x=1,15; a=0,12
15	$Y = \prod_{k=1}^{7} \frac{tg^3(x+a) - 5k(\sin x + a)}{\sin^3(x+a)^4}$	x=2,45; a=2,12
16	$Y = \sum_{k=0}^{6} \left(k \times tg\frac{1-x}{1+x} + k \times \sin^2 5x + e^{5a} \right)$	x=2,25; a=1,88

5. Области действия и видимости переменных в программах в среде Visual C++ 2010

Область действия переменной – это часть или части программного кода, в которых данная переменная **определена** (доступна для действий в данном месте программы).

С точки зрения области действия переменных различают три типа переменных:

- локальные;
- глобальные;
-формальные.

Локальные переменные – это переменные, объявленные в середине блока, в частности, внутри описания функции. Локальная переменная доступна в середине блока, в котором она объявлена. Блок открывается и закрывается фигурными скобками. Область действия локальной переменной – данный блок или функция.

Формальные переменные (параметры) – это переменные, объявленные при описании функции как ее аргументы. Формальные параметры используются в теле функции, как локальные переменные. Область действия формальных параметров – тело функции.

Глобальные переменные – это переменные, объявленные в основной программе вне какой-либо функции. Они могут быть использованы в любом месте программы. Область действия глобальной переменной – вся программа.

Задание 7.6. Разработать программу для определения вероятности обслуживания заявки в системе массового обслуживания (СМО) по формуле

$$P = 1 - \alpha^n \Big/ n! \sum_{k=0}^{n} \frac{\alpha^k}{k!};$$

где: α – поток заявок на обслуживание; n – количество приборов обслуживания.

Введем дополнительную переменную S:

$$S = \sum_{k=0}^{n} \frac{\alpha^k}{k!};$$

Для решения поставленной задачи необходимо разработать программы двух функций для вычисления факториала числа и возведения числа в степень.

Функция вычисления факториала числа рассмотрена выше.

Разработаем программу для функции **st(…)** возведения в степень по формуле

$$B = C^R = C*C*C* \ldots *C = \prod_{j=0}^{R} C;$$

Тогда программа для вычисления вероятности обслуживания заявки примет вид:

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

float  st(float c, int R);                  //Объявление (прототип) функции
                                            //возведения в степень
int fact(int x);                            //Объявление (прототип)  функции
                                            //вычисления факториала
int _tmain(int argc, _TCHAR* argv[])
{
int N, k;
float Alpha, S, P;
cout<<"Vvedite Alpha i N :"<<endl;          //Ввод  α  и N
cin>>Alpha>>N;
S=0;
   for(k=0; k<=N; k++)
   S = S+st(Alpha,k)/factorial(k);          //Вычисление  S
P = 1- st(Alpha,N)/(fact(N)*S);             //Вычисление P
cout<<endl<<"P= "<<P<<endl;
}
float st(float c, int R)                    //Описание функции возведения
                                            //числа в степень
{
int j;                                      //Объявление параметра цикла
float y;                                    //Объявление переменной y
y=1;                                        //Присваивание y начального
                                            //значения для произведения
   for(j=0; j<=R; j++)                      //Цикл для вычисления y
   {
      if(j==0) y=1;
      else y=y*c;                           //Вычисление очередного y
   }
return y;                                   //Возвращение окончательного
                                            //значения y в программу
}
```

```
int factorial(int x)                    //Описание функции факториала
{
int j,y;
y=1;
   for(j=0; j<=x; j++)
   {
      if(j==0)
      y=1;
   else y=y*j;
}
return y;
}
```

Вид экрана после выполнения программы следующий:

```
Vvedite Alpha i N :
2  4
P= 0.904762
```

Из программы видно, что функции **fact(…)** и **st(…)** используют одни и те же **по названию** локальные переменные **j** и **y**, имеющие разные значения в зависимости от области использования (функции). Здесь четко выполняется принцип действия и видимости локальных переменных только в пределах текущих описаний функций.

6. Функции и массивы.

Если в качестве аргумента функции используется массив, то необходимо указать адрес начала массива и его размер.

Заглавие функции, обрабатывающей массив, необходимо записать следующим образом:

float function (float A[n]);

или

float function (float A[ ], int n);

В этом случае вызов функции **function** из основной программы запишется следующим образом:

function (B, k);

Задание 7.7. Задан массив **A** из **N** произвольных чисел. Сформировать новый массив **B**, каждый элемент которого равняется частному от деления соответствующего элемента массива **A** на его максимальный элемент. На экран вывести начальный массив **A**, его максимальный элемент и массив **B**. Поиск максимального элемента массива **A** оформить функцией.

Запрограммируем функцию нахождения максимального элемента массива и его номера и используем ее в разрабатываемой программе следующим образом:

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

```
#include "iostream"
using namespace std;
double Max(double B[],int n);          //Объявление (прототип)  функции
поиска макс. элемента
int _tmain(int argc, _TCHAR* argv[])
{
const int N=50;                        //Максимальный размер исходного массива
int i, k;                              //Объявление реального размера массива
double A[N], M;                        //Объявление массива A и переменной  M
cout<<"Vvedite razmer massiva :"<<endl;
cin>>k;                                //Ввод реального размера массива
cout<<" Vvedite massiv:"<<endl;
   for(i=0; i<k; i++)
   cin>>A[i];                          //Ввод исходного массива A
cout<<endl;
M=Max(A, k);                           //Обращение к функции поиска  Max()
cout<<" Ishodnui massiv "<<endl;
   for(i=0; i<k; i++)                  //Цикл для печати  исходного массива A
   cout<<A[i]<<' ';
cout<<endl<<"Max= "<<M<<endl;          //Вывод максимального  элемента
cout<<" Novui  massiv "<<endl;
   for(i=0; i<k; i++)                  //Цикл для расчета и вывода элементов
                                       //нового  массива
   cout<<A[i]/M<<' ';
cout<<endl;
getch();
return 0;
}
double Max(double B[],int n)           //Описание функции поиска максимального
                                       //элемента массива
{
int j;
double C=B[0];                         //Присваивание переменной C начального
                                       //знач-я (1-го элемента)
   for(j=0; j<n; j++)                  //Цикл для перебора элементов массива и
                                       //сравнения их с C
      if (B[j]>C) C=B[j];
return C;                              //Возвращение в основную программу
                                       //значения  C
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite razmer massiva :
7
 Vvedite massiv:
1 6 0 3 7 9 5

 Ishodnui massiv
1 6 0 3 7 9 5
Max= 9
 Novui massiv
0.111111 0.666667 0 0.333333 0.777778 1 0.555556
```

В качестве дополнительного задания разработать программу для выполнения **Задания 7.7**, в которой ввод и вывод элементов одномерного массива производится путем обращения к соответствующим функциям.

Задание 7.8. Задан массив **A** из **N** произвольных чисел. Исследовать программу для нахождения максимального элемента этого массива и его номера, которые вывести на экран. Поиск максимального элемента массива и его номера оформить функцией, а сами переменные использовать как глобальные, т. е. “видимые” и в основной программе, и в описаниях функций.

Запрограммируем функцию нахождения максимального элемента массива и его номера и используем ее в разрабатываемой программе следующим образом:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
double Max(double B[], int n);          //Объявление (прототип) функции Max()
double C;                               //Объявление глобальной переменной C
int k;                                  //Объявление глобальной  переменной k
int _tmain(int argc, _TCHAR* argv[])
{
const int N=50;                         //Объявление максимального размера
                                        //исходного массива
int i, m;                               //Объявление реального размера массива
double A[N];                            //Объявление исходного массива  A[N]
cout<<"Vvedite razmer massiva :"<<endl;
cin>>m;                                 //Ввод реального размера массива
cout<<endl<<"Vvedite massiv :"<<endl;
for(i=0; i<m; i++) cin>>A[i];           //Ввод элементов исходного массива A[i]
cout<<endl;
Max(A, m);                              //Обращение (вызов) к функции Max()
cout<<"Max= "<<C<<endl;                 //Вывод максимальн. элемента массива A
                                        //(глобальная переменная)
cout<<" Nomer  max "<<k<<endl;          //Вывод номера максимальн. элемента
getch();
return 0;
```

```
}
double Max(double B[], int n)        //Заголовок описания (кода) функции  Max
{
int j;                               //Объявление параметра цикла
C=B[0];                              //Глобальной переменной C присваиваем
                                     //начальное значение
   for(j=0; j<n;j++)                 //Цикл для поиска максимального элемента
      if(B[j]>C)
   {
   C=B[j];                           //Определяем максимальный элемент
   k=j;                              //Номер максимальн. элемента заносим в
                                     //глобальную переменную k
   }
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite razmer massiva :
8

Vvedite massiv :
11 23 15 7 8 12 4 9

Max= 23
 Nomer max 1
```

Задание 7.9. Самостоятельно разработать алгоритм и программу для выполнения следующего задания. (таб. 7.2). Вычисление **Y** оформить как функцию.

Задана исходная квадратная матрица 4x4 (элементы – числа подряд от 1 до 16). Самостоятельно составить программу в соответствии со своим вариантом (таб. 7.3). Номер варианта определяется по номеру компьютера.

Необходимую операцию с матрицей запрограммировать как функцию. Ввод исходной матрицы запрограммировать с клавиатуры. Исходную матрицу, полученную матрицу и другие результаты вывести на экран.

Перед разработкой проекта начертить в отчете блок-схему алгоритма решения задачи и записать формулу для расчета **Y**.

Таблица 7.3 Задание для самостоятельного программирования

№	Условие задания
1	Найти сумму и произведение всех отрицательных элементов заданной матрицы
2	Найти сумму минимальных элементов каждого столбца заданной матрицы
3	Найти среднее арифметическое максимального и минимального значений элементов заданной матрицы

4	Найти среднее арифметическое каждого из столбцов заданной матрицы
5	Получить новую матрицу путем деления всех элементов заданной матрицы на ее максимальный элемент
6	Получить новую матрицу путем деления всех элементов заданной матрицы на ее минимальный элемент
7	Найти сумму и произведение всех негативных элементов заданной матрицы
8	Найти сумму первых четырех элементов заданной матрицы и заменить ею элементы первой строки.
9	Найти сумму максимальных элементов каждого столбца заданной матрицы
10	Получить новую матрицу вычитанием из каждого элемента заданной матрицы ее минимального элемента
11	Заменить элементы первой строки заданной матрицы элементами ее второго столбца
12	Найти сумму элементов четных столбцов заданной матрицы
13	Найти сумму элементов главной диагонали и заменить ею последний столбец заданной матрицы
14	Найти сумму элементов четных строк заданной матрицы

Контрольные вопросы

1.Что такое функция?

2.В каких ситуациях используются функции?

3.Как объявляется функция?

4.Как описывается функция?

5.Какие способы передачи аргументов существуют и как они реализуются?

6. Укажите правильный прототип функции **FF**, использующей в качестве аргумента целый массив **W[n]**.

1) **double FF(double B[], int n);**
2) **double FF(int W[], int n);**
3) **FF(double B[], int n);**

7. Укажите правильное обращение к функции **FF**, использующей в качестве аргумента целый массив **W[n]**.

1) **func (int W[], n);**
2) **FF(W, n);**
3) **func (int W[][]);**

Лабораторная работа № 8

ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ СРЕДЫ VISUAL C++ 2010 ПО ОБРАБОТКЕ МАССИВОВ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ

1. Описание массивов символьных переменных в среде Visual C++ 2010

В языке C++ массив символьных переменных или **строка** - это одномерный массив типа **char**, состоящий из символов:

char A[11];

Символьная строка – это последовательность символов, дополненная специальным символом-ограничителем, указывающим конец строки. Ограничивающий символ записывается управляющей последовательностью **“\0”**. Для такой символьной строки применяют название “строка С” (было предложено разработчиком языка). В других ветвях языка C++ существуют другие представления символьных строк.

Каждый символ в строке занимает один байт.

Символьная константа **“\0”** , ограничивающая символьную строку, называется нулевым байтом. Ее следует учитывать при определении соответствующего массива символов: если строка должна содержать **N** символов, то в определении массива следует указать **N+1** элемент.

Например, определение

char A[11];

означает, что строка содержит 10 элементов типа **char** (символов), а последний байт зарезервирован для нулевого байта.

В качестве символов могут использоваться.

1. Прописные буквы латинского и российского алфавитов.
2. Строчные буквы латинского и российского алфавитов.
3. Цифры от 0 до 9.
4. Символы пунктуации: **. ;** и т. п.
5. Символьные константы.
6. Управляющие символы.
7. Пробел.
8. Шестнадцатеричные цифры.

Символьные массивы при их определении могут инициализироваться как обычный массив:

char A[13]={‘K’,’h’,’a’,’r’,’k’,’o’,’v’,’-‘,’2’,’0’,’1’,’4’};

а могут – как символьная строка, т. е. последовательность символов, заключенных в двойные кавычки:

char A[13]=”Kharkov-2014”;

Отличие этих двух способов заключается в том, что во втором случае автоматически будет прибавлен еще и нулевой байт. К тому же второй способ короче.

Для выделения места в памяти под символьный массив произвольного размера необходимо указать количество символов в строке (если оно известно) или задать явно больший размер массива.

char B[80]="Это инициализация массива символов";

В данном случае указан размер массива 80, хотя для размещения этой строки необходимо было указать 35 (с учетом нулевого байта).

Инициализировать символьный массив можно и без указания его размера:

char B[ ]= "Это инициализация массива символов";

В этом случае компилятор сам определит необходимый размер памяти под этот массив.

Задание 8.1. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char stroka[30], A[50];                 //Объявление символьных переменных
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
cin>>stroka;                            //Ввод символьной переменной  stroka
cout<<"Vu vveli stroku:"<<endl;
cout<<stroka<<endl;                     //Вывод символьной переменной  stroka

cout<<"Vvedite novuyu ctroku  <  30  simvolov::"<<endl;
cin>>stroka;                            //Ввод новой символьной переменной
stroka
cout<<"Vu vveli novuyu stroku: "<<stroka<<endl;     //Вывод символьной
                                                    //переменной  stroka
cout<<"Vvedite novuyu ctroku  <  50  simvolov::"<<endl;
cin>>A;                                 //Ввод символьной переменной  A
cout<<"Vu vveli novuyu stroku: "<<A<<endl;          //Вывод символьной
                                                    //переменной  A
cout<<"A0= "<<A[0]<<endl;               //Вывод 0-го элемента  переменной  A
cout<<"A8= "<<A[8]<<endl;               //Вывод 8-го элемента  переменной  A
getch();
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite ctroku  <  30  simvolov:
wwwwwwwwwwwqqqqqqqqq
Vu vveli stroku:
wwwwwwwwwwwqqqqqqqqq
Vvedite novuyu ctroku  <  30  simvolov::
aaaaaddddddffff
Vu vveli novuyu stroku: aaaaaddddddffff
Vvedite novuyu ctroku  <  50  simvolov::
ssssddddgggg
Vu vveli novuyu stroku: ssssddddgggg
A0= s
A8= g
```

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр8**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 8.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr8-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр8**. Над вставленным рисунком проставьте номер задания – **8-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

При вводе символов с пробелами, последние игнорируются операторами ввода >> и вывода <<. Поэтому при работе со строками вместо этих операторов целесообразней использовать следующую функцию:

getline(ИмяСимвольнойПеременной, РазмерСимвольнойПеременной);

где **ИмяСимвольнойПеременной** указывает на строку, в которую осуществляется ввод; **РазмерСимвольнойПеременной** – число символов, подлежащих вводу.

Задание 8.2. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
```

```
char stroka[30], A[50];                    //Объявление символьных переменных
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
   cin.getline(stroka,20);                 //Ввод символьной переменной  stroka
cout<<"Vu vveli stroku:"<<endl;
cout<<stroka<<endl;                        //Вывод символьной переменной stroka
cout<<"Vvedite novuyu ctroku  <  30  simvolov:"<<endl;
   cin.getline(A,20);                      //Ввод символьной переменной  A
cout<<"Vu vveli novuyu stroku: "<<endl<<A;  //Вывод символьной переменной  A
getch();
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Vvedite ctroku  <  30  simvolov:
www sss ttttt
Vu vveli stroku:
www sss ttttt
Vvedite novuyu ctroku  <  30  simvolov:
qqq yyyyyyy ppppppppp
Vu vveli novuyu stroku:
qqq yyyyyyy ppppppppp
```

При использовании функции **getline()** размер переменной меньше или равен размеру объявленной символьной строки.

Объявленная в вышеприведенной программе строка **stroka** может принять 70 символов. Например, если в функции **getline(stroka, 20)** указано число 20, то при вводе строки с 37 символами введется строка из 30 символов. Остальные символы будут отброшены.

2. Функции для обработки строк в среде Visual C++ 2010

Для работы со строками существуют специальные функции, описание которых находится в заглавном файле **string.h**, который необходимо включать в программу оператором **include**:

#include <string.h>;

Рассмотрим функции, которые используются наиболее часто.

2.1. Определение длины строки

Очень часто при работе со строками необходимо знать, сколько символов содержит строка. Для получения информации о длине строки используется функция **strlen()**. Вызов функции имеет вид:

strlen(Имя массива);

Функция возвращает значение на единицу меньшее, чем отводится под массив (без учета нулевого байта).

Задание 8.3. Исследовать программу, в которой вводятся и выводятся символьные переменные. При выполнении задания опробуйте ввод и вывод различных символов, в том числе с пробелами.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[80];
int k;
cout<<"Vvedite ctroku < 30 simvolov:"<<endl;
   cin.getline(A,30);                  //Вызов функции getline() для ввода массива A
cout<<"Vu vveli stroku: "<<endl<<A;
   k=strlen(A);                        //Вызов функции strlen(A) для определения
                                       //количества символов в массиве A
cout<<endl<<"k= "<<k<<endl;            //Вывод переменной k (кол. символов в A)
getch();
return 0;
}
```

Вид экрана после работы программы:

```
Vvedite ctroku  <  30  simvolov:
wwwwwwwww  RRRRRRRRR  xxxxxxxxxx
Vu vveli stroku:
wwwwwwwww  RRRRRRRRR  xxxxxxx
k= 29
```

2.2. Копирование и присоединение строк

Значения строк могут копироваться из одной строки в другую. Копирование осуществляется с помощью следующих функций.

Функция **strcpy(S1,S2)** используется для побайтного копирования строки **S2** в строку **S1**. Копирование прекращается при достижении нулевого байта. Поэтому длина строки **S1** должна быть достаточно большой, чтобы в нее поместилась строка **S2**.

Задание 8.4. Исследовать использование функции **strcpy()**.

```
#include <string.h>                    //Добавление библиотеч. файла для
                                       //работы со строками
#include "stdafx.h"
#include <conio.h>
#include "iostream"
```

```
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[80];
int k;
cout<<"Vvedite ctroku  <  30  simvolov:"<<endl;
cin.getline(A,30);                          //Ввод символьной переменной  A
cout<<"Vu vveli stroku: "<<endl<<A;
strcpy(A, "Proverka kopirovaniya");         //Вызов функции strcpy(A) для
                                            //копирования строки в строку
cout<<endl<<"Novaya stroka: "<<A;           //Вывод новой символьной
                                            //переменной  A
getch();
return 0;
}
```

Вид экрана после работы программы:

```
Vvedite ctroku  <  30  simvolov:
aaaaaaaaaaaaaaa ddddddd
Vu vveli stroku:
aaaaaaaaaaaaaaa ddddddd
Novaya stroka: Proverka kopirovaniya_
```

Функция **ctrncpy()** отличается от функции **ctrcpy()** тем, что включает еще один параметр. Он указывает количество символов, которые необходимо копировать из строки **S2** в строку **S1**. Функция имеет вид:

strncpy(S1, S2, n);

где **n** – количество символов (целое без знака).

Если длина **S1** меньше длины **S2**, то происходит урезание символов.

Задание 8.5. Исследовать использование функции **strncpy()**.

```
#include <string.h>
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[]="0123456789";              //Ввод символьной переменной  A
char B[]="qwertyuiop";              //Ввод символьной переменной  B
cout<<"S2= "<<A<<endl;              //Вывод символьной переменной  A
cout<<"S1= "<<B<<endl;              //Вывод символьной переменной  B
strncpy(B,A,4);
cout<<"S2new= "<<B<<endl;           //Вывод новой символьной переменной  B
getch();
return 0;
}
```

Вид экрана после работы программы:

```
S1= 0123456789
S2= qwertyuiop
S1new= 0123456789
S2new= 0123tyuiop
```

То есть, из строки **A** в строку **B** будут скопированы 4 первых символа и размещены в начале строки **B**.

2.3. Присоединение строк

Присоединение (**конкатенация**) строк используется для образования новой строки символов из двух и более исходных строк. Для этой цели используются функции

strcat(S1, S2) и strncat(S1, S2, n);

Функция **strcat(S1, S2)** присоединяет строку **S2** к строке **S1** и помещает ее в массив, где находилась строка **S1**. Строка **S2** не изменяется. Вновь полученная строка S1 автоматически завершается нулевым байтом.

Задание 8.6. Исследовать использование функции **strcat()**:

```
include <string.h>
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[30], B[30];
strcpy(A, "Hello, ");              //Копирование строки "Hello, " в строку A
strcpy(B, "World!");               //Копирование строки "World!" в строку B
cout<<"A= "<<A<<endl;
cout<<"B= "<<B<<endl;
strcat(A, B);                      //Присоединение строки B к строке A
cout<<endl<<"A= "<<A<<endl;
cout<<"B= "<<B<<endl;
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
A= Hello,
B= World!

A= Hello, World!
B= World!
```

Функция **strncat(S1, S2, n)** также осуществляет присоединение строк, однако присоединяет лишь указанное в третьем параметре количество символов, например:

```
char S1[80]="Dlya prodolgeniya ";
char S2[80]="nagat knopku ОК !";
strncat(S1,S2,7);
cout<<S1<<endl;
```

В результате на экран будет выведена строка:

Dlya prodolgeniya nagat knopku ОК !

Задание 8.7. Самостоятельно создайте проект с использованием функций для работы со строками: **getline()**, **strlen ()**, **strcpy()** и **ctrncpy()**, **strcat()** и **strncat()**. Все функции должны быть последовательно использованы в одной программе. В качестве символьных строк необходимо использовать свою фамилию и фамилию следующего по списку в журнале студента (в латинском написании).

При разработке проекта отладьте программу вначале для одной функции, затем добавьте код для операций с другой функцией и отладьте программу уже для двух функций и т. д. После операций с каждой функцией должен быть организован вывод исходных строк и результата (по аналогии с заданиями 8.1–8.6).

Контрольные вопросы

1. В программе на С++ определен массив **char A[11]**. Это означает, что строка содержит:
    1. 10 символов;
    2. 11 символов;
    3. 12 символов.
2. В языке С++ для копирования строк используется функция:
    1. **strlen( )**;
    2. **strcpy( )**;
    3. **strcat( )**.

3 В языке С++ конкатенация строк – это:
    1. Копирование одной строки в другой;
    2. Сравнение двух строк;
    3. Присоединение одной строки к другому.
4. Что такое символьная строка?
5. Что такое нулевой байт?
6. Как инициализируется символьный массив?
7. Как объявляется символьный массив?
8. Для чего применяется функция **getline()**?
9. Какие аргументы используются в функции **getline()**?
10. Для чего применяется функция **strcpy()**?
11. Какие аргументы используются в функции **strcpy()**?
12. Для чего применяется функция **strcat()**?
13. Какие аргументы используются в функции **strcat()**?
14. Для чего применяется функция **strncat()**?
15. Какие аргументы используются в функции **strncat()**?

Лабораторная работа № 9 (2 занятия)

АДРЕСАЦИЯ ПЕРЕМЕННЫХ И УКАЗАТЕЛИ В СРЕДЕ VISUAL C++ 2010

Цель работы**: изучение адресации памяти и исследование особенностей применения указателей в Visual C++ 2010.**

1. Понятие адреса переменной и указателя в Visual C++ 2010

Переменная занимает в памяти компьютера определенную область (набор ячеек). Расположение переменной в памяти, т. е. данный именованный набор ячеек, определяется адресом. При объявлении переменной для нее резервируется место в памяти. Размер зарезервированной памяти зависит от типа данной переменной.

Для доступа к содержимому выделенной памяти служит его **имя** (идентификатор). Для того чтобы узнать адрес конкретной переменной, применяется операция **взятия адреса**. Синтаксис операции следующий:

&ИмяПеременной,

т. е. перед именем переменной ставится знак **&.**

Задание 9.1. Исследовать программу, в которой используется операция взятия адреса для двух переменных **A** и **B**:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A=23.1;
double B=57.88;
cout<<"Znachenie A= "<<A<<endl;
cout<<"Adres A= "<<&A<<endl;        //Операция взятия адреса переменной
                                    //A и вывод адреса  &A
cout<<"Znachenie B= "<<B<<endl;
cout<<"Adres B= "<<&B<<endl;        //Операция взятия адреса перемен-
                                    //ной B и вывод адреса  &B
getch();
return 0;
}
```

После отладки и выполнения программы получим следующий результат:

```
Znachenie A= 23.1
Adres A= 002EF800
Znachenie B= 57.88
Adres B= 002EF7F0
```

!!! При исследовании программ по этой теме следует учитывать, что адреса переменных для каждого конкретного компьютера будут отличаться от приведенных в методическом пособии.

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр9**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 9.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr2-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr>**, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр9**. Над вставленным рисунком проставьте номер задания – 2**-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

В языке C++ есть возможность осуществлять непосредственный доступ к памяти компьютера. Для этого предусмотрен специальный тип переменных – указатели.

Указатель – это переменная, содержащая адрес некоторого объекта. Объектом может быть переменная или функция. В общем случае указатель – это целое число.

На рис. 9.1 показана взаимосвязь между адресом переменной и указателем на этот адрес.

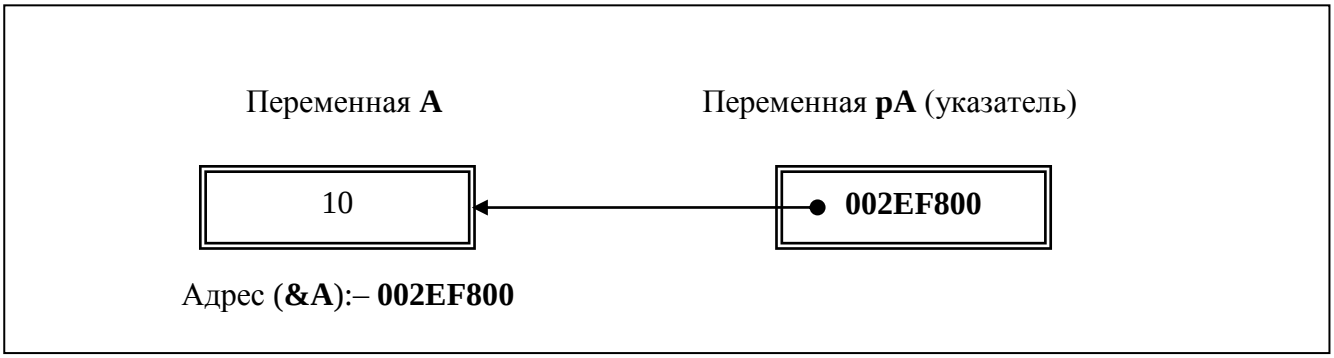


Рис. 9.1. Взаимосвязь между адресом переменной и указателем на этот адрес.

Такая взаимосвязь отображается следующим образом:

pA = &A;

Приведенная инструкция означает, что переменной **pA** присваивается адрес ячейки со значением **A** (в нашем случае **002EF800**).

Операндом оператора **&** не могут быть ни выражение, ни константа, ни регистровая переменная. Унарный оператор **&** называется **оператором адресации.**

Имена указателям даются в соответствии с правилами, приятыми в языке программирования **C** для обычных переменных.

Если переменная будет указателем, то она должна быть объявлена в программе. Указатель в программе объявляется следующим образом:

ТипОбъекта *Идентификатор;

Здесь **ТипОбъекта** определяет тип данных, на которые ссылается указатель с именем **Идентификатор**. Символ * (звездочка) означает, что следующая за ней переменная является указателем. При объявлении указателя под него резервируется 4 байта.

Примеры объявления указателей:

```
Char *A
int *temp, i, *z;
double f, *ptr;
```

Здесь объявлены указатели **A**, **temp**, **z**, **ptr** и переменные **i** и **f** .

Поскольку указатель является ссылкой на некоторую область памяти, ему может быть присвоен только значение **адреса переменной**, а не значение самой переменной.

Рассмотрим пример объявления и инициализации указателя.

Задание 9.2. Исследовать программу, в которой объявляются и инициализируются указатели **pA** и **pB** с присваиванием им значений адресов двух переменных **A** и **B**.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A=57.97;
double *pA=&A;              //Инициализация указателя pA и
//присваивание ему адреса A
double B=340;
double *pB;                 //Объявление указателя pB
pB =&B;                     //Присваивание указателю pB значения
                            //адреса B
cout<<"A="<<A<<"   &A= "<<&A<<endl;//Вывод переменной A и ее адреса A
cout<<"    pA = "<<pA<<endl;    //Вывод указателя pA
cout<<"B= "<<B<<"   &B= "<<&B<<endl; //Вывод переменной B и адреса
cout<<"   pB = "<< pB <<endl;          //Вывод указателя pB
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
A= 57.97    &A= 002CF7B4
              pA  = 002CF7B4
B= 340    &B= 002CF798
              pB  = 002CF798
```

Очевидно, что значение указателя совпадает со значением адреса соответствующей переменной.

2. Разыменование (разадресация) указателей

Указатели помогают осуществлять непосредственный доступ к памяти. Для того чтобы получить (прочитать) значение, записанное по адресу, который находится в указателе, используют операцию непрямого обращения или **разыменования** (*). Для этого используется имя указателя со звездочкой перед ним.

Тогда для ранее использовавшихся обозначений будет справедливо записать

A=*pA.

Задание 9.3. Исследовать программу, в которой разыменуются указатели **pA** и **pB**, т. е. определяются значения переменных **A** и **B** по значениям указателей на эти переменные.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A=57.97;
double *pA=&A;                  //Инициализация указателя uA и
                        //присваивание ему адреса A
double B=340;
double *pB;                     //Объявление указателя pB
pB =&B;                         //Присваивание указателю pB значения
                        //адреса B
cout<<"A= "<<A<<"   &A= "<<&A<<endl;
cout<<"  pA = "<< pA <<endl;        //Вывод указателя pA
cout<<"*pA =  "<<*pA <<endl;       //Разыменование  указателя pA и вывод
                        //результата
cout<<"B= "<<B<<"   &B= "<<&B<<endl;
cout<<"   pB = "<< pB <<endl;       //Вывод указателя pB
cout<<"*pB = "<<*pB <<endl;        //Разыменование  указателя pB и вывод
                        //результата
getch();
return 0;
}
```

После выполнения программы внимательно изучите на экране следующую информацию:

```
A= 57.97    &A= 0016FE90
              pA  = 0016FE90
* pA  =   57.97
B= 340    &B= 0016FE74
              pB  = 0016FE74
* pB  = 340
```

Язык программирования C++ позволяет работать с указателями так же, как и с переменными стандартных типов. Однако операции над указателями отличаются некоторыми особенностями.

Задание 9.4. Разработать программу определения адресов целых чисел от 0 до 9 и строчных букв латинского алфавита.

Программный код решения задания следующий:

```
#include <stdio.h>
#include <conio.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j = 0;
  char c='a', *psymbol;
// Адрес символа 'a'
psymbol=&c;
printf("\n\t Figures, symbols and their addresses:\n");
```

```
    for (i = 0; i < 10; ++i)
        printf("\n\t %3d) %2d --> %5p", i + 1, i, &i);
printf("\n");
for (*psymbol='a'; *psymbol<='z'; (*psymbol)++)
        printf("\n\t %3d) %2c --> %5p", ++j, *psymbol, psymbol);
printf("\n\n Press any key: ");
getch();
return 0;
}
```

В программе использован спецификатор формата **%5p** для определения адреса переменных. Число **5** определяет отступ от левого края на пять позиций.

После выполнения программы получим следующий результат:

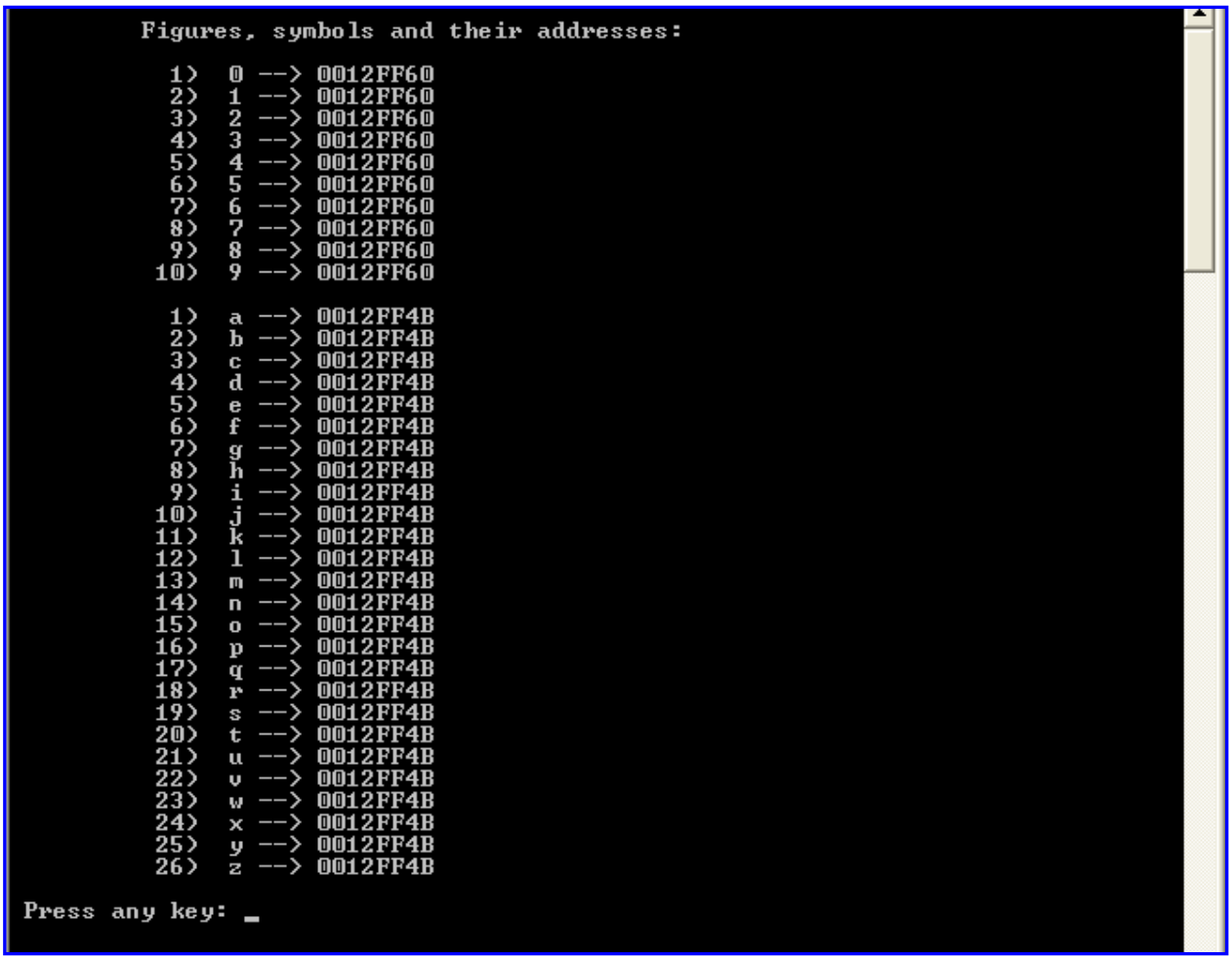


Рис. 9.2. Вывод адресов цифр и строчных букв

Задание 9.5. Внесите в программный код для Задания 2.4 такие изменения:

добавьте определение адресов прописных букв латинского алфавита и выведите их дополнительным столбцом к адресам строчных букв.

3. Операция присваивания указателей

Указатели одного и того же типа могут использоваться в операциях присваивания, как и другие любые переменные.

Для указателей одного типа можно, например, выполнять присваивание без разыменования, поскольку указатели сами по себе являются переменными.

Пусть определен еще один указатель типа **int**, например **p2A**. Тогда возможно произвести присвоение:

pA = p2A;

После этого указатель **p2A** будет указывать на ту же переменную **A**, что и указатель **pA**.

Задание 9.6. Исследовать программу, в которой применяется операциях присваивания указателей **px** и **g**, а **g** затем разыменуется для проверки - определяется, совпадает ли значение ***g** со значением переменной **x**:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x=10;
int *px, *g;
px=&x;
g=px;
cout<<"px= "<<px<<endl;
cout<<"g= "<<g<<endl;
cout<<"x= "<<x<<endl <<  " *g= "<<*g<<endl;
getch();
return 0;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
px= 002FFB74
g= 002FFB74
x= 10
 *g= 10
```

Очевидно, что операция присваивания между указателями и последующее разыменование указателя **g** не внесли погрешностей и значение разыменованного указателя **g** совпадает со значением переменной **x**.

4. Операции с указателями

В языке **C** допустимы следующие основные операции над указателями:

1. Присваивание; получение значения того объекта, на который он указывает (синонимы: косвенная адресация, разыменование, раскрытие ссылки) (см. выше);
2. Получение адреса самого указателя;
3. Унарные операции изменения значения указателя;
4. Аддитивные операции;
5. Операции сравнений (отношений).

Унарные операции **++** и **--** позволяют позиционировать указатель на следующую и предыдущую ячейки памяти, в которых хранятся значения типов, связанных с типом указателя. При этом значение указателя меняется на величину, определяемую размером соответствующего типа. Например, для указателя типа **char*** операция **++** увеличит значение адреса на **sizeof**

(char), для указателя типа **int*** операция **--** уменьшит значение адреса на **sizeof(int)** и т. д. Это свойство унарных операций **++** и **--** используется для последовательного обращения к значениям одного типа, связанного с типом указателя, хранящимся в смежных ячейках памяти. В таком смысле унарные операции **++** и **--** сходны с операциями увеличения и уменьшения счетчика цикла при последовательном обращении к элементам массива.

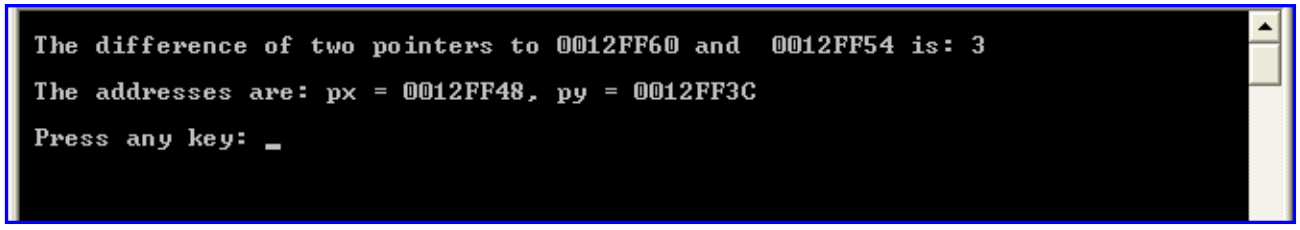
Задание 9.7. Разработать программу однозначного задания типа разностей указателей и определения адресов заданных указателей.

При решении данного примера подключим заголовок **stddef.h** для определения типа разности указателей с помощью зарезервированного имени типа **ptrdiff_t**.

Программный код решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x, y;
int *px, *py;
ptrdiff_t z;
// Взятие адресов переменных
px = &x;
py = &y;
// Разница двух указателей
z = px - py;
printf("\n The difference of two pointers to %p and  %p is: %d",px,py,(int) z);
printf("\n\n The addresses are: px = %p, py = %p\n", &px, &py);
printf("\n Press any key: ");
getch();
return 0;
}
```

После выполнения программы на экран будет выведена следующая информация:



Задание 9.8. В программном коде для задания 9.7:

1. Поменяйте местами переменные **x** и **y**. Проанализируйте результат выполнения программы.
2. Для переменных произведите инициализацию в соответствии с номером компьютера, на котором выполняется лабораторная работа, и текущего дня недели.
3. Рассмотрите решение примера для следующих типов: **char**, **long int**, **unsigned int, float**, **double**, **long double**.
4. Вывод результатов осуществите с помощью одной функции **printf()**.

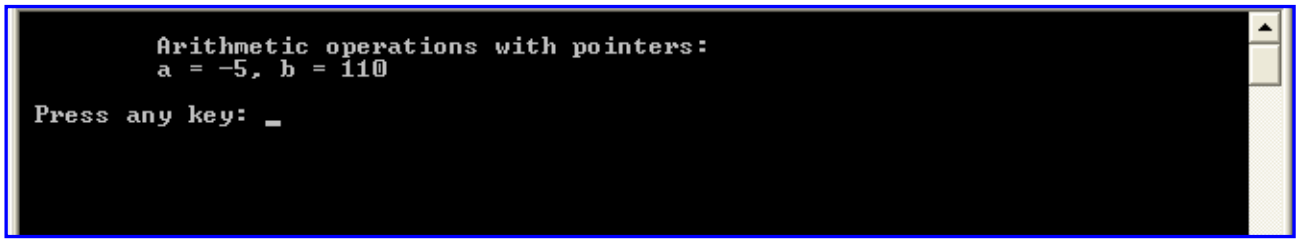
Задание 9.9. Разработать программу арифметических операций с указателями.

При выполнении примера следует иметь в виду, что операции **&** и ***** имеют более высокий приоритет, чем обычные арифметические операции.

Программный код решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x = 2, y = 7, a, b, *ptr, *ptr2;
ptr = &a;
ptr2 = &b;
*ptr = x - y;
*ptr2 = y - x - *ptr + 100;
printf("\n\t Arithmetic operations with pointers:\n");
printf("\t a = %d, b = %d\n", a, b);
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:



Следует обратить внимание на то, что переменные **a** и **b** сначала не были определены, а в результате приобрели некоторые значения.

Задание 9.10. В программу для задания 9.9 внесите следующие изменения:

1. Примените типы данных **double** и **float**.
2. Напишите программу для выполнения операций вычитания, умножения и деления с применением указателей.

Задание 9.11. Разработать программу двухуровневой адресации для объектов целого типа.

Случай, когда указатель содержит адрес другого указателя, называется многоуровневой адресацией. При двухуровневой адресации первый указатель содержит адрес второго указателя, в котором находится адрес объекта с нужным значением. Объявление указателя на указатель делается с помощью **двух звездочек** перед именем переменной.

Программный код решения задачи имеет следующий вид:

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

```
int _tmain(int argc, _TCHAR* argv[])
{
int x, y = 8;
int *ptr, **ptr2;
x = 7;
ptr = &x;
ptr2 = &ptr;
**ptr2 = *ptr + 10;
printf("\n\t The value of x = %d. 1 st pointer is: %d. 2 nd pointer is: %d\n", x,      *ptr,
**ptr2);
    ptr = &y;
    ptr2 = &ptr;
    **ptr2 = 88;
    printf("\n\t The value of y = %d\n", y);
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:

Задание 9.12. Доработайте программу для задания 9.11:

1. Выведите на экран пользователя адреса указателей.
2. Организуйте цикл инкрементирования первого указателя, начиная с X до 10X, где X – номер компьютера, на котором выполняется лабораторная работа. Сделайте вывод значений переменной, на которую дает ссылку первый указатель, и значений второго указателя.
3. Напишите программу трехуровневой адресации при задании целых чисел, равных X и 10X, где X – номер компьютера, на котором выполняется лабораторная работа.

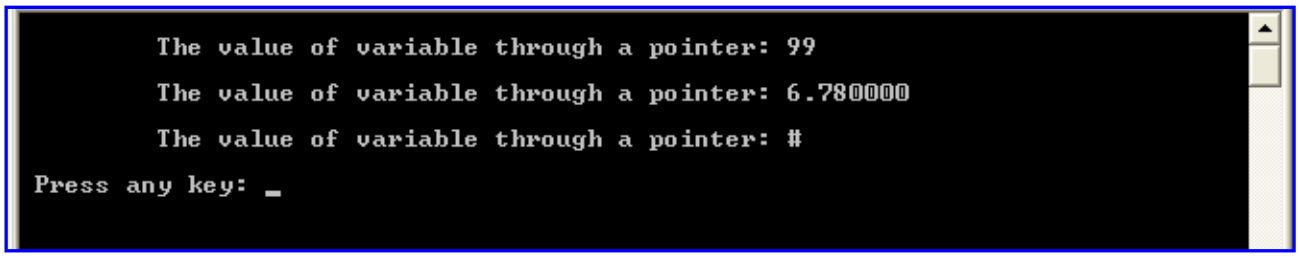
Задание 9.13. Разработать программу для определения и инициализации переменных разных типов и одного указателя типа **void.** Последовательно присваивая указателю адреса переменных, вывести значения переменных с помощью разыменования указателя.

Программный код для решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
```

```
int x = 99;
double y = 6.78;
char symbol = '#';
void *ptr;
ptr = &x;
printf("\n\t The value of variable through a pointer: %d\n", *(int *) ptr);
ptr = &y;
printf("\n\t The value of variable through a pointer: %lf\n", *(double *) ptr);
ptr = &symbol;
printf("\n\t The value of variable through a pointer: %c\n", *(char *) ptr);
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:



Особенностью использования указателя типа **void** является то, что при его разыменовании необходимо осуществлять преобразования типов. Прежде чем выполнить разыменование указателя, его приводят к указателю соответствующего типа.

Задание 9.14. Разработайте программный код на базе предыдущего, где:

1. Добавьте переменные типа **float**, **unsigned**, **long** и обеспечьте ввод их значений с клавиатуры. Выведите адреса и значения переменных с помощью разыменования указателя.
2. Задайте порядок (нумерованную последовательность) инициализации переменных и создайте вывод значений указателя на основе переключателя **switch**. Номер инициализируемой переменной задайте с клавиатуры.
3. Введите операцию двухуровневой адресации с применением указателя типа **void**. Выведите значения двух указателей с помощью их разыменования.

Задание 9.15. Разработайте программу для реализации следующего условия:

1. Определить и инициализировать переменную типа **double**.

2. Определить указатели типа **char ***, **int ***, **double ***, **void *** и инициализировать их адресом переменной.

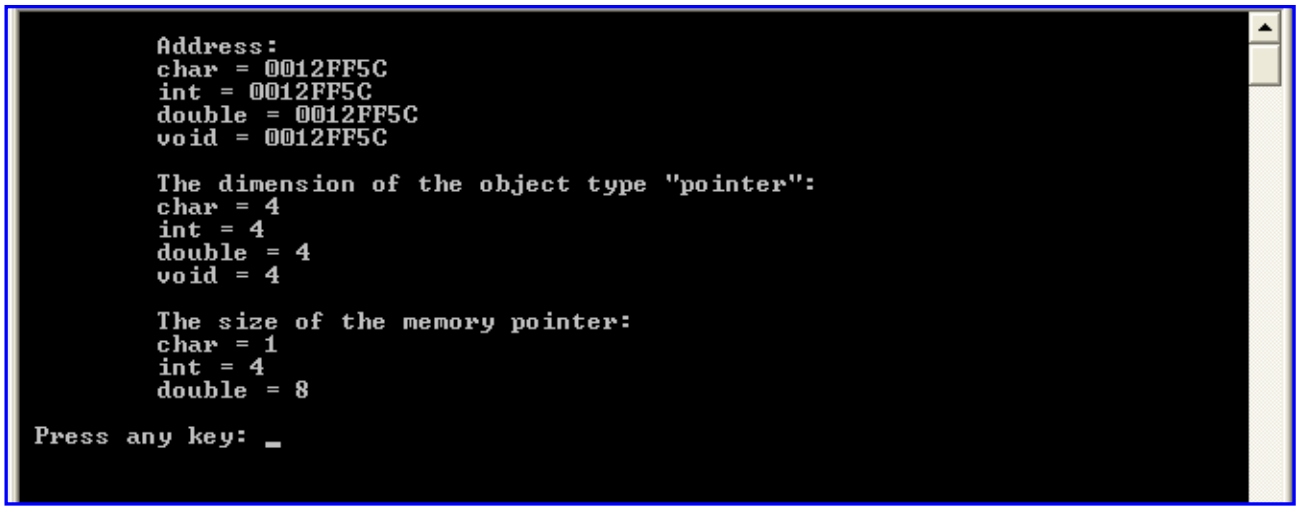
3. Вывести на экран пользователя значения указателей, их размеры и длины участков памяти, которые связаны с выражениями, разыменовывающими указатели.

Программный код для решения задачи следующий:

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

```
int _tmain(int argc, _TCHAR* argv[])
{
double d = 6.78, *dp;
char *cp;
int *ip;
void *vp;
    //Адресация с приведением типов
cp = (char *)&d;
ip = (int *)&d;
dp = (double *)&d;
vp = &d;
printf("\n\t Address:\n\t char = %p\n\t int = %p\n\t double = %p\n\t void = %p\n", cp, ip,
dp, vp);
    //Размеры указателей и памяти разыменованных указателей:
printf("\n\t The dimension of the object type \"pointer\":\n\t char = %d\n\t int = %d\n\t
double = %d\n\t void = %d\n",
sizeof(cp), sizeof(ip), sizeof(dp), sizeof(vp));
printf("\n\t The size of the memory pointer:\n\t char = %d\n\t int = %d\n\t double =
%d\n",sizeof(*cp),sizeof(*ip),sizeof(*dp));
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:



Как видно из полученного результата, размеры участков памяти, выделенных указателям разных типов, одинаковы.

Задание 9.16. Разработайте программу на базе предыдущей:

1. В программу добавьте вывод размера памяти для разыменованного указателя типа **void**.
2. Выведите значения указателей заданных типов. Определите указатель с правильным доступом к значению переменной **d = 6.78**.
3. Объявление указателей и взятие адреса сделайте в одной строчке для соответствующего типа.
4. В программу добавьте строки по вводу целого, вещественного типов данных, а также одиночного символа. Затем с помощью указателей выведите на консоль значения введенных данных.

Контрольные вопросы

1. Каково общее назначение указателей в языке **C**?
2. Какие арифметические операции допускаются для указателей?
3. Какие унарные операторы используются с указателями? Как они называются?
4. Для каких типов данных может быть использован указатель?
5. Как числовые значения указателей изменяются при их инкрементировании в зависимости от типов данных.
6. С помощью какого формата осуществляется вывод на консоль адресов переменных заданного типа?
7. Что такое многоуровневая адресация? Как она организуется в языке **C**?
8. Как осуществляется инициализация указателей на вещественные типы данных?
9. Как осуществляется инициализация указателей на символьный тип данных?
10. Какой смысл имеет значение указателя **NULL**?
11. Что произойдет, если применить к указателю со значением **NULL** операцию разыменования?
12. Как следует определять и инициализировать указатель на константу?
13. Как следует определять и инициализировать константный указатель?
14. В чем отличие константного указателя от указателя на константу?
15. Что будет выведено на экран после выполнения программы

int A=300;

cout<<&A;?

    1. Значение **A**, то есть 300;
    2. Адреса ячеек, в которых записано значение **A;**
    3. Сообщение об ошибке.

16. Укажите правильное объявление указателя в C++.
    1. ***double pA;**
    2. **double *pA;**
    3. **double pA*;**
17. Возможна ли следующая инициализация указателя:

char A=”yes”; char *pA=&A;?

    1. Возможна;
    2. Невозможна;
    3. Такой конструкции в C++ нет.
18. Что означает оператор **double *pA=&A;**?
    1. Инициализация переменной **A;**
    2. Объявление указателя **pA**;
    3. Инициализация указателя **pA** и присваивание ему адреса переменной **A**.
19. Что такое указатель в языке C++?
    1. Адрес некоторой переменной или функции;
    2. Переменная, содержащая адрес некоторой переменной;
    3. Стандартная функция.
20. Как объявляется указатель с именем **pA** в языке C++?
    1. **int *pA**;
    2. **int «pA**;
    3. **int &A**.
21. Как обозначается операция взятия адреса переменной **A** в языке C++?
    1. ***pA**;
    2. **«pA**;
    3. **&A**.
22. Как обозначается операция разыменования указателя **pA** в языке C++?

1. ***pA**;
2. «**pA**;
3. **&pA**.

23. Что означает операция разыменования указателя в языке С++?
1. Получение адреса переменной в указателе;
2. Получение значения переменной, записанной по адресу, который находится в указателе;
3. Запись переменной по адресу, который находится в указателе.

24. Какое значение примет **Y** в программе

```
double  X=10.1;
double *pA;
pA=&X;
Y=*pA;
```

1. **Y**=10.1;
2. В переменную **Y** запишется адрес, по которому находится **X**;
3. Будет выдано сообщение об ошибке с указанием на последнюю строку.

25. Можно ли в языке С++ выполнять арифметические операции над указателями?
1. Можно;
2. Нельзя;
3. Только операцию присваивания.

26. Что будет выведено на экран дисплея после выполнения программы

```
int *pA;
   for (i=0; i<100; i++)
   cout<<*(pA+i)<<" ";
```

?
1. Значения элементов какого-то массива;
2. Значения указателей;
3. Такую конструкцию в языке С++ использовать нельзя.

27. Укажите на возможность такого объявления указателя **int **ppA;**
1. Возможно;
2. Невозможно;
3. Все зависит от содержания программы.

28. На сколько байтов изменится значение **pC** в программе

```
int  C[20];
int *pC=&C;
pC++;
```

?
1. Не изменится;
2. Увеличится на 4 байта**;**
3. Уменьшится на 4 байта

Лабораторная работа № 10

НАСТРОЙКА КОМПИЛЯТОРА ЯЗЫКА C++. ИССЛЕДОВАНИЕ ОПЕРАТОРОВ ВВОДА И ВЫВОДА

Рассматривается инструментальная среда разработки приложений Microsoft Visual Studio 2010 в режиме компилятора языка **C**. Приводится простейшая программа на языке **C**, которая иллюстрирует использование средств элементарного текстового вывода на консоль.

1. Настройка компилятора языка C++. Исследование операторов ввода и вывода языка C

Язык **C** в основе своей был создан в 1972 г. как язык для операционной системы UNIX [1]. Его автором считается Денис М. Ритчи (Dennis M. Ritchie).

Популярность языка обусловлена, прежде всего, тем, что большинство операционных систем были написаны на нем. Однако сначала его распространение задержалось из-за того, что не было удачных компиляторов.

Некоторое время отсутствовала единая политика по стандартизации языка **C**. В начале 1980-х гг. в Американском национальном институте стандартов (ANSI) был сформирован комитет по стандартизации языка **C**. В 1989 г. работа комитета по языку **C** была ратифицирована, и в 1990 г. вышел в свет первый официальный документ по стандарту языка **C**. Появился стандарт 1989, т. е. **C89** [2]. К разработке стандарта по языку **C** была также привлечена Международная организация по стандартизации (ISO). Появился стандарт ISO/IEC 9899:1990, или ANSI **C99** [2].

В данном пособии за основу принимается стандарт языка **C** от 1989 г.

Язык **C** является языком высокого уровня, но в нем заложены возможности, которые позволяют программисту (пользователю) работать непосредственно с аппаратными средствами компьютера и общаться с ним на достаточно низком уровне [2]. Многие операции, выполняемые на языке **C**, сродни языку Ассемблера. Поэтому язык **C** часто называют языком среднего уровня.

Для написания программ в практических разделах данного учебного пособия будет использоваться компилятор языка **C++**, а программирование вестись в среде **Microsoft Visual Studio 2010**. Предполагается, что на компьютере установлена эта интегрированная среда, доступна в следующих вариантах:

1. **Express** – бесплатная среда разработки, включающая только базовый набор возможностей и библиотек;
2. **Professional** – поставка, ориентированная на профессиональное создание программного обеспечения и командную разработку, при которой созданием программы одновременно занимаются несколько человек;
3. **Premium** – издание, включающее дополнительные инструменты для работы с исходным кодом программ и создания баз данных;
4. **Ultimate** – наиболее полное издание Visual Studio, содержащее все доступные инструменты для написания, тестирования, отладки и анализа программ, а также дополнительные инструменты для работы с базами данных и проектирования архитектуры ПО.

Отличительной особенностью среды **Microsoft Visual Studio 2010** является то, что она поддерживает работу с несколькими языками программирования и программными платформами. Поэтому перед тем как писать программу на языке **C**, необходимо выполнить несколько подготовительных шагов по созданию проекта и выбора и настройки компилятора языка **C** для трансляции исходного кода.

Задание 10.1. Изучить настройку компилятора и проанализировать несложный вариант применения оператора **printf**.

Запустить среду **Microsoft Visual Studio 2010**. После запуска появится стартовая страница (рис. 10.1).

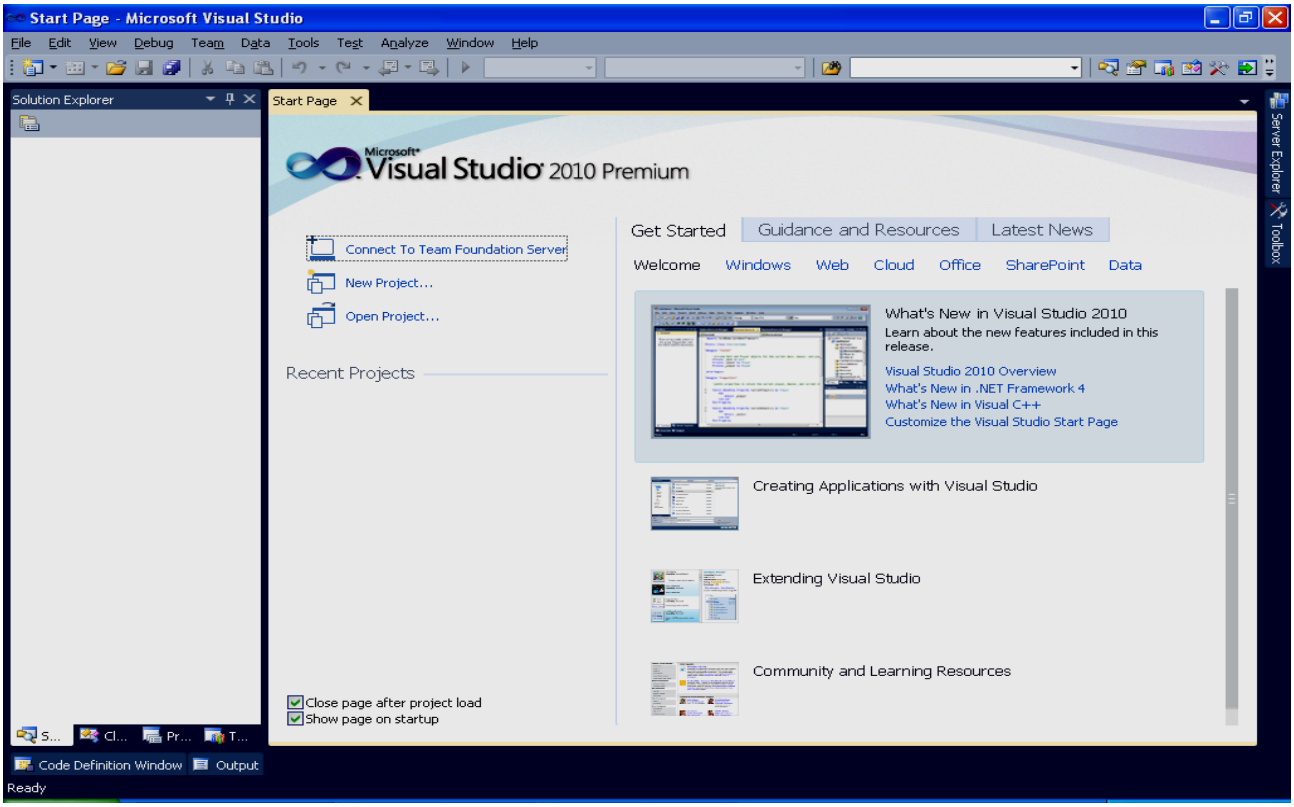


Рис. 10.1. Стартовая страница Visual Studio 2010

Следующим шагом является создание нового проекта. Для этого в меню **File** необходимо выбрать **New→Project** (или нажать комбинацию клавиш **Ctrl+Shift+N**). Результат выбора пунктов меню для создания нового проекта показан на рис. 10.2.

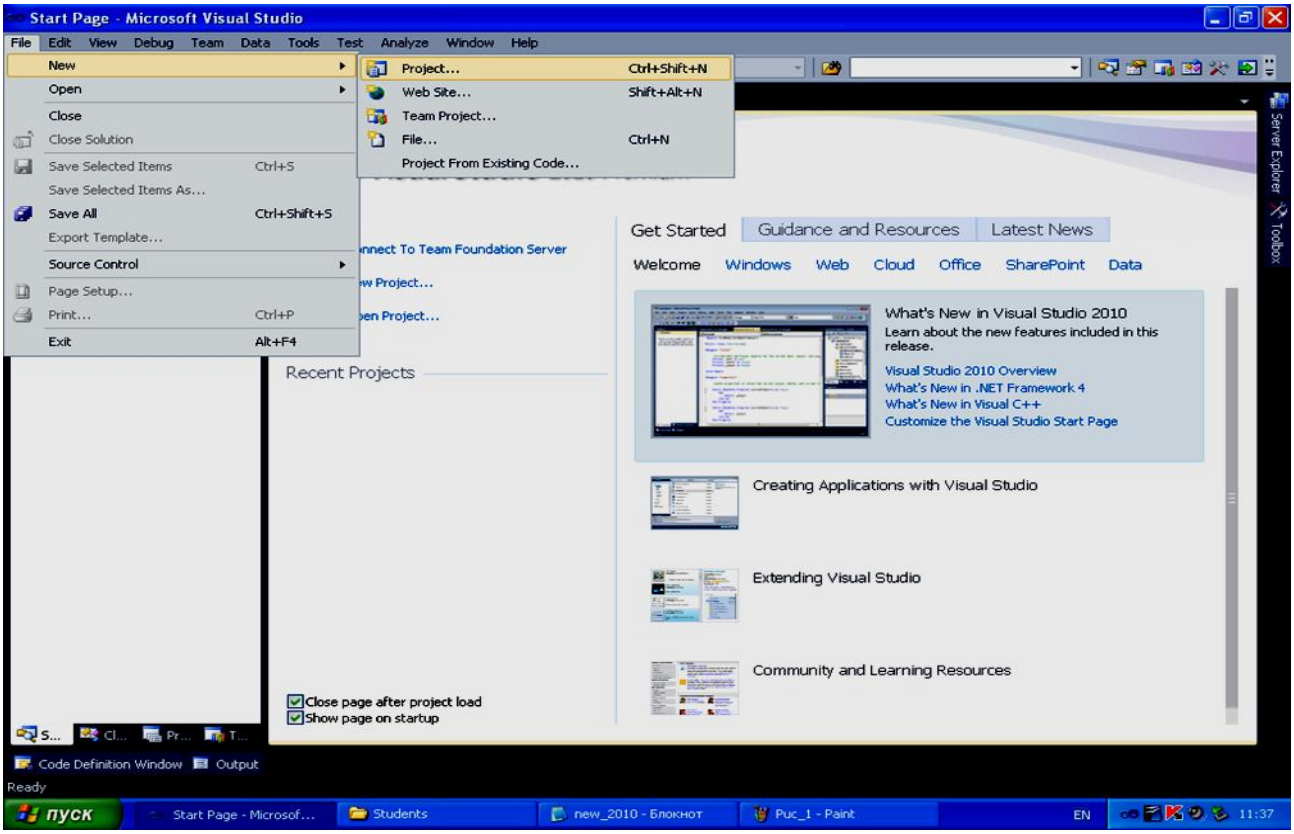


Рис. 10.2. Окно с выбором нового проекта

Среда Visual Studio отобразит окно **New Project**, в котором необходимо выбрать тип создаваемого проекта.

Проект (project) используется в Visual Studio для логической группировки нескольких файлов, содержащих исходный код, на одном из поддерживаемых языков программирования, а также любых вспомогательных файлов. Обычно после сборки проекта (которая включает компиляцию всех всходящих в проект файлов исходного кода) создается один исполняемый модуль.

В окне **New Project** следует развернуть узел **Visual C++**, затем обратиться к пункту **Win32** и на центральной панели выбрать **Win32 Console Application** (рис. 10.3).

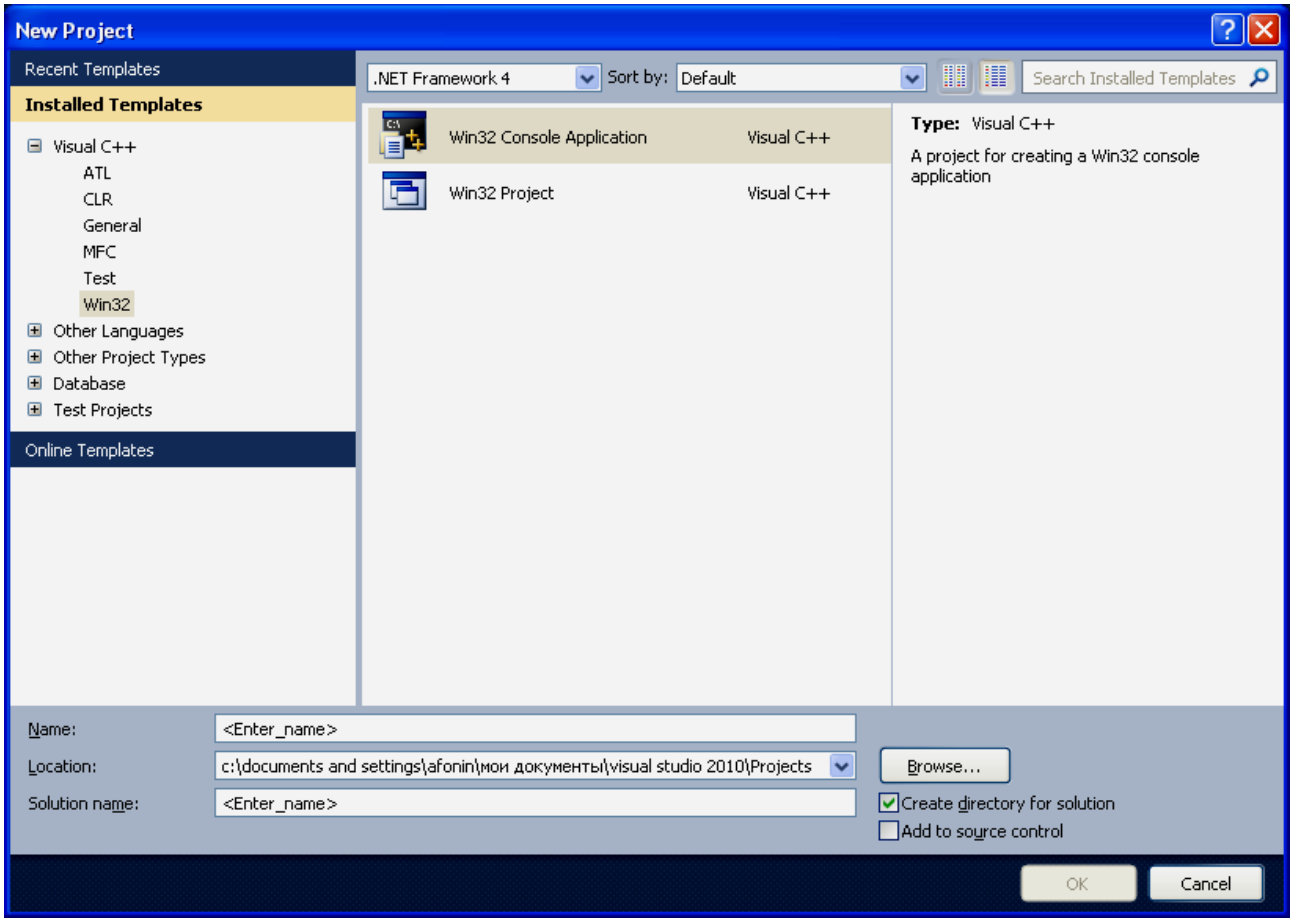


Рис. 10.3. Выбор типа проекта

После выбора типа проекта в поле редактора **Name** (где по умолчанию имеется <Enter_name>) необходимо ввести его имя, например **hello**. В поле **Location** можно указать путь размещения проекта или выбрать его (путь) с помощью клавиши (кнопки) **Browse**. По умолчанию проект сохраняется в специальной папке **Projects**.

Выбор имени проекта может быть достаточно произвольным: допустимо использовать числовое значение, допустимо имя задавать через буквы русского алфавита. В дальнейшем будем давать проекту имя, набранное с помощью букв латинского алфавита и цифр.

Пример выбора имени проекта показан на рис. 10.4.

Одновременно с проектом Visual Studio создает решение. **Решение** (solution) – это способ объединения нескольких проектов для организации более удобной работы с ними.

После нажатия кнопки **OK** откроется окно **Win32 Application Wizard** (мастер создания приложений для операционных систем Windows) (рис. 10.5).

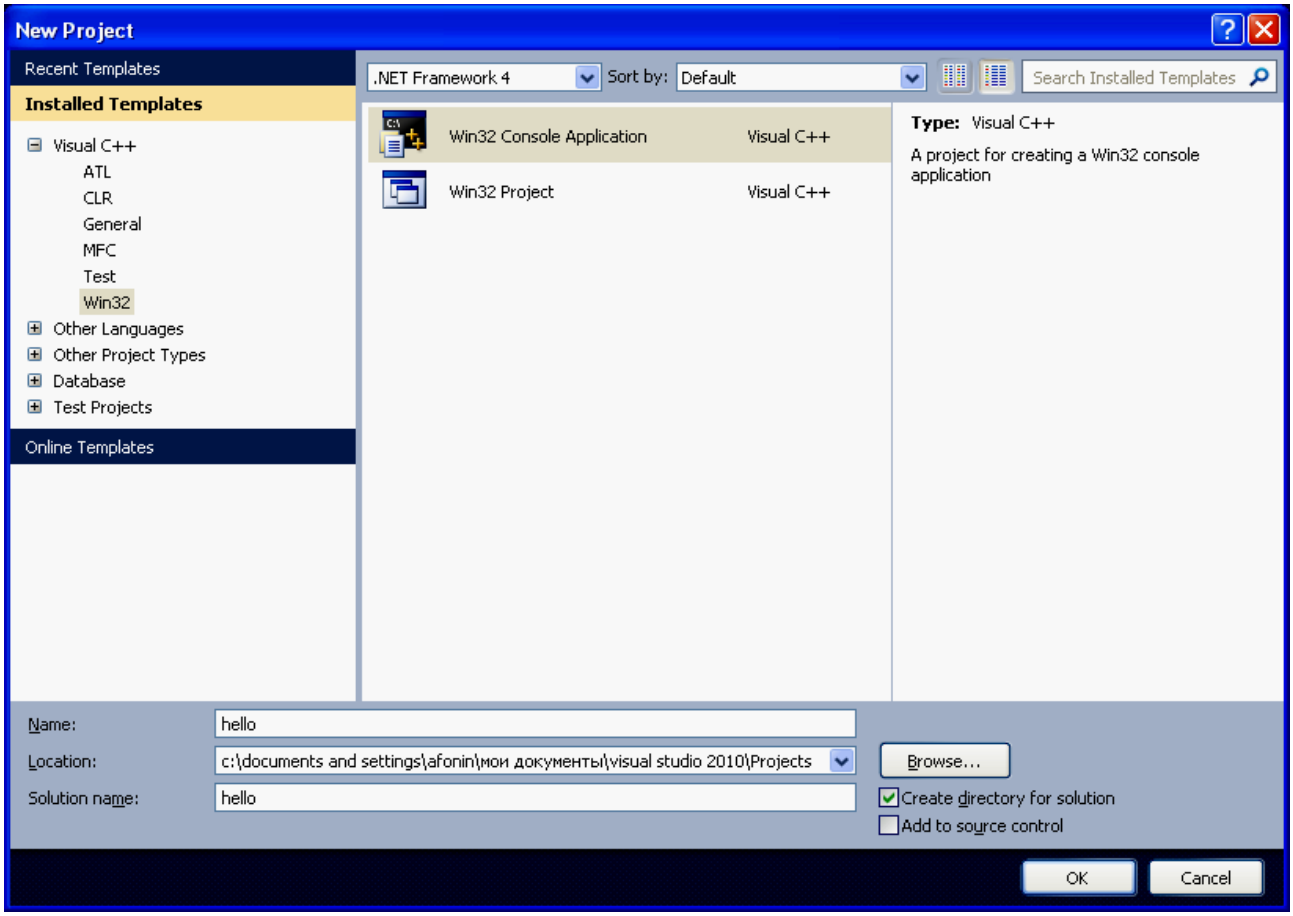


Рис. 10.4. Пример задания имени проекта

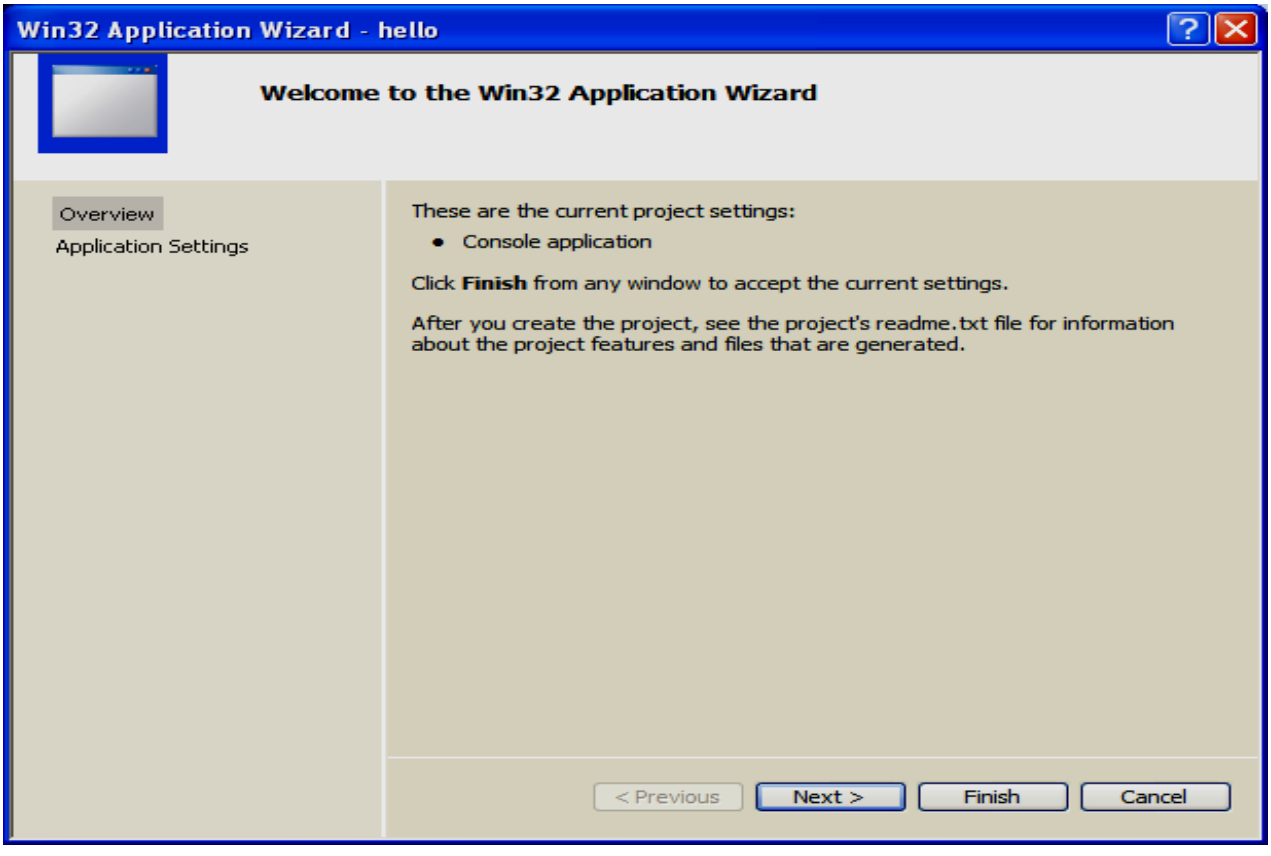


Рис. 10.5. Мастер создания приложения

На первой странице мастера представлена информация о создаваемом проекте, на второй можно сделать его первичные настройки. После обращения к странице **Application Settings** или нажатия кнопки **Next** получим окно, представленное на рис. 10.6.

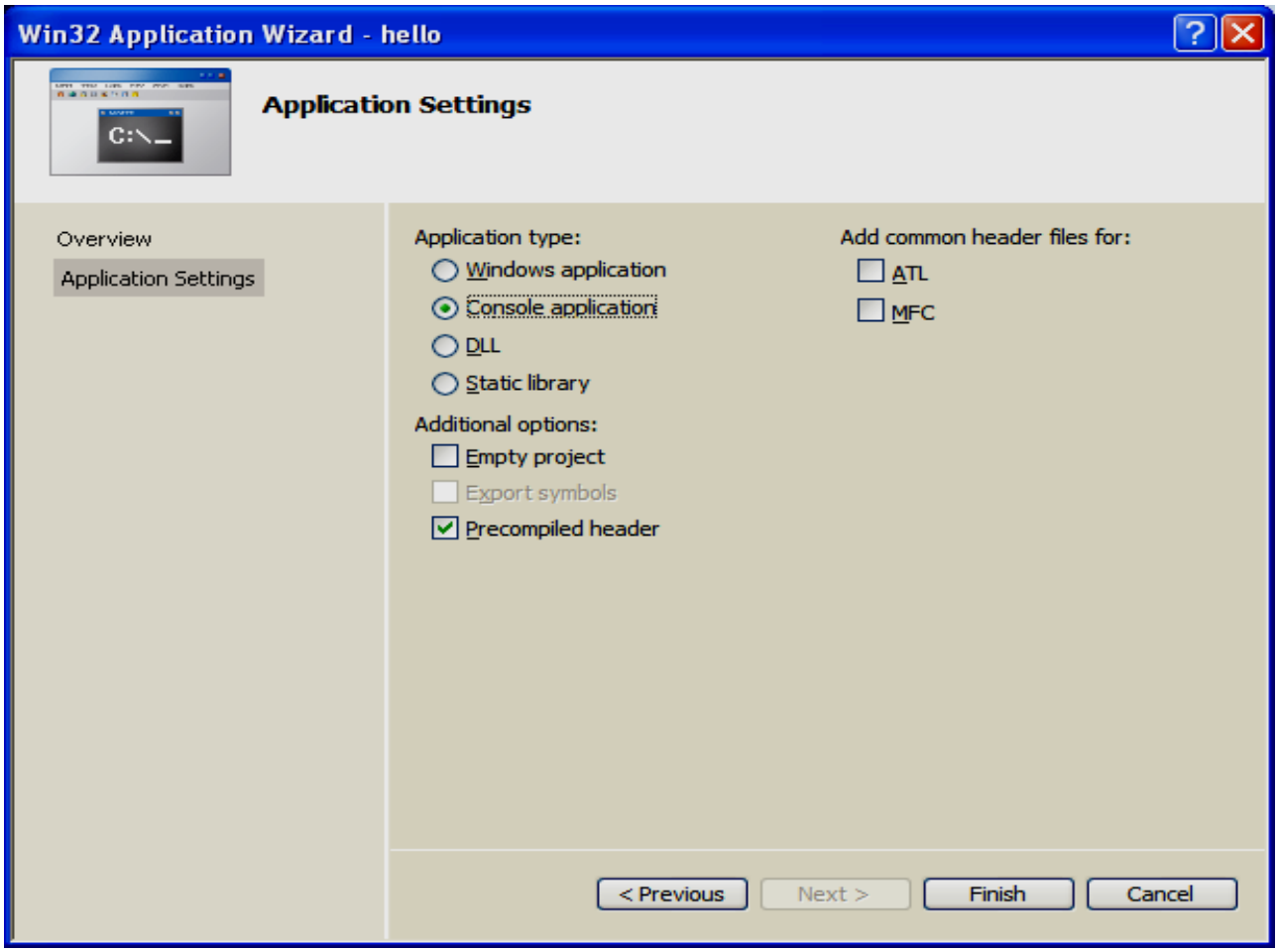


Рис. 10.6. Страница мастера настройки проекта по умолчанию

В дополнительных опциях (**Additional options**) следует поставить галочку в поле **Empty project** (пустой проект) и снять (убрать) ее в поле **Precompiled header** (рис. 10.7).

Здесь и далее будем создавать проекты по приведенной схеме, т. е. проекты в консольном приложении, которые выполняются целиком программистом (за счет выбора **Empty project**). После нажатия кнопки **Finish** получим экранную форму (рис. 10.8), где приведена последовательность действий добавления файла для создания исходного кода к проекту. Стандартный путь для этого:

1. Подвести курсор мыши к папке **Source Files** из узла **hello** в левой части открытого проекта приложения;
2. Нажать правую клавишу мышки (ПКМ);
3. В открывшемся контекстном меню выбрать **Add**, а затем **New Item** (новый элемент).

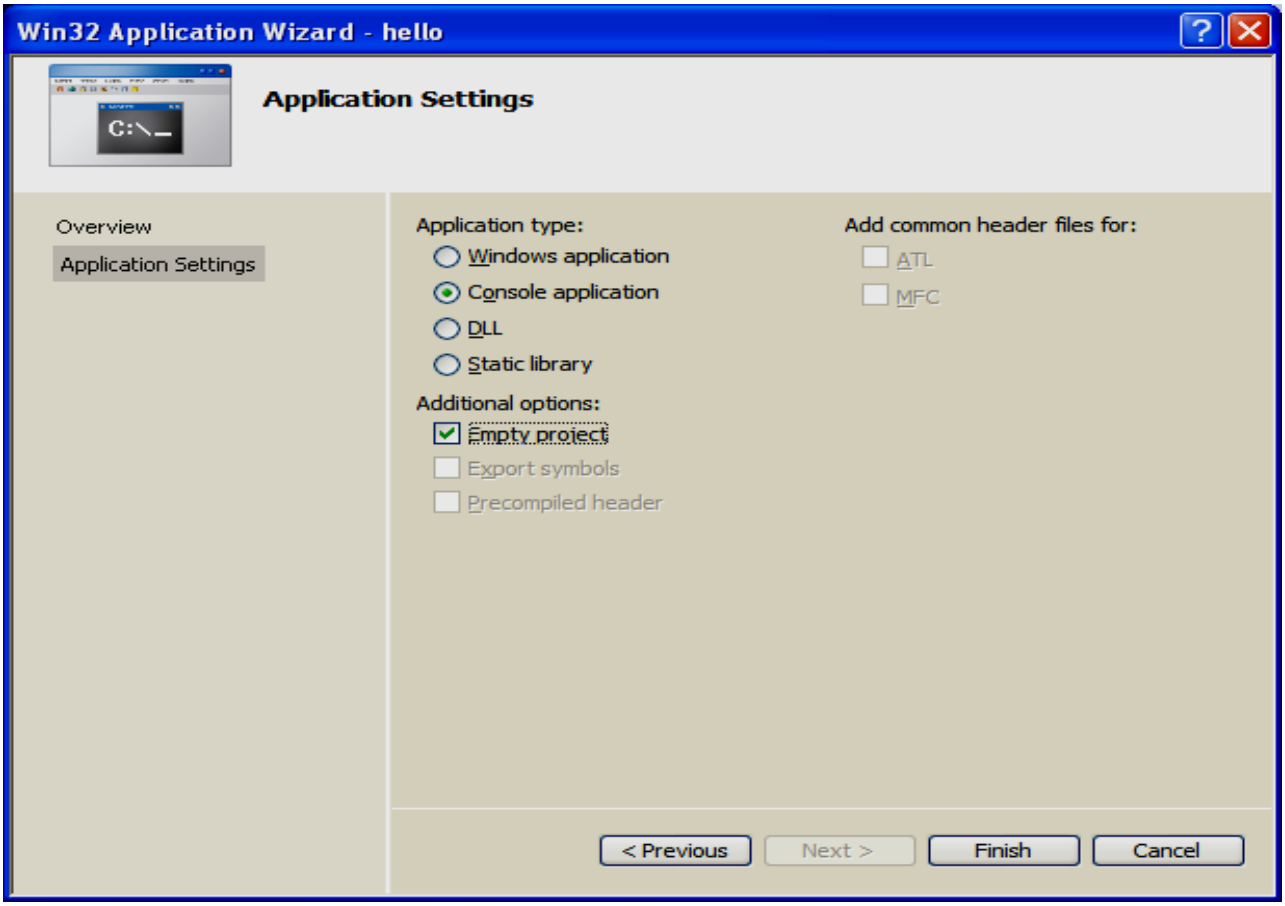


Рис. 10.7. Выполненная настройка мастера приложений

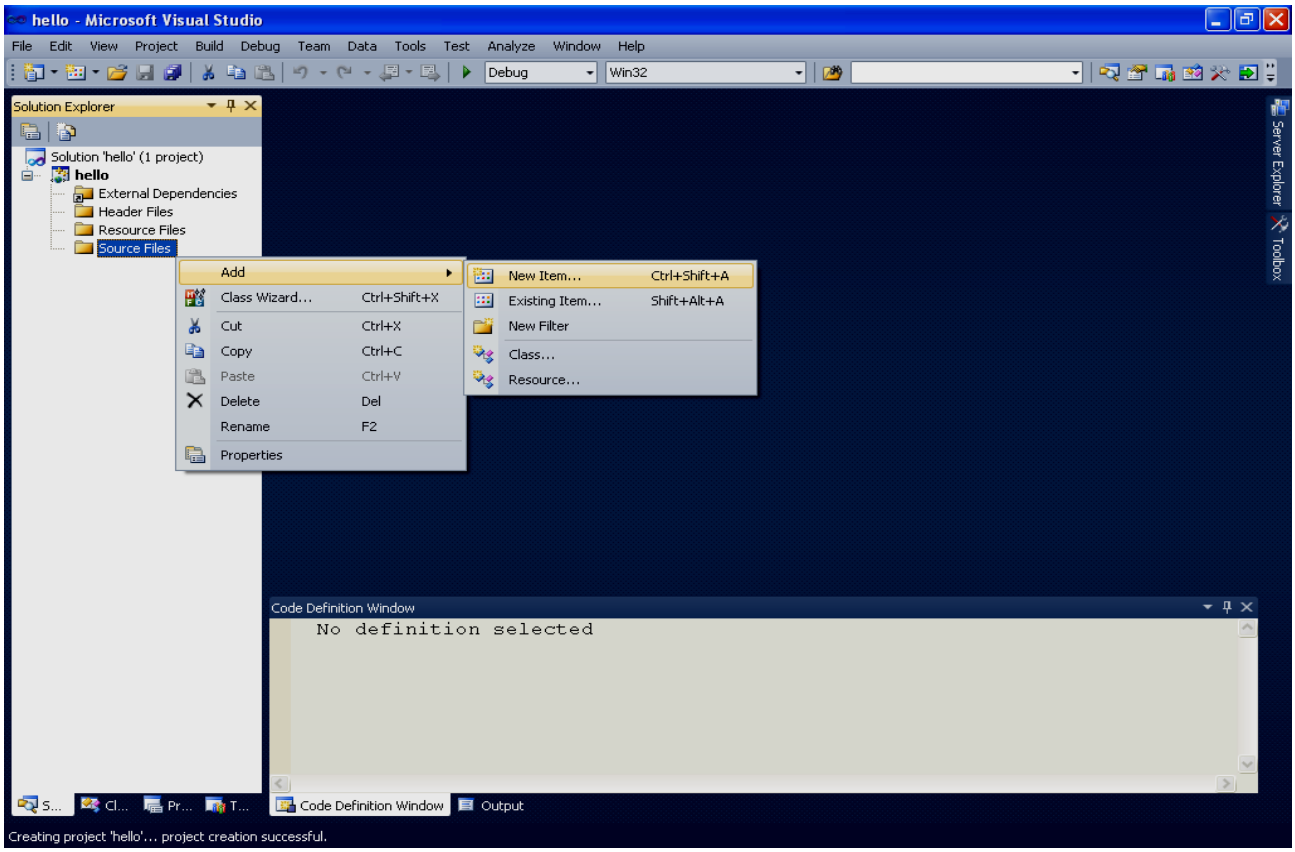


Рис. 10.8. Меню добавления нового элемента к проекту

Выбрав и нажав **New Item**, получим окно (рис. 10.9), где через пункт меню **Code** узла **Visual C++** выполнено обращение к центральной части панели, в которой осуществляется выбор типа файлов. В данном случае требуется обратиться к закладке **C++ File** (.cpp).

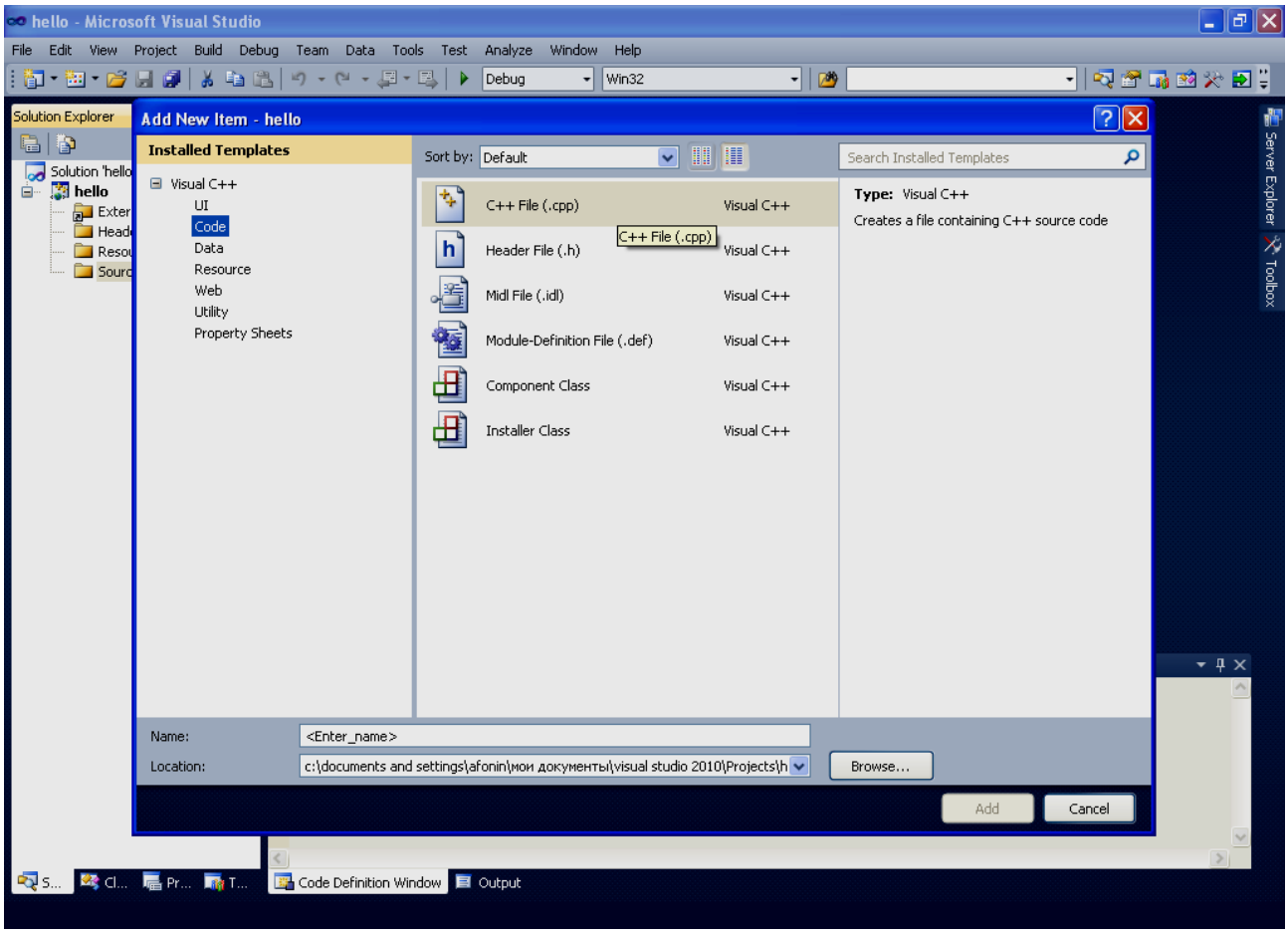


Рис. 10.9. Окно выбора типа файла для подключения к проекту

Теперь в поле редактора **Name** (в нижней части окна) следует задать имя нового файла и указать расширение **.c**. Например, **main.c**. Имя может быть достаточно произвольным, но имеется негласное соглашение, что оно должно отражать назначение файла и логически описывать исходный код, который в нем содержится. В проекте, состоящем из нескольких файлов, есть смысл выделить файл, содержащий главную функцию программы, т.е. ту, с которой она начнет выполняться. В данном пособии такому файлу мы будем задавать имя **main.c**, где расширение **.c** указывает на то, что этот файл содержит исходный код на языке **C**, и он будет транслироваться соответствующим компилятором. Программам на языке **C** принято давать указанное расширение. После задания имени файла в поле редактора **Name** получим форму, приведенную на рис. 10.10.

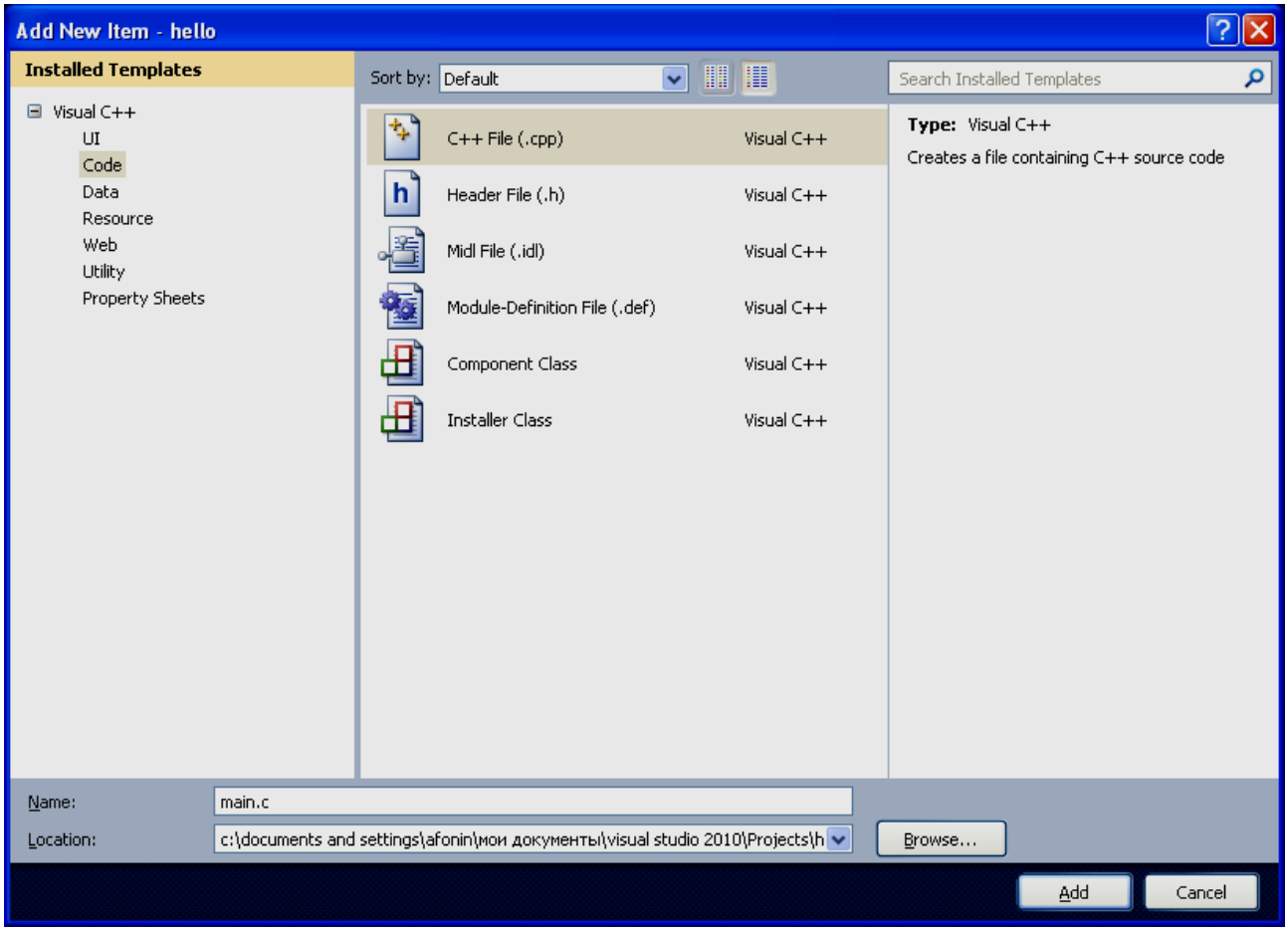


Рис. 10.10. Задание имени файла, подключаемого к проекту

Затем следует нажать кнопку **Add** (добавить). Добавленный файл отображается в дереве **Solution Explorer** под узлом Source Files (файлы с исходным кодом), и для него автоматически открывается редактор.

В левой панели в папке **Solution Explorer** отображаются файлы, включенные в проект в папках. Приведем их описание.

Папка **Source Files** (файлы исходного кода) предназначена для файлов с исходным кодом. В ней отображаются файлы с расширением **.c.**

Папка **Header Files** сожержит заголовочные файлы с расширением **.h.**

В папке **Resource Files** представлены файлы ресурсов, например изображения и т.д.

Папка **External Dependencies** отображает файлы, не добавленные явно в проект, но использующиеся в файлах исходного кода, например включенные при помощи директивы #include. Обычно в этой папке присутствуют заголовочные файлы стандартной библиотеки, применяющиеся в проекте.

Следующий шаг состоит в настройке проекта. Для этого в меню **Project** главного меню следует выбрать **hello Properties** (или одновременно нажать клавиши Alt+F7).

После того как откроется окно свойств проекта, следует обратиться (с левой стороны) к **Configuration Properties**; появится ниспадающий список (рис. 10.11). Далее нужно обратиться к узлу **General** (общие) и через него в левой панели выбрать **Character Set** (набор символов), где установить свойство **Use Multi-Byte Character Set**. Эта настройка позволяет задать кодировку символов – ANSI или UNICODE – для компиляции программы.

Для совместимости со стандартом C89 мы выбираем **Use Multi-Byte Character Set**. Это позволяет применять многие привычные функции, например, по выводу информации на консоль.

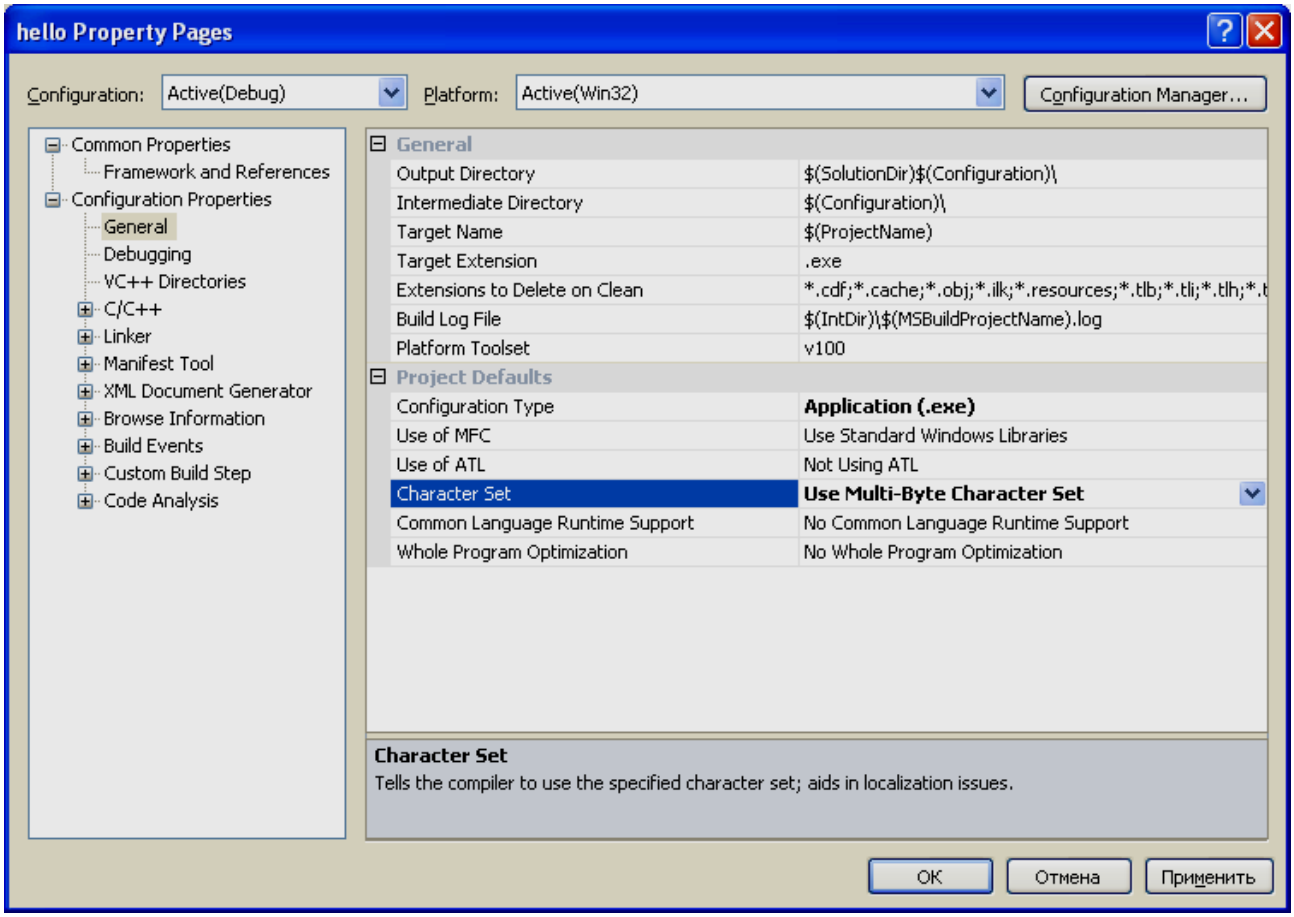


Рис. 10.11. Меню списка свойств проекта

После осуществления выбора следует нажать кнопку **Применить.**

Затем необходимо выбрать узел **C/C++** и в ниспадающем меню выбрать пункт **Code Generation** (создание кода), через который в правой части панели обратиться к закладке **Enable C++ Exceptions** (включить C++ исключения), установив для нее **No** (запрещение исключений **C++**).

Результат установки выбранного свойства представлен на рис. 10.12. Затем нужно нажать кнопку **Применить.**

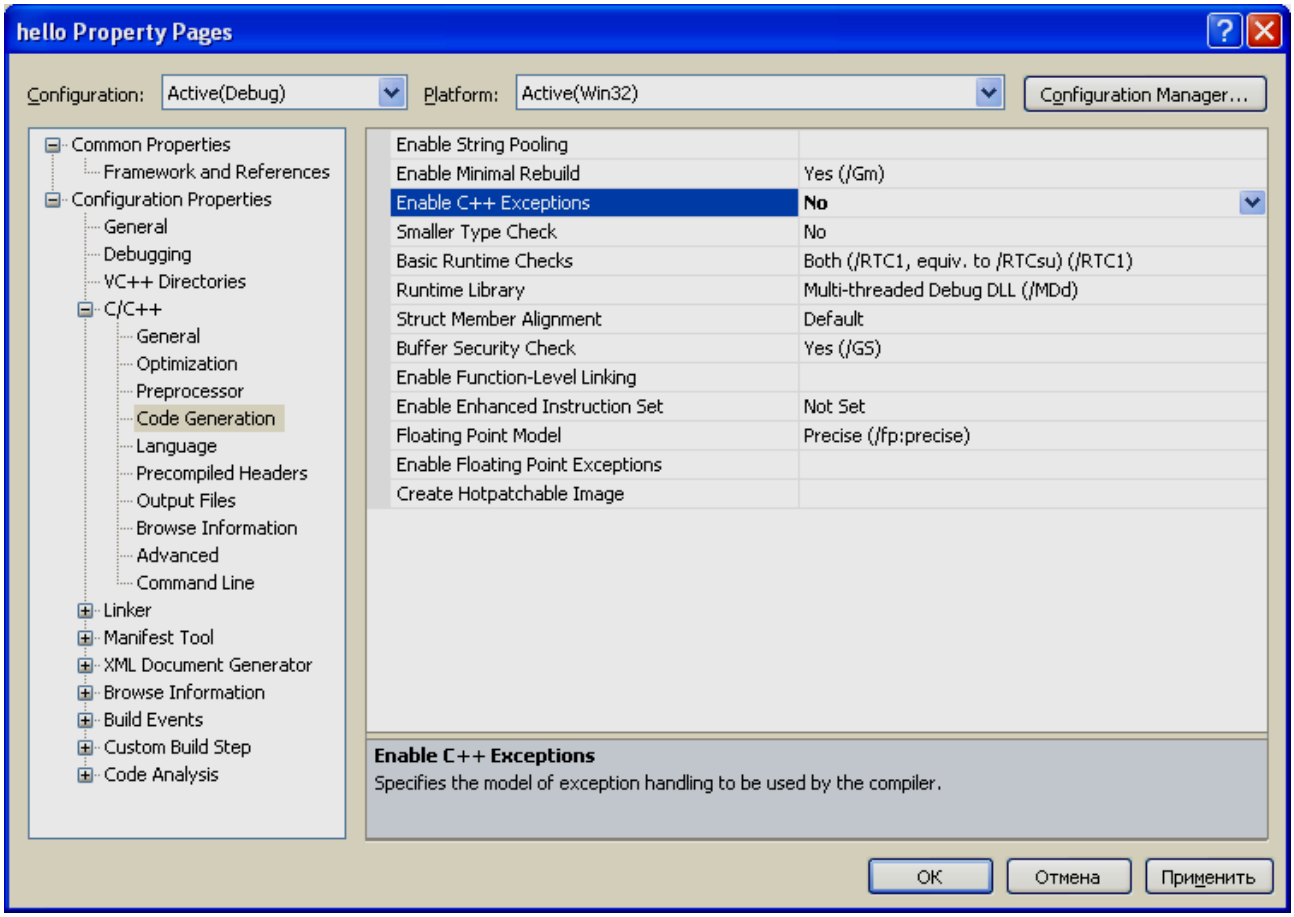


Рис. 10.12. Страница свойств для запрещения исключений **C++**

Далее в ниспадающем меню узла **C/C++** необходимо выбрать пункт **Language** и через него обратиться в правую часть панели, где установить следующие значения: для свойства **Disable Language Extensions** (дополнительные языковые расширения фирмы Microsoft) – **Yes (/Za)**, для **Treat Wchar_t As Built in Type** (рассматривать тип wchar_t как встроенный тип) – **No (/Zc:wchar_t–)**, для **Force Conformance in For Loop Scope** (соответствие стандарту определения локальных переменных в операторе цикла for) –**Yes(Zc:forScope)**, для **Enable Run-Time Type Information** (разрешить информацию о типах во время выполнения) – **No (/GR–)**, для свойства **Open MP Support** (разрешить расширение Open MP; используется при написании программ для многопроцессорных систем) – **No(/openmp–)**. Результат выполнения этих действий показан на рис. 10.13.

После выполнения указанных действий следует нажать клавишу **Применить**. Далее в ниспадающем списке узла **C/C++** следует выбрать пункт **Advanced** (дополнительно) и в правой панели изменить свойство **Compile As** в свойство компиляции языка **C**, т. е. **Compile as C Code (/TC)**. Результат установки компилятора языка **C** представлен на рис. 10.14.

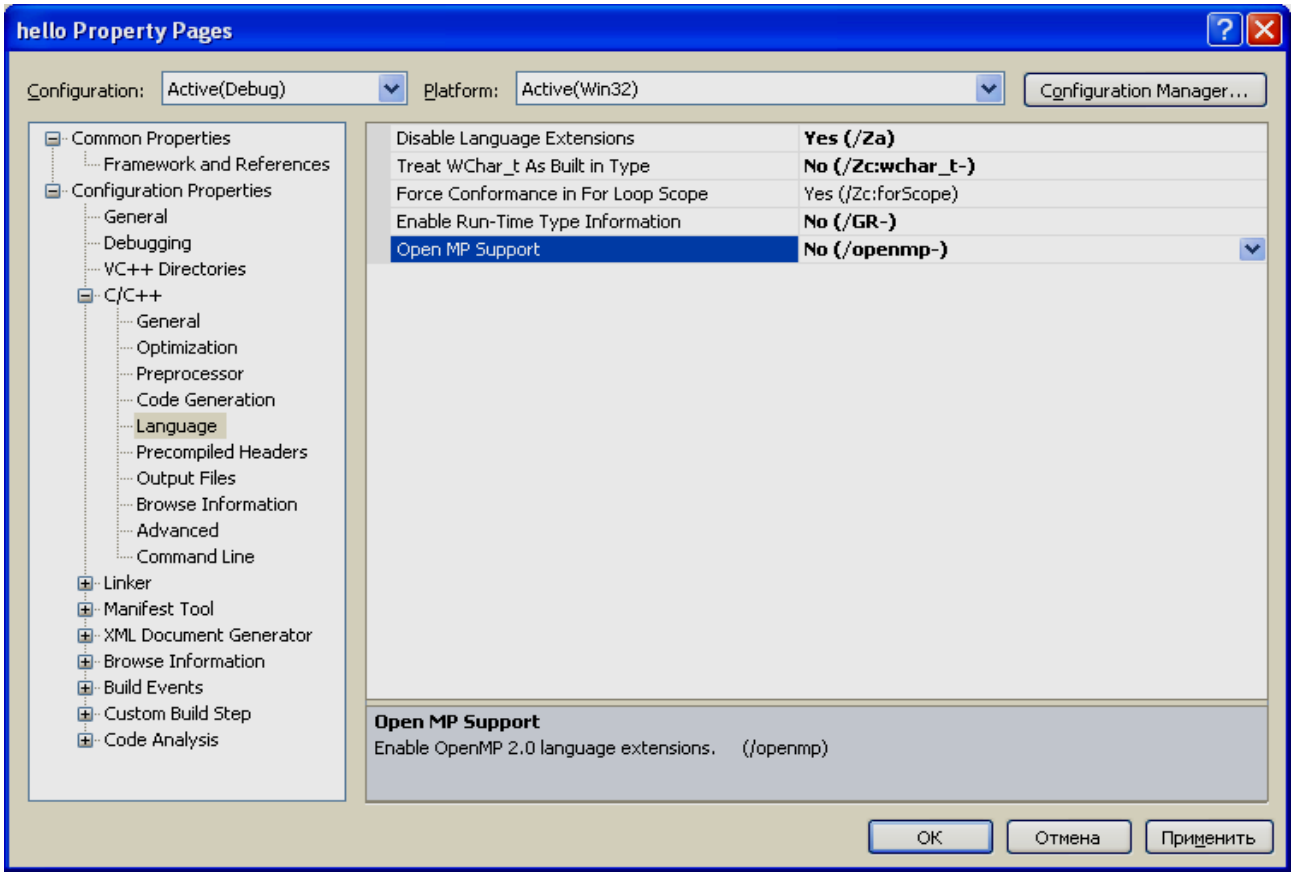


Рис. 10.13. Страница свойств закладки Language

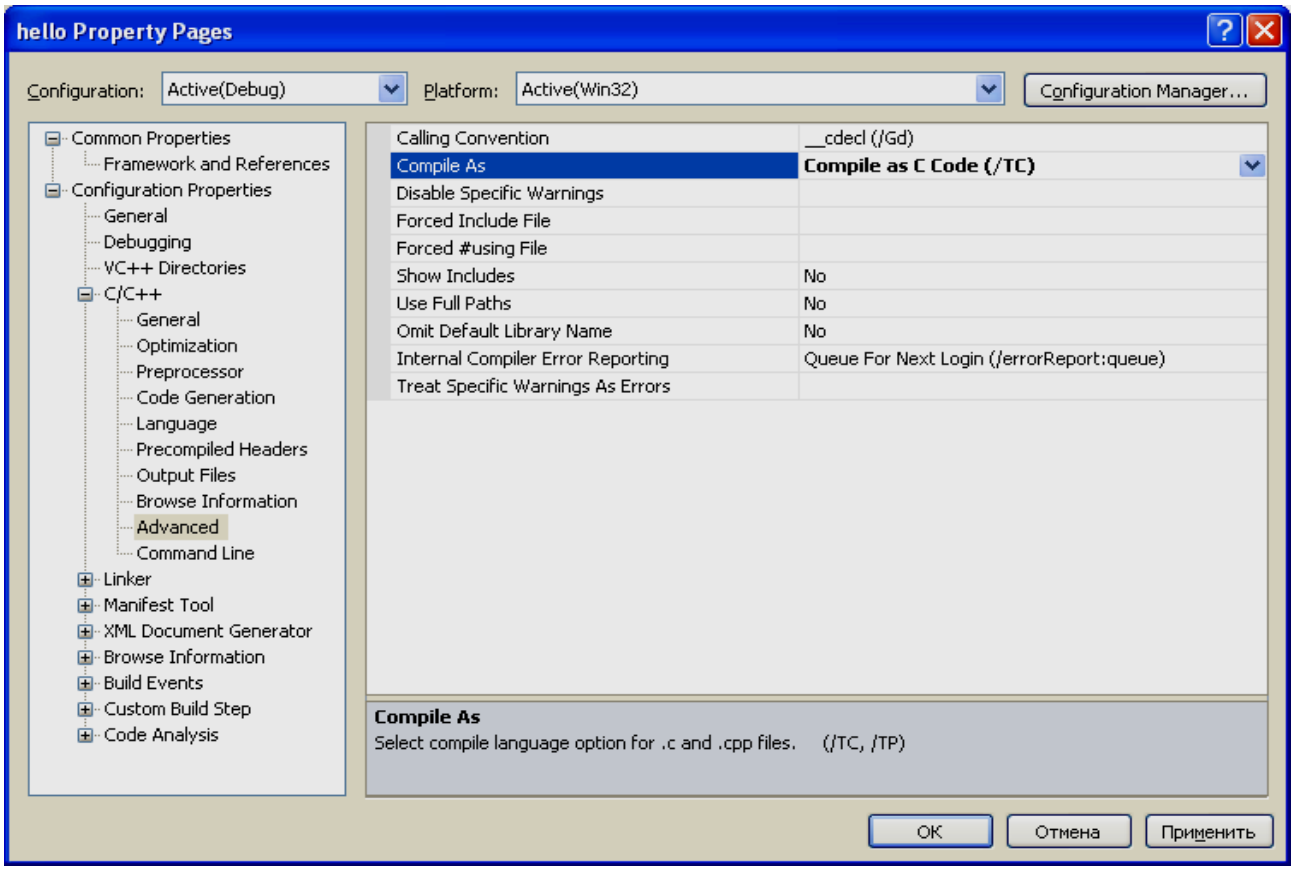


Рис. 1.14. Результат выбора режима компиляции языка **C**

После нажатия клавиш **Применить** и **ОК** откроется подготовленный проект с пустым полем редактора кода, в котором можно начать писать программы. В этом редакторе наберем программу, выводящую традиционное приветствие «Hello, World» для начинающих программистов:

```
//Фамилия студента и номер лабораторной работы.номер задачи
#include <stdio.h>
#include <conio.h>
int main(void)
{
printf("\n\t Hello, World!\n");
printf("\n ...Press any key:  ");
_getch();
return 0;
}
```

Для компиляции созданной программы можно обратиться в меню **Build** или, например, нажать клавиши **Ctrl + F7**. В случае успешной компиляции получим экранную форму, показанную на рис. 10.15.

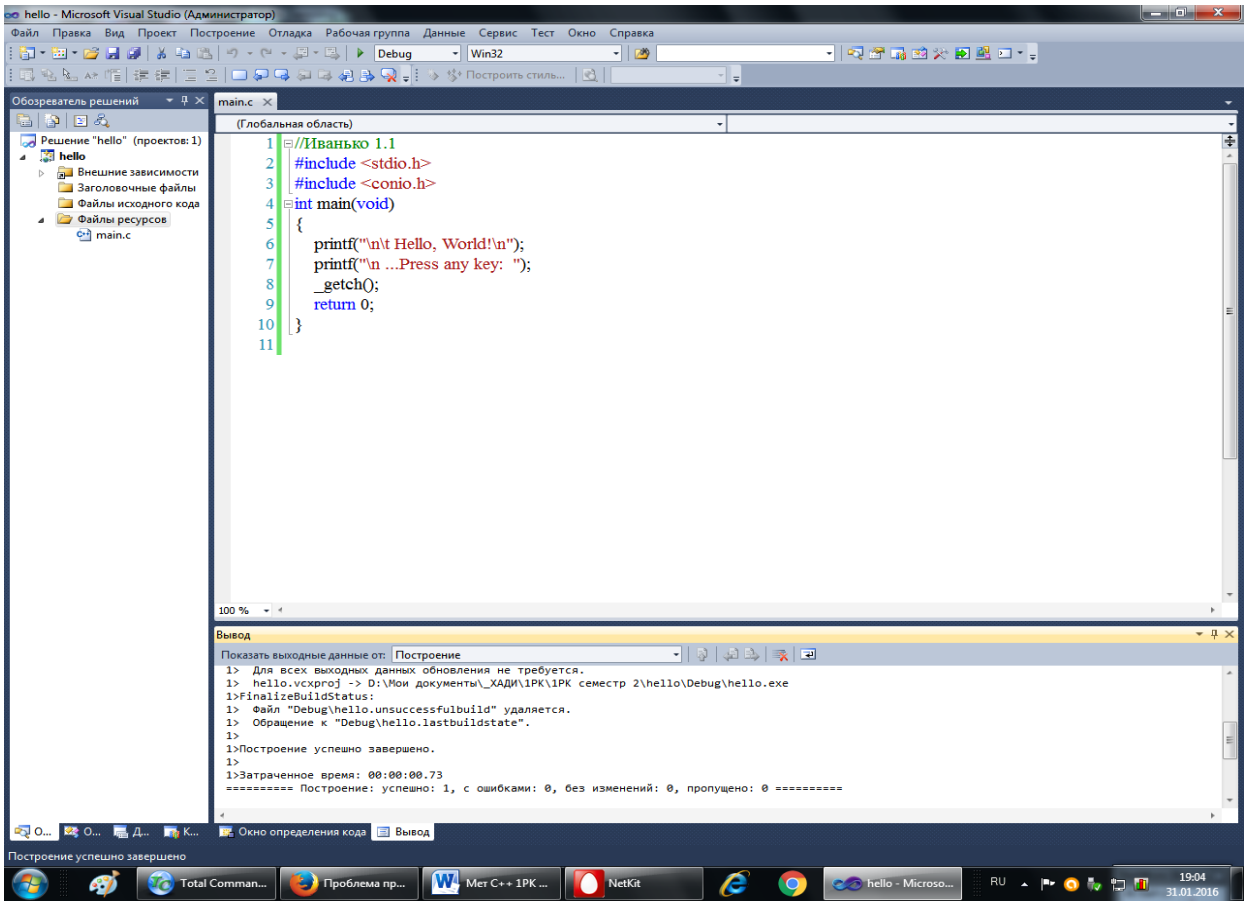


Рис. 10.15. Успешно скомпилированная первая программа на языке **С**

Для приведенного кода программы запуск на ее исполнение из окна редактора в Visual Studio 2010 выполняется нажатием клавиши **F5** (рис. 10.16) или через пункт меню **Отладка.**

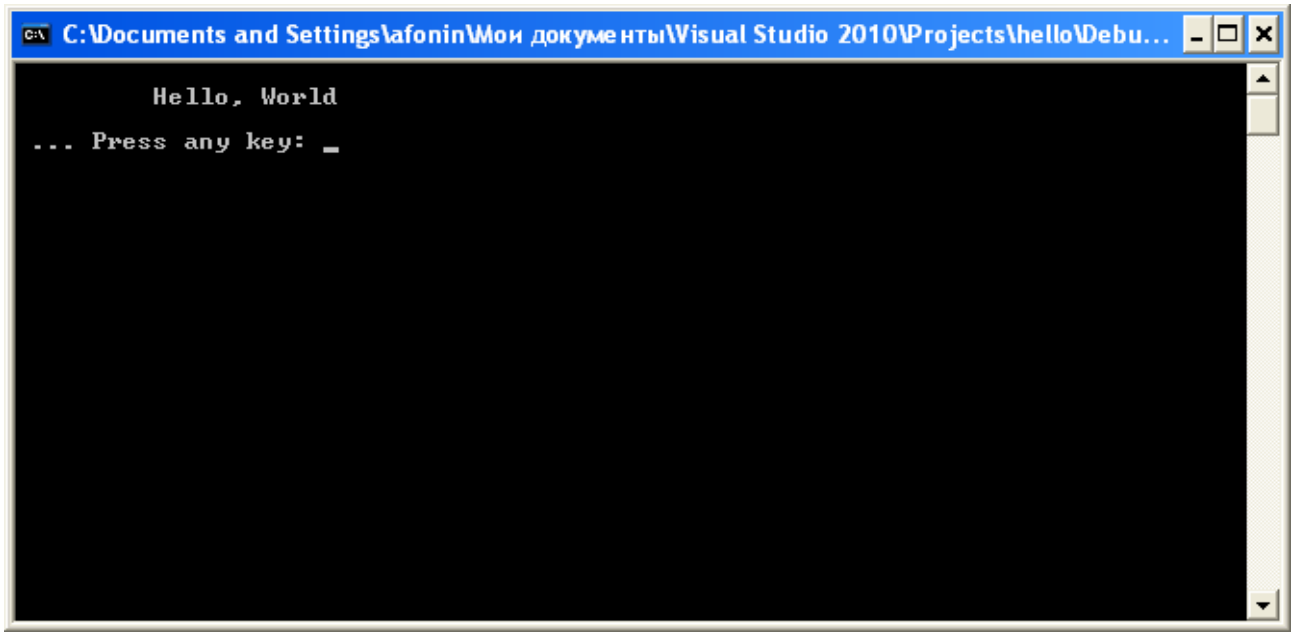


Рис. 10.16. Консольный вывод первой программы на языке **C**

Проведем анализ этой программы:

1. В приведенной программе используется заголовочный файл с именем **stdio.h** (стандартный ввод-вывод), который должен быть включен в начало программы. Для вывода сообщения на консоль используется функция **printf().**

2. Для работы с консолью в программу включен также заголовочный файл **conio.h**, поддерживающий функцию **_getch()**, которая извлекает символ из потока ввода, т. е. предназначена для приема сообщения о нажатии какой-либо (почти любой) клавиши на клавиатуре. С другими компиляторами, возможно, потребуется **getch()**, т. е. без префиксного нижнего подчеркивания.

3. Строка программы

int main (void)

сообщает системе, что именем программы является **main()** – главная функция и что она возвращает целое число, о чем указывает аббревиатура **int.**

Имя **main() –** это специальное имя, которое указывает, где программа должна начать выполнение. Наличие круглых скобок после слова **main** свидетельствует о том, что это имя функции. Если содержимое этих скобок отсутствует или в них представлено служебное слово **void**, то это означает, что в функцию **main()** не передается никаких аргументов. Тело функции **main()** ограничено парой фигурных скобок. Все утверждения программы, заключенные в них, будут относиться к этой функции.

В теле главной функции **main()** имеются, в свою очередь, другие функции:

1. Функция **printf()** находится в библиотеке компилятора языка **C** и печатает или отображает те аргументы, которые были подставлены вместо параметров.

Символ **\n** означает единый символ newline (новая строка), т. е. с его помощью выполняется перевод на новую строку. Символ **\t** осуществляет табуляцию, т. е. начало вывода результатов программы с отступом вправо.

2. Функция без параметров **_getch()** извлекает символ из потока ввода. С другими компиляторами, может потребоваться **getch()** без префиксного нижнего подчеркивания.

Запись **return 0** указывает на то, что выполнение функции **main()** закончено и что в систему возвращается значение **0** (целое число). Ноль используется в соответствии с соглашением об индикации успешного завершения программы.

Все операторы в программе завершаются символом точки с запятой.

Все файлы проекта сохраняются в той папке, которая сформировалась после указания в поле **Location** имени проекта (hello). На рис. 10.17 показаны папки и файлы проекта **Visual Studio 2010**.

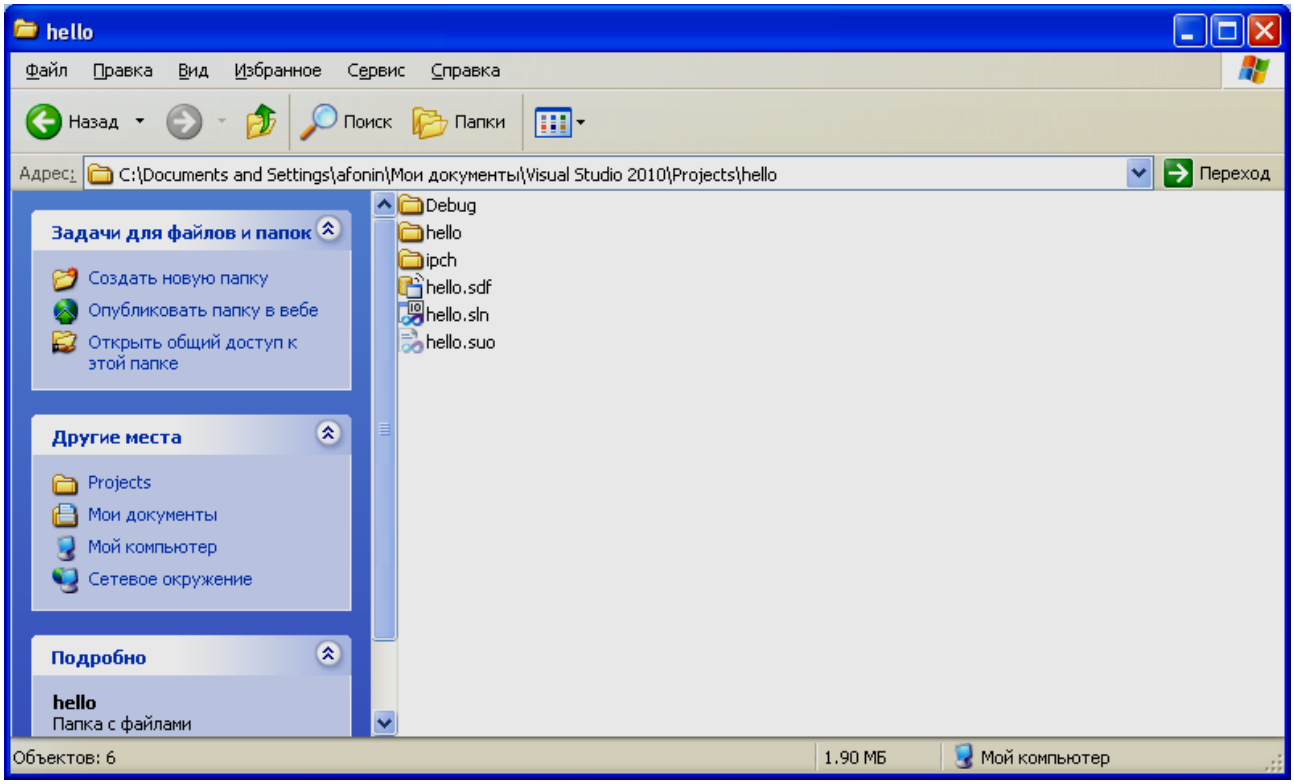


Рис. 10.17. Файлы и папки созданного проекта

Каждый файл обладает некоторым значением:

- **hello.sln** – файл решения для созданной программы. Он содержит информацию о том, какие проекты входят в данное решение. Обычно эти проекты расположены в отдельных подкаталогах. Например, наш проект находится в подкаталоге hello;
- **hello.suo** – файл настроек среды Visual Studio при работе с решением включает информацию об открытых окнах, их расположении и прочих пользовательских параметрах.
- **hello.sdf** – файл, содержащий вспомогательную информацию о проекте, который используется инструментами анализа кода Visual Studio, такими как IntelliSense для отображения подсказок об именах и т. д.

Файлы папки **Debug** представлены на рис. 10.18.

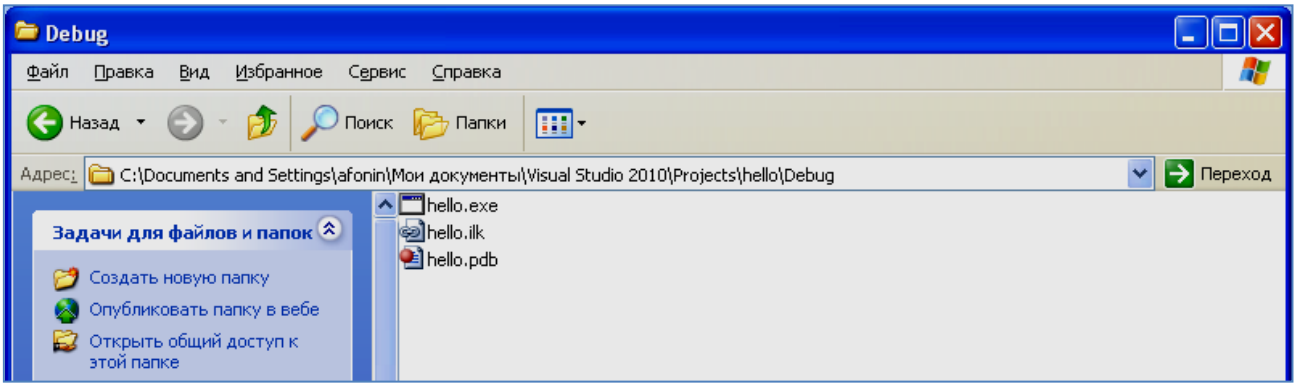


Рис. 10.18. Файлы папки Debug

Рассмотрим файлы папки Debug:

- **hello.exe** – исполняемый файл проекта;
- **hello.ilk** – файл «incremental linker», используемый компоновщиком для ускорения процесса компоновки;
- **hello.pdb** – отладочная информация (информация об именах в исполняемых файлах, используемая отладчиком).

Файлы папки **hello** показаны на рис. 10.19:

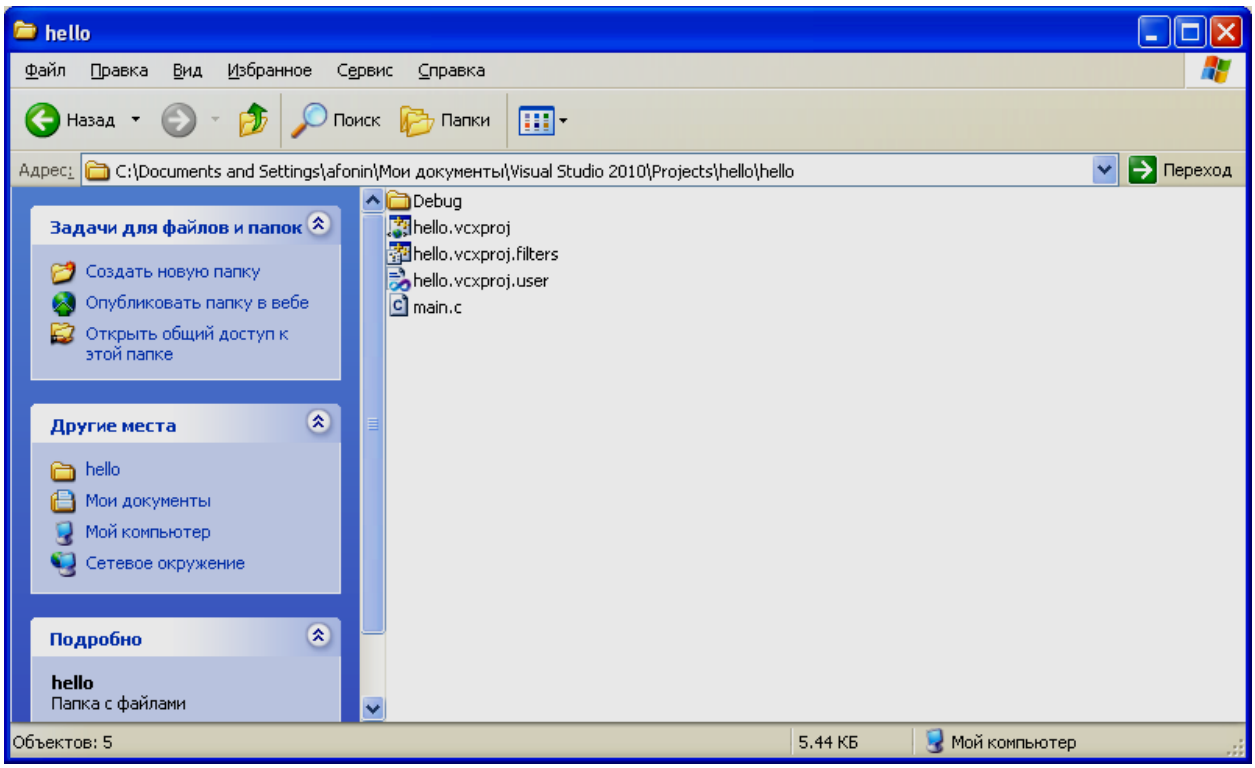


Рис. 10.19. Содержимое папки **hello**

Характеристика содержимого папки **hello**:

- **main.c** – файл исходного программного кода;
- **hello.vcxproj** – файл проекта;
- **hello.vcxproj.user** – файл пользовательских настроек, связанных с проектом;
- **hello.vcxproj.filters** – файл с описанием фильтров, используемых Visual Studio Solution Explorer для организации и отображения файлов с исходным кодом.

Поместите окно DOS с результатами решения **Задания 10.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr10-1.cpp** и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр10**. Над вставленным рисунком проставьте номер задания – **10-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS.

Откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

Задание 10.2. Напишите программу, которая выводила бы на консоль название факультета, где Вы учитесь, номер группы, Ваши фамилию, имя и отчество в разных строках дисплея (консоли) с помощью одной функции **printf()**.

Задание 10.3. Вывод выполните с помощью нескольких функций **printf()** (количество функций должно соответствовать каждой порции информации).

Задание 10.4. Для задания 1.2 вывод информации выполните в различных строках подряд, т. е. без межстрочного пропуска.

Проверьте программу без ключевого слова **void** для функции **main()**.

Примечание. Вывод требуемой информации осуществляется с помощью букв латинского алфавита. Комментарии в программе могут быть сделаны после символа // или внутри комбинации символов **/* */**.

2. Исследование возможностей операторов по выводу строк

В приводимых ниже программах используются такие средства ввода-вывода, как **printf()**, **getchar()**, **gets()**, **scanf()**. Приведем характеристику данных функций в соответствии с их прототипами.

Прототип функции **printf()**имеет следующий вид:

int printf(const char *format, …);

Функция **printf()** записывает в стандартный поток **stdout** (стандартный выходной поток данных) значения аргументов из заданного списка в соответствии со строкой форматирования, адресуемой параметром **format**.

Строка форматирования состоит из элементов двух типов. К элементам первого типа относятся символы, которые выводятся на экран. Элементы второго

типа содержат спецификации формата, определяющие способ отображения аргументов.

Спецификация формата начинается символом процента **%**, за которым следует код формата. На спецификации могут воздействовать модификаторы, задающие ширину поля, точность и признак выравнивания по левому краю. Целое значение, расположенное между **%** и командой форматирования, играет роль спецификации минимальной ширины поля. Наличие этого спецификатора приводит к тому, что результат будет заполнен пробелами или нулями, чтобы выводимое значение занимало поле, ширина которого не меньше заданной минимальной ширины.

По умолчанию в качестве заполнителя используется пробел (пробелы). Для заполнения нулями перед спецификацией ширины поля нужно поместить нуль.

Например, спецификация формата **%05d** дополнит нулями выводимое целое число, в котором менее пяти цифр, чтобы общая длина равнялась пяти символам. Действие модификатора точности зависит от кода формата, к которому он применяется. Чтобы добавить этот модификатор, следует поставить за спецификатором ширины поля десятичную точку, а после – требуемое значение точности (число знаков после десятичной точки).

Применительно к целым числам модификатор точности задает минимальное количество выводимых цифр. При необходимости перед целым числом будут добавлены нули. Если данный модификатор применяется к строкам, то число, следующее за точкой, задает максимальную длину поля. Например, спецификация **%5.7s** выведет строку длиной не менее пяти, но не более семи символов. Если выводимая строка окажется длиннее максимальной длины поля, конечные символы будут отсечены.

По умолчанию все выводимые значения выравниваются по правому краю: если ширина поля больше выводимого значения, то оно будет выровнено по правому краю поля. Чтобы установить выравнивание по левому краю, нужно поставить знак «минус» (–) сразу после знака процента.

Например, спецификация формата **%–10.4f** обеспечит выравнивание вещественного числа с четырьмя десятичными знаками по левому краю в 10-символьном поле.

Существуют два модификатора формата, позволяющие функции **printf()** отображать короткие и длинные целые числа. Модификатор **l** (латинская буква «эль») уведомляет функцию **printf()** о длинном типе значения. Модификатор **h** сообщает функции **printf()**, что нужно вывести число короткого целого типа. Кроме того, модификатор **l** можно поставить перед командами форматирования вещественных чисел, в этом случае он уведомит о выводе значения типа **long double**.

Спецификаторы формата для функции **printf()** перечислены в табл. 1.1.

Таблица 10.1. Спецификаторы формата функции **printf()**

Код	Формат
%c	Символ
%d	Десятичное целое число со знаком
%i	Десятичное целое число со знаком
%e	Экспоненциальное представление числа (в виде мантиссы и порядка, e — на нижнем регистре)
%E	Экспоненциальное представление числа (в виде мантиссы и порядка, E — на верхнем регистре)
%f	Десятичное число с плавающей точкой
%F	Десятичное число с плавающей точкой (только стандарт C99; если применяется к бесконечности или нечисловому значению, то выдает надписи **INF, INFINITY**(бесконечность) или **NAN** (not a number) на верхнем регистре. Спецификатор %f выводит
%g	их эквиваленты на нижнем регистре)
%G	Использует более короткий из форматов **%e** или **%f**
%o	Использует более короткий из форматов **%E** или **%F**
%s	Восьмеричное число без знака
%u	Символьная строка
%x	Десятичное целое число без знака
%X	Шестнадцатеричное без знака (строчные буквы)
%p	Шестнадцатеричное без знака (прописные буквы)
%n	Выводит указатель. Соответствующий аргумент должен быть указателем на целое число. Этот спецификатор указывает, что в целочисленной переменной, на которую указывает ассоциированный с данным спецификатором указатель, будет храниться число символов, выведенных к моменту обработки спецификации
%%	Выводит знак процента

Прототип функции **getchar()** имеет вид

int getchar(void);

Функция **getchar()** возвращает из стандартного потока **stdin** (входного потока данных) следующий символ. При чтении символа предполагается, что он имеет тип **unsigned char**, который потом преобразуется в целый. При достижении конца файла, как и при обнаружении ошибки, функция **getchar()** возвращает значение **EOF** (end of file – конец файла).

Прототип функции **gets** имеет следующий вид

char *gets(char *str);

Функция **gets()** читает символы (включая пробелы) из стандартного потока **stdin** и помещает их в массив символов, адресуемый указателем ***str** (далее – массив символов). Символы читаются до тех пор, пока не встретится разделитель строк или значение **EOF**. Для реализации **EOF** на клавиатуре следует набрать одновременно Ctrl + Z. Вместо разделителя строк в конец строки вставляется нулевой символ, свидетельствующий о ее завершении.

Следует учесть, что не существует способа ограничить количество символов, которое прочитает функция **gets()**. Поэтому массив, адресуемый указателем ***str**, может переполниться, и тогда программа выдаст непредсказуемые результаты.

Прототип функции **scanf()** имеет вид

int scanf(const char *format, …);

Функция **scanf()** предназначена для ввода данных общего назначения, которая читает поток **stdin** и сохраняет информацию в переменных, перечисленных в списке аргументов. Если в строке форматирования встретится разделитель, то функция **scanf()** пропустит один или несколько разделителей во входном потоке.

Под разделителем, или пробельным символом, подразумевают пробел, символ табуляции **\t** или разделитель строк **\n**. Все переменные должны передаваться посредством своих адресов, например с помощью символа **&**.

Управляющая строка, задаваемая параметром **format**, состоит из символов трех категорий: спецификаторов формата; пробельных символов; символов, отличных от пробельных. Спецификация формата начинается знаком **%** и сообщает функции **scanf()** тип данного, которое будет прочитано. Спецификации формата функции **scanf()** приведены в табл. 1.2.

Таблица 10.2. Спецификаторы формата функции **scanf()**

Код	Формат
%c	Читает один символ
%d	Читает десятичное целое число
%i	Читает целое число в любом формате (десятичное, восьмеричное или шестнадцатеричное)
%u	Читает беззнаковое целое десятичное число
%hd	Читает десятичное целое число типа **short int**
%e	Читает число с плавающей точкой (и в экспоненциальной форме)
%f	Аналогично коду **%e**
%lf	Читает число с плавающей точкой
%F	Читает десятичное число с плавающей точкой типа **double**
%g	Аналогично коду **%f** (для стандарта **C99**)
%G	Читает число с плавающей точкой.
%o	Аналогично коду **%g**

%x	Читает восьмеричное число
%X	Читает шестнадцатеричное число
%s	Аналогично коду **%x**
%p	Читает строку
%n	Читает указатель
%[ ]	Принимает целое значение, равное количеству прочитанных до сих пор символов
%%	Просматривает набор символов. Читает знак процента

Строка форматирования читается слева направо, и спецификации формата сопоставляются с аргументом в порядке их перечисления в списке аргументов. Символ *****, стоящий после знака **%** и перед кодом формата, прочитает переменные заданного типа, но запретит их присваивание. Команды форматирования могут содержать модификатор максимальной длины поля. Он представляет собой целое число, располагаемое между знаком **%** и кодом формата, которое ограничивает количество читаемых для всех полей символов. Если входной поток содержит больше заданного количества символов, то при последующем обращении к операции ввода чтение начнется с того места, в котором «остановился» предыдущий вызов функции **scanf()**. Если разделитель (например, пробел) встретится раньше, чем достигнута максимальная ширина поля, то ввод данных завершится. В этом случае функция **scanf()** переходит к чтению следующего поля. При чтении одиночных символов символы табуляции и разделители строк читаются подобно любому другому символу.

В программах возникает необходимость определять константы. В языке **C++** типы констант можно задавать явно при использовании суффиксов. Например:

```
long int  j = –12345678L;          /* суффикс L */
unsigned int a = 678U;             /* суффикс U */
float x = 123.45F;                 /* суффикс F */
long double z = 12345678.99L;      /* суффикс L* /
```

По умолчанию спецификации **f**, **e**, **g** заставляют функцию **scanf()** присваивать вводимым переменным тип **float**. Если перед одной из этих спецификаций поставить модификатор **l**, то функция **scanf()** присвоит прочитанные данные переменной типа **double**.

Функция **scanf()** поддерживает спецификатор формата общего назначения, называемый набором сканируемых символов. В этом случае определяется набор символов, которые могут быть прочитаны функцией **scanf()** и присвоены соответствующему массиву. Для определения такого набора символы, подлежащие сканированию, необходимо заключить в квадратные скобки. Открывающая квадратная скобка должна следовать сразу за знаком процента. При использовании набора сканируемых символов функция **scanf()** продолжает читать символы и

помещать их в соответствующий массив до тех пор, пока не встретится символ, отсутствующий в данном наборе. Если первым символом в наборе является знак ^, то получится обратный эффект: входное поле будет читать до тех пор, пока не встретится символ из заданного набора сканируемых символов, т. е. знак ^ заставляет функцию **scanf()** читать только те символы, которые отсутствуют в наборе сканируемых символов. Если в строке форматирования встретился символ, отличный от разделителя, то функция **scanf()** прочитает и отбросит его. Если заданный символ не найден, то функция **scanf()** завершает работу.

В MS Visual Studio 2010 рекомендуется в целях безопасности применять функции **gets_s()** и **scanf_s()**. Для них при чтении символа или строки следует указать размер в байтах, соответственно для символа или строки. Например, **scanf_s("%c", &ch, 1)**.

В Visual Studio 2010 тип данных **char** занимает 1 байт. Размерность типа можно определять с помощью функции **sizeof()**. Например, **sizeof(char)**.

Задание 10.5. Написать программу ввода символа, строки, действительных и целых чисел. Действительные числа сложить, целые перемножить. Для действительных чисел необходимо использовать типы **float** и **double**.

Программный код решения следующий:

```
#include <stdio.h>
#include <conio.h>
int main (void) {

    //Объявления
char ch, str[79+1];          // С учетом одного места для символа '\0'
int x, y, z;
float a, b, c;
double A, B, C;
    //Выполнение программы
printf("\n\t Enter a symbol: ");
ch=getchar();
printf("\t The symbol is: %c\n", ch);
   _flushall();
printf("\n\t Enter a string: ");
gets_s(str, 79);
printf("\t The string is: %s\n", str);
a=2.42F;
b=3.58F;
c=a+b;
printf("\n\t The sum %1.2f and %1.2f (as float) is equal: %1.4f\n", a, b, c);
A=12.1234567796602;
B=2.7182818284509;
C=A+B;
printf("\n\t The sum %1.13f and %1.13f \n\t is equal (as double): %1.13f\n",A,B,C);
```

```
x=2;
y=7;
z=x*y;
printf("\n\t Multiplication %d on %i (as an integer) is equal: %d\n", x, y, z);
printf("\n\n Press any key: ");
_getch();
return 0;
}
```

В объявлении символьного массива **str[79+1]** в методических целях сделано напоминание, что для строки следует предусмотреть символ ее завершения, т. е. '**\0**'. В определении переменных типа **float** добавлены суффиксы **F**.

Результат выполнения программы показан на рис. 10.20.

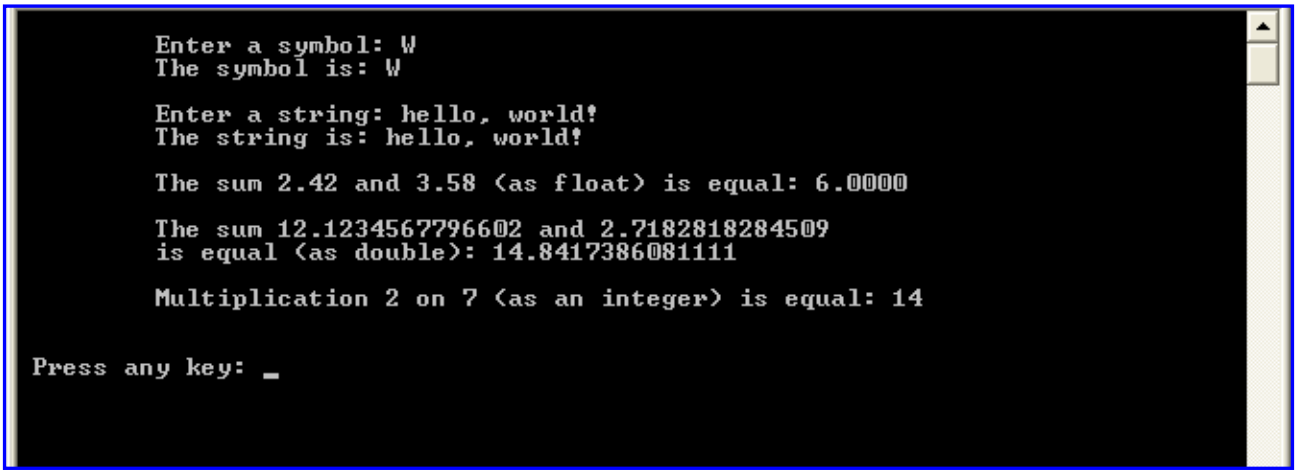


Рис. 10.20. Результат программы с базовыми типами данных

Добавим считывание данных с помощью функции **scanf_s()**. В результате получим следующую программу:

```
#include <stdio.h>
#include <conio.h>
int main (void)
{
   // Объявления
char ch, str[79+1];
int x, y, z;
float a, b, c;
double A, B, C;
   // 1-я часть
printf("\n\t Enter a symbol: ");
ch = getchar();
printf("\t The symbol is: %c\n", ch);
   _flushall();
printf("\n\t Enter any string: ");
```

```
gets_s(str, 79);
printf("\t The string is: %s\n", str);
a=2.42F;
b=3.58F;
c=a+b;
printf("\n\t The sum %1.2f and %1.2f (as float) is equal: %1.4f\n", a, b, c);
A=12.1234567796602;
B=2.7182818284509;
C=A+B;
printf("\n\t The sum %1.13f and %1.13f \n\t is equal (as double): %1.13f\n",A,B,C);
x=2;
y=7;
z=x*y;
printf("\n\t The multiplication %d on %i (as an integer) is equal: %d\n", x, y, z);
   // 2-я часть
printf("\n\t Enter a symbol: ");
scanf_s("%c", &ch, 1);
printf("\t The symbol is: %c\n", ch);
printf("\n\t Enter any string: ");
scanf_s("%s", str, 79);
printf("\t The string is: %s\n", str);
   _flushall(); // для устранения пустой строки
printf("\t Enter a real number a: ");
scanf_s("%f", &a);
printf("\t Enter a real number b: ");
scanf_s("%f", &b);
c=a+b;
printf("\t The sum %1.2f and %1.2f (as float) is equal: %1.4f\n", a, b, c);
printf("\n\t Enter a real number A: ");
scanf_s("%lf", &A);
printf("\t Enter a real number B: ");
scanf_s("%lf", &B);
C=A+B;
printf("\t The sum %1.13f and %1.13f \n\t is equal (as double): %1.13f\n", A, B,C);
printf("\n\t Enter an integer x: ");
scanf_s("%d", &x);
printf("\t Enter an integer y: ");
scanf_s("%i", &y);
z=x*y;
printf("\t Multiplication %d on %i (as an integer) is equal: %d\n", x, y, z);
printf("\n\n Press any key: ");
_getch();
return 0;
}
```

Результат выполнения видоизмененной программы представлен на рис. 10.21.

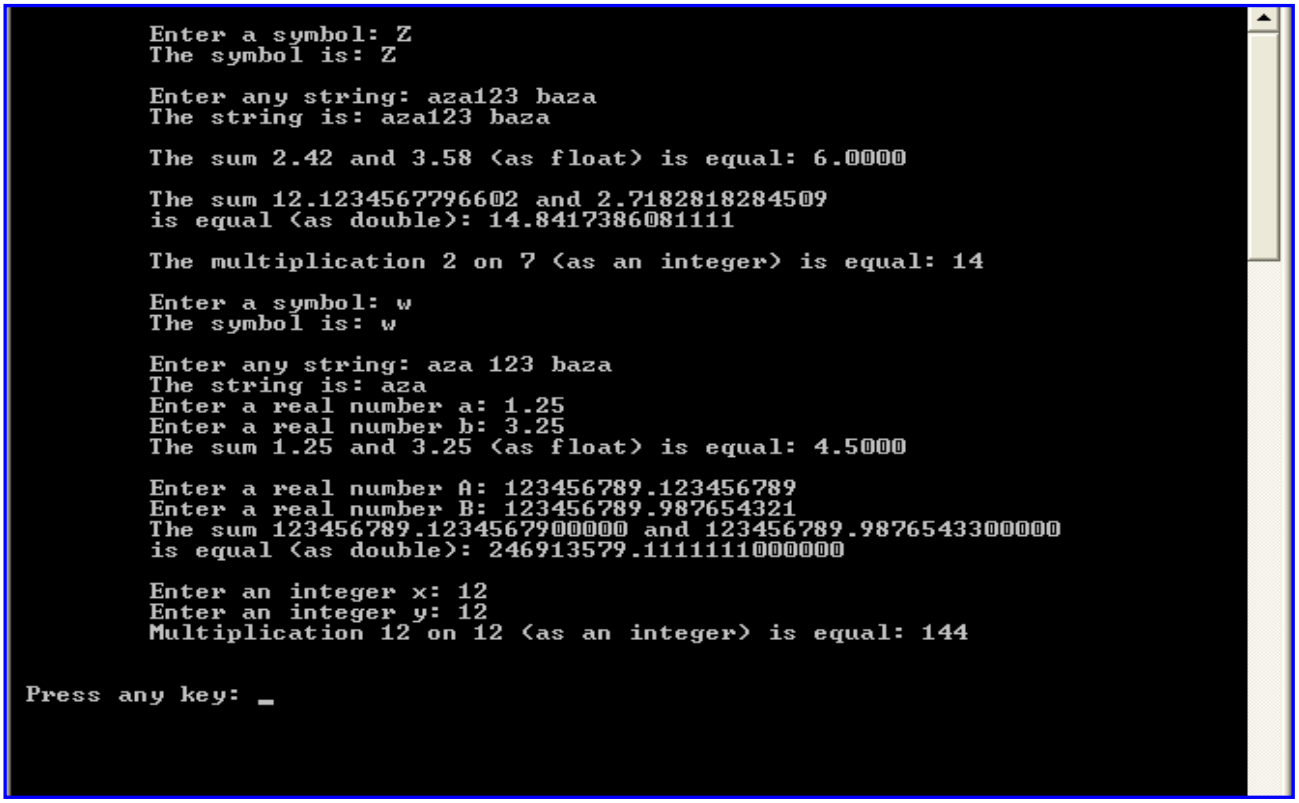


Рис. 10.21. Результат программы с различным вводом данных

В начале функции **int main(void)** сделано объявление переменных, которые будут использоваться в программе. Каждый тип переменных отделен от следующего запятой. В частности, **char str[80]** означает массив символов, в квадратных скобках указано их число. Функции **printf()**выводят либо только сообщения, либо еще заданные переменные соответствующих типов.

Функция **_flushall()** нужна для того, чтобы устранить пробельную строку, образовавшуюся после действия либо функции **getchar()**, либо функции **scanf_s()**. В некоторых компиляторах функция **_flushall()** используется без префиксного нижнего подчеркивания.

Функция **gets_s()** позволяет считывать символы с наличием разделителей, в частности с пробелами.

Следует обратить внимание на формат записи функций **scanf_s()**. Если сканируются числа или одиночные символы, то их присваивание переменным, которые были объвлены через соответствующие типы данных, осуществляется с помощью взятия адреса этих переменных, т. е. с помощью символа **&**, например **scanf_s("%c",&ch, 1)**. При сканировании массива символов, т. е. строки, символ **&** не используется. Обращение к адресу осуществляется с помощью указателей (будут рассмотрены позднее). Имя массива само по себе является указателем. Для сканирования чисел типа **double** в функции **scanf_s()** используется спецификатор **l**.

Задание 10.6. Разработайте программу для ввода и вывода следующей информации:

1. В качестве вводимых символов используйте начальные буквы своей фамилии

(набранные латиницей). В отчет вставьте полученный результат.

2. В качестве вводимой строки используйте свои фамилию и имя (буквами латинского алфавита). В отчет вставьте полученный результат.
3. Подсчитайте количество символов, на которые производит отступ от левого края символ табуляции **\t**. Для вывода строки предусмотрите табулированный вывод с помощью спецификатора формата **%xs**, где **x** – требуемое число позиций отступа.
4. Объявите массив символов в размере одного символа и введите строку с помощью функции **gets_s()**, превышающую число 1.
5. Запустите программу без функций **_flushall()**. В отчет вставьте полученный результат.
6. Ввод чисел произведите со знаком «+» и «–». Введите только целые числа. Введите только вещественные числа.
7. Сложите два вещественных числа типа **float** с десятью знаками после десятичной точки. Выведите и проанализируйте результат сложения. Объясните результат.
8. Изучите работу функции **puts()** и примените ее вместо функции **printf()**, которая использовалась в режиме сообщений. В отчет вставьте полученный результат. Объясните результат.

Задание 10.7. Разработать программу вычисления алгебраических выражений с приведением типов **char**, **int**, **float**, **double**.

Исследуйте следующий программный код для решения данного задания:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>
int main (void) {
char ch;
int a, b;
float x;
double y, z;
ch='C';
a=2;
b=5;
x=5.5F;
y=6.0;
z=ch+a+b+x+y;

   // Результат приведения типов
printf("\n\t The result of the conversion types: %lf\n", z);
   // Явное преобразование типов
z=(double)ch+(double)a+(double)b+(double)x+y;
```

```
printf("\n\t The apparent conversion types: %lf\n", z);
z=sqrt((double)a/b);
printf("\n\t z = sqrt(%d/%d) = %lf\n", a, b, z);
printf("\n\t z = lg(%d/%d) = %lf\n", a, b, log10(z));
printf("\n\t z = ln(%d/%d) = %lf\n", a, b, log(z));
printf("\n Press any key: ");
_getch();
return 0;
}
```

В программу включена библиотека математических функций **math.h**, в которой **sqrt()** – функция извлечения квадратного корня, **log10()** – логарифмическая функция по основанию 10, **log()** – функция натурального логарифма. Все эти функции возвращают результат типа **double** и вычисляют от выражения также типа **double**.

Результат выполнения программы представлен на рис. 10.22.

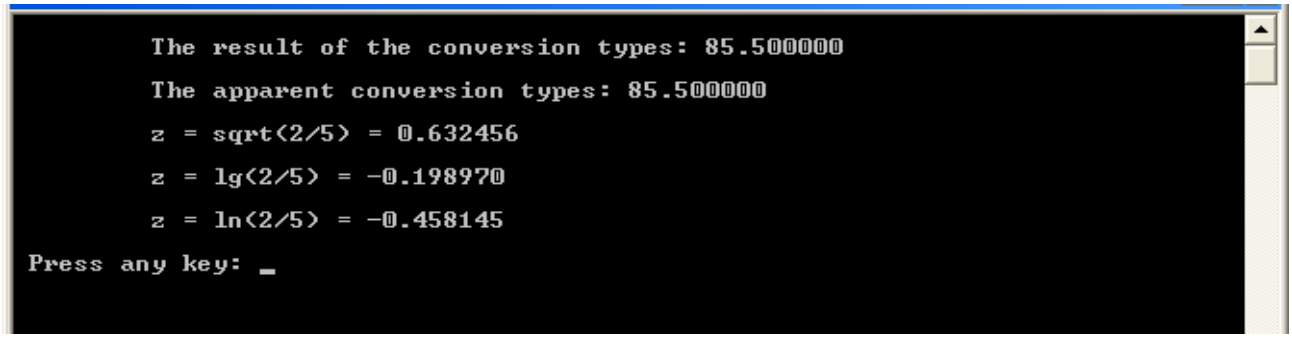


Рис. 10.22. Вычисления с приведением типов

Задание 10.8. Разработать программу для выполнения следующих операций:

1. В качестве вводимого символа используйте первую букву своей фамилии (буквами латинского алфавита). В отчет вставьте полученный результат.
2. Переменную **y** задайте в виде **y=6**.
3. Вычислите заданный квадратный корень без явного приведения типов.
4. Определите переменные с суффиксами **U**, **F**, **L** и произведите с ними арифметические действия.
5. Вычислите логарифм от числа 10X по основанию 2X, где X – номер компьютера, на котором выполняется лабораторная работа. Выполните тестовый пример для проверки вычислений, т. е. для заданного основания подберите число, от которого логарифм с заданным основанием даст целое число. Например, **log10(100.0) = 2.0000**.

Задание 10.9. Разработать программу вычисления площади круга и длины его окружности по заданному радиусу, вводимому пользователем с клавиатуры, а также вывода на консоль максимальных значений чисел типа **int**, **float** и **double**.

Для решения примера следует воспользоваться математической библиотекой компилятора, т. е. включить в программу заголовочные файлы **<math.h>**, **<limits.h>**, **<float.h>**.

Исследуйте следующий программный код для решения данного задания:

```
#include <stdio.h>
#include <conio.h>
#define _USE_MATH_DEFINES // для числа пи (π)
#include <math.h>
#include <limits.h>
#include <float.h>
int main (void){
double R, Sr, Lr;
printf("\n Enter a real greater than zero: ");
scanf_s("%lf", &R);
Sr=M_PI*R*R;
Lr=2*M_PI*R;
printf("\n Area of a circle of radius R = %g is %g", R, Sr);
printf("\n The length of a circle of radius R = %g is %g",R,Lr);
puts("");
printf("\n  Maximum integer: %d\n ", INT_MAX);
printf(" Maximum real number of float: %g\n ", FLT_MAX);
printf("Maximum real number type double: %g\n ", DBL_MAX);
printf("\n Press any key: ");
_getch();
return 0;
}
```

В программу включена константа **_USE_MATH_DEFINES** для работы с числом **M_PI** (π). Остальные константы можно найти в справочной документации компилятора системы MS Visual Studio 2010.

Функция **scanf_s()** определена в компиляторе языка **C** системы MS Visual Studio 2010. С ней компилятор обеспечивает безопасную работу и не выдает предупреждений.

Результат выполнения программы представлен на рис. 10.23.

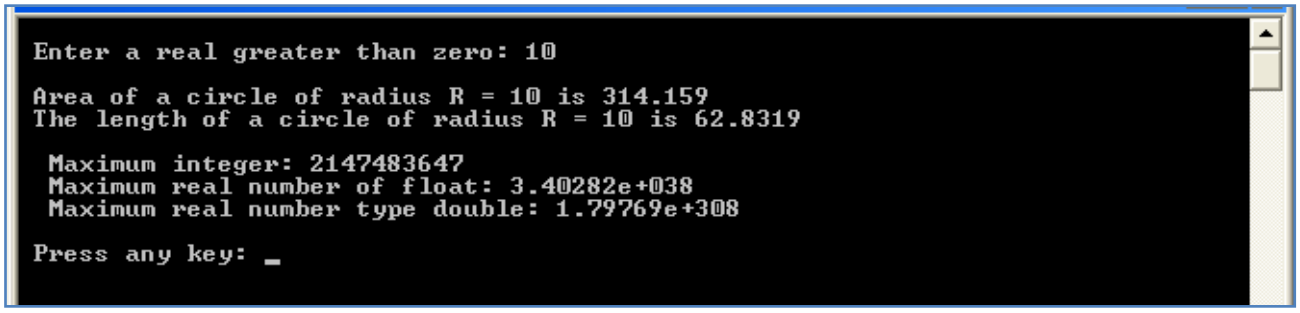


Рис. 10.23. Пример использования предопределенных констант

Задание 10.10. Разработать программу для выполнения следующих операций:

1. Напишите программу по вычислению площади эллипса. Данные для расчета должны вводиться с клавиатуры пользователем.
2. Рассчитайте величины lg(exp(100)) и ln(exp(100)), пользуясь стандартными функциями математической библиотеки (**math.h**) .
3. Используя справочную документацию компилятора, в программе предусмотрите вывод минимальных значений указанных типов данных.
4. Используя справочную документацию компилятора, в программе предусмотрите вывод максимальных и минимальных значений для следующих типов даных: **char**, **short int**, **long int**.

Контрольные вопросы

1. Какие компиляторы языка **C++** Вам известны?
2. Какое имя имеет исполняемый файл созданного проекта?
3. Каково назначение заголовочных файлов **stdio.h**, **conio.h**?
4. Как будет работать программа без заголовочного файла **conio.h**?
5. В каком месте программы находится точка ее входа?
6. Как осуществляется табуляция строки на консоли и на сколько позиций выполняется отступ от левого края?
7. Какое значение имеет главная функция проекта **main()** в программах на языке C++?
8. Чем отличаются функции **printf()** и **puts()** при консольном выводе информации?
9. Для чего в программах на **C** используется заголовочный файл **math.h**?
10. С какими разделителями может происходить считывание информации с консоли при использовании функции **gets_s()**?
11. Какой тип данных возвращает функция **gets_s()** при считывании информации?
12. Как осуществляется считывание последовательности различных типов данных с консоли с помощью одной функции **scanf_s()**?
13. Как выводится на консоль последовательность различных типов данных при использовании одной функции **printf()**?
14. Как осуществляются автоматическое и принудительное приведение типов в языке **C**?
15. Какие машинно зависимые типы данных имеются в языке **C**?

Лабораторная работа № 11 (2 занятия)

УКАЗАТЕЛИ И МАССИВЫ В СРЕДЕ VISUAL C++ 2010

Цель работы: изучение взаимосвязи указателей и массивов, как числовых, так и символьных; анализ допустимых операций с указателями и массивами в среде Visual C++ 2010.

1. Доступ к элементам одномерных массивов через указатели

Одной из наиболее распространенных конструкций с использованием указателей являются массивы. В результате применения указателей при операциях с массивами требуется меньшее количество используемой памяти и обеспечивается высокая производительность.

В языке **C++** массивы – это упорядоченные данные (элементы) одного типа.

Например, имеется массив **A** из десяти элементов и указатель **pA**:

```
double A[10];
double *pA;
```

тогда язык C++ позволяет записать следующую операцию присваивания

pA=A.

которая является инициализацией указателя **pA** адресом массива **A**. При этом адрес первого элемента массива **A** присваивается указателю **pA.**

В то же время можно применить прямую адресацию:

pA=&A[0].

Обе формы записи эквивалентны.

Аналогичные утверждения будут справедливыми для других типов массивов: **char**, **float**, **double** и пр.

!!!Имя массива без индекса образует указатель на начало этого массива.

Следует помнить следующее: отличие имени массива от указателя состоит в том, что имя массива не может быть изменено. Оно всегда указывает на одно и то же место в памяти – на нулевой элемент.

Язык C++ дает возможность получения доступа к элементам массивов, используя привычные операции с массивами, так и используя указатели. Например, если имя массива с пятью элементами **A**, то компилятор преобразует i-й элемент ($0 \le i < 5$) по правилам работы с указателями с операцией разыменования:

A[i]=*(A+i).

Здесь **A** как бы указатель, а **i** – целочисленная переменная. Сумма **(A+i)** указывает на i-й элемент массива, а операция разыменования (оператор раскрытия ссылки *****) дает его значение.

Продемонстрируем это на примере.

Задание 11.1. Необходимо разработать программный код для вывода элементов целого произвольного массива **A[5]** как привычным способом, так и с помощью указателей.

Программный код для решения данной задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, A[5]={23,32,17,29,12};              //Инициализация массива A[5]
int *pA;                                   //Объявление указателя на int
   for(i=0; i<5; i++)                      //Цикл для вывода элементов A[i]
      cout<<"   A["<<i<<"]= "<<A[i];       //Вывод i-го элемента массива  A
pA=A;                                      //Инициализация указателя pA
cout<<endl<<"   pA="<<A<<endl;            //Вывод указателя на массив  A
   for(i=0; i<5; i++)                      //Цикл для вывода элементов A[i]
      cout<<"   A["<<i<<"]= "<<*(A+i);     //Вывод i-го элемента массива  A при
                                           //помощи указателя
getch();
return 0;
}
```

В результате выполнения программы получен следующий результат:

```
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12
pA=0028FB74
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12_
```

Очевидно, что при помощи адресации и указателя исходный массив выведен на экран правильно, т. е. доступ к элементам массива через указатель возможен.

Поместите окно DOS с результатами решения **Задания 11.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr11-1.cpp** и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр11**. Над вставленным рисунком проставьте номер задания – **11-1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS.

Эти действия выполняйте строго для всех заданий лабораторной работы.

Откройте пункт меню **Файл** и выполните команду **Закрыть решение**.

Задание 11.2. Необходимо разработать программный код для вывода элементов массива **A=[5,4; 6,2; 2,6; 8,4; 7,3; 11,8; 4,3; 10,2]** обычным способом и с помощью указателей. Элементы выводимого массива в операторе вывода с использованием указателей необходимо увеличить в 5 раз. Для вывода использовать оператор **print**.

2. Операции с указателями на массивы

Если указатель **ptr** указывает на первый элемент (с нулевым индексом) массива **A**, то для обращения к следующему элементу массива допустимы такие формы:

pA=&A[1]

или

pA+=1.

Соответственно выражение ***(pA+1)** будет ссылаться на значение, содержащееся в элементе **A.**

Выражение

ptr+=n

заставит указатель ***ptr** ссылаться на элемент массива, находящийся на расстоянии **n** от того, на который ранее ссылался указатель, независимо от типа элементов, содержащихся в массиве. Разумеется, значение **n** должно быть в допустимых пределах для данного объема массива.

При работе с указателями и массивами удобны операторы инкремента **(++)** и декремента **(--)**. Использование оператора инкремента с указателем аналогично операции суммирования с единицей, а операция декремента имеет тот же эффект, что и вычитание единицы из указателя.

В языке программирования **C++** корректной операцией является сравнение указателей. К указателям применяются операции сравнения >, >=, **!**=, ==, <=, < [3]. Сравнивать указатели допустимо только с другими указателями того же типа или с константой **NULL**, обозначающей значение условного нулевого адреса.

Константа **NULL** – это особое значение переменной-указателя, присваиваемое ей в том случае, когда она не должна иметь никакого значения. Его можно присвоить переменной-указателю любого типа. Оно представляет собой целое число нуль. Особое значение имеет сравнение двух указателей, которые связаны с одним и тем же массивом данных.

Задание 11.3. Необходимо разработать программный код для вывода элементов целого массива **A[5]** с использованием указателей. Доступ к элементам массива **A** осуществить при помощи операции инкремента указателей.

Программный код для решения данной задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, A[5]={23,32,17,29,12};                  //Инициализация массива A[5]
int *pA;                                       //Объявление указателя на int
   for(i=0; i<5; i++)                          //Цикл для вывода элементов A[i]
      cout<<"   A["<<i<<"]= "<<A[i];           //Вывод i-го элемента массива  A
pA=A;                                          //Инициализация указателя pA
cout<<endl<<"   pA="<<A<<endl;                 //Вывод указателя на массив  A
   for(i=0; i<5; i++)                          //Цикл для вывода элементов A[i]
      cout<<"   A["<<i<<"]= "<<*(pA++);        //Вывод i-го элемента массива  A
при
                                               //помощи указателя
getch();
return 0;
}
```

В результате выполнения программы получен следующий результат:

```
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12
pA=0028FB74
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12_
```

Очевидно, что при помощи адресации и указателей исходный массив выведен на экран правильно, т. е. доступ к элементам массива через указатели возможен и при использовании операции инкремента.

Задание 11.4. Необходимо разработать программный код для вывода элементов массива **A=[5]** с помощью указателей и операции инкремента указателей.

Программный код для решения данной задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, A[5]={23,32,17,29,12};                  //Инициализация массива A[5]
int *pA;                                       //Объявление указателя на массив
   for(i=0; i<5; i++)                          //Цикл для вывода элементов A[i]
```

```
      cout<<"   A["<<i<<"]= "<<A[i];     //Вывод i-го элемента массива  A
pA=A;                                    //Инициализация указателя pA
cout<<endl<<"   pA="<<A<<endl;           //Вывод указателя на массив  A
   for(pA=&A[0],i=0;pA<=&A[4],i<=4;pA++,i++)
      cout<<"   A["<<i<<"]= "<<*pA;     //Вывод массива с использованием
                                         //инкремента указателя (вариант 1)
cout<<endl;
for(pA=&A[0],i=0; i<5; i++)
cout<<"   A["<<i<<"]= "<<pA[i];       //Вывод массива с использованием
                  //инкремента указателя (вариант 2)
getch();
return 0;
}
```

В результате выполнения программы получен следующий результат:

```
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12
pA=0018F7BC

A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12
A[0]= 23   A[1]= 32   A[2]= 17   A[3]= 29   A[4]= 12
```

Очевидно, что при помощи адресации и указателей исходный массив выведен на экран правильно, т. е. доступ к элементам массива через указатели возможен и при использовании операции инкремента.

Задание 11.5. Необходимо разработать программный код для вывода элементов массива **A=[5,4; 6,2; 2,6; 8,4; 7,3; 11,8; 4,3; 10,2]** с помощью указателей и операции инкремента указателей.

Организовать вывод значений адресов элементов массива **A** и значений элементов массива A параллельными столбцами.

Элементы выводимого массива в операторе вывода с помощью указателей необходимо увеличить на 10.

Задание 3.6. Самостоятельно разработать программу для выполнения следующих действий:

1. Задать массив **X** действительных чисел из 8 элементов непосредственной инициализацией в программном коде:

X[8]={3.9 2.5 3.6 6.2 5.0 3.3 2.7 4.6}.

2. Вычислить массив **Y** по заданной формуле, согласно своему варианту (таблица 3.1) с использованием указателей.

!!! Определить свой вариант как номер компьютера.

3. Массивы **X** и **Y** вывести на экран с использованием указателей.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи и записать формулу для расчета на языке С++ с указателями.

Таблица 11.1 Исходные данные и формулы для расчета Y

№ варианта	Формула для расчета **Y**	Значение **a**
1	$Y_i = \operatorname{tg}\frac{1-x_i}{1+x_i} + \sin^2 5x_i + e^{5a}$	a=1,88
2	$Y_i = \frac{\sin\frac{x_i+1}{4}}{\sin^2 5x_i + e^{3a}}$	a=0,56
3	$Y_i = \frac{\sin^3(x_i+a) - \cos^2(x_i+a)}{(x_i+a)^4}$	a=1,77
4	$Y_i = \frac{\operatorname{tg}^3(x_i+a) - \arccos^2(x_i+a)}{(x_i+a)^4}$	a=1,33
5	$Y_i = \operatorname{ctg}\frac{1-3x_i}{1+2x_i} + \cos^2 5x_i + e^{3a}$	a=0,46
6	$Y_i = \frac{\cos^3(x_i+a) - 7(x_i+a)}{\operatorname{tg}(x_i+a)^4}$	a=1,82
7	$Y_i = \frac{\cos\left(\frac{3a+x_i}{4}\right)}{\sin^3 3x_i + e^{4a}}$	a=0,82
8	$Y_i = \frac{\sin^3(x_i+a) - \cos^2(x_i+a)}{(x_i+a)^4}$	a=1,47
9	$Y_i = \frac{\operatorname{tg}\frac{4a^2+1}{4}}{\cos^3 2x_i + e^{2a}}$	a=1,34
10	$Y_i = \sin\frac{1-x_i}{1+x_i} + \operatorname{tg}^4 5x_i + e^{5a}$	a=1,28
11	$Y_i = \frac{\sin^3(x_i+a) - \arccos^2(x_i+a)}{\cos(x_i+a)^4}$	a=0,66
12	$Y_i = \frac{\operatorname{ctg}\left(\frac{x_i^3+1}{4}\right)}{\cos^2 5x_i + e^{3a}}$	a=1,12

13	$Y_i = \frac{ctg^3(3x_i + a) - \sin^2(x_i + 7a)}{(5x_i + a)^3}$	a=1,82
14	$Y_i = \frac{\arcsin^3 \frac{4x_i + 1}{4}}{ctg^2(3x_i + e^{3a})}$	a=1,42
15	$Y_i = \frac{\sin^3 \frac{3x_i + 1}{2}}{tg^2 5x_i + e^{3a}}$	a=1,72
16	$Y_i = \frac{\sin^3(x_i + a) - \cos^2(x_i + a)}{(x_i + a)^4}$	a=1,19
17	$Y_i = \frac{arcctg\frac{x_i^3 + 1}{4}}{\cos^2 5x_i + e^{3a}}$	a=0,12
18	$Y_i = \frac{tg^3(x_i + a) - 5(\sin x_i + a)}{\sin^3(x_i + a)^4}$	a=2,12

Задание 11.7. Самостоятельно разработать программу для выполнения следующих действий:

1. Задать массив **X** действительных чисел из 8 элементов непосредственным присваиванием в коде:

X[8]={3.9 2.5 3.6 6.2 5.0 3.3 2.7 4.6}.

2. Вычислить массив **Y** по заданной формуле согласно своего варианта (таблица 11.1) с использованием указателей.

3. Вычислить сумму элементов массива **Y**) с использованием указателей.

Массив **X** необходимо инициализировать непосредственно в программном коде. На экран вывести исходный массив **X**, массив **Y**, сумму элементов массива **Y** и число его элементов.

Перед вводом программного кода самостоятельно разработать и начертить в отчете по лабораторной работе блок-схему алгоритма решения данной задачи) с использованием указателей и записать формулу для расчета на языке С++.

Задание 11.8. Разработать программу сортировки одномерного массива, состоящего из 10 элементов вещественного типа двойной точности, с помощью указателей.

Программный код для решения данной задачи следующий:

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

```
#define N 10
int _tmain(int argc, _TCHAR* argv[])
{
double A[N]={3.9, 2.5, 3.6, 6.2, 5.0, 3.3, 2.7, 4.6, 9.3, 8.1};
double *pmin[N], *temp;
int i, j;
   for(i=0; i<N; i++)                    //Цикл для вывода элементов A[i]
   cout<<"A["<<i<<"]="<<A[i]<<endl;      //исходного массива
cout<<endl;
// Взятие адресов элементов исходного массива
//в предположении, что они образуют отсортированный массив
   for(i=0; i<N; ++i)
   pmin[i]=&A[i];

//Сортировка массива по убывани ю
   for(i=0; i<N-1; ++i)
      for(j=i+1; j<N; ++j)
      {
         if(*pmin[i]<*pmin[j])
         {
         temp=pmin[i];
         pmin[i]=pmin[j];
         pmin[j]=temp;
         }
      }
for(i=0; i<N; i++)                          //Цикл для вывода элементов
cout<<"Anew["<<i<<"]="<<*pmin[i]<<endl;     //исходного массива,
                                            //отсортированных по
убыванию
cout<<endl;
   for(i=0; i<N; i++)                  //Цикл для вывода исходного массива
   cout<<"A["<<i<<"]= "<<A[i] <<endl;
getch();
return 0;
}
```

В программе следует обратить внимание на то, что при сортировке производятся операции с адресами элементов исходного массива, т. е. с указателями, а в самом исходном массиве элементы не сортируются.

В результате выполнения программы получен следующий результат:

```
A[0]= 3.9
A[1]= 2.5
A[2]= 3.6
A[3]= 6.2
A[4]= 5
A[5]= 3.3
A[6]= 2.7
A[7]= 4.6
A[8]= 9.3
A[9]= 8.1

Anew[0]= 9.3
Anew[1]= 8.1
Anew[2]= 6.2
Anew[3]= 5
Anew[4]= 4.6
Anew[5]= 3.9
Anew[6]= 3.6
Anew[7]= 3.3
Anew[8]= 2.7
Anew[9]= 2.5

A[0]= 3.9
A[1]= 2.5
A[2]= 3.6
A[3]= 6.2
A[4]= 5
A[5]= 3.3
A[6]= 2.7
A[7]= 4.6
A[8]= 9.3
A[9]= 8.1
```

Очевидно, что при помощи адресации и указателей исходный массив отсортирован и выведен на экран правильно. Кроме этого, нижний массив в результатах показывает, что сортировка происходила на уровне адресов (указателей), а сами элементы отсортированы не были.

Задание 11.9. Разработайте программу сортировки массива из **Задания** 3 по возрастанию и выполните вывод отсортированного и исходного массивов в два параллельных столбца.

3. Доступ к элементам многомерных массивов через указатели

В многомерных массивах указатели содержат адреса элементов массива построчно. Рассмотрим пример двухмерного целочисленного массива **M** размера 3 × 5, т. е. состоящего из 3 строк и 5 столбцов, и определим указатель:

```
int M[3][4]= {{1,2,3,4},{–6,–7,–8,–9},{11,12,13,14}};
int *pM;
```

Элементы массива (по индексам) располагаются в ячейках памяти по строкам в следующем порядке:

M[0][0], M[0][1], M[0][2], M[0][3], M[1][0], M[1][1], M[1][2], M[1][4], M[2][0], M[2][1], M[2][2], M[2][3].

Сначала запоминается первая строка, затем – вторая, потом – третья. В данном случае двухмерный массив – это массив трех одномерных массивов, состоящих из 5 элементов.

Указатель содержит адреса элементов в порядке расположения их в памяти. Поэтому тождественны равенства:

pM == &M[0][0]; // 1-я строка, 1-й столбец;
pM + 1 == &M[0][1]; // 1-я строка, 2-й столбец;
pM + 2 == &M[0][2]; // 1-я строка, 3-й столбец;
pM + 3 == &M[0][3]; // 1-я строка, 4-й столбец;
pM + 4 == &M[1][0]; // 2-я строка, 1-й столбец;
pM + 5 == &M[1][1]; // 2-я строка, 2-й столбец;
pM + 6 == &M[1][2]; // 2-я строка, 3-й столбец;
pM + 7 == &M[1][3]; // 2-я строка, 4-й столбец;
pM + 8 == &M[2][0]; // 3-я строка, 1-й столбец;
pM + 9 == &M[2][1]; // 3-я строка, 2-й столбец;
pM + 10 == &M[2][2]; // 3-я строка, 3-й столбец;
pM + 11 == &M[2][3]; // 3-я строка, 4-й столбец.

Практически следует выполнить инициализацию указателя, взяв, например, адрес первого элемента матрицы:

pM=&M[0][0].

После этого обращение к элементам матрицы, можно производить через указатель:

M[i,j]=*(pM+i*n+j);

где **i** – номер строки заданной матрицы, **j** – номер столбца, **n** – число столбцов в матрице.

Задание 11.10. Разработать программу вывода на экран элементов целочислен-ной матрицы

A[2][2]= {{1,2,3,}{4,5,6}};

через указатели.

Сформировать вектор (одномерный массив) указателей элементов исходной матрицы, вывести его на экран. Здесь же вывести на экран элементы матрицы **A** как разыменованные элементы вектора указателей.

Программный код для решения данной задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
#define n 2                          //Инициализация константы n
#define m 3                          //Инициализация константы m

int _tmain(int argc, _TCHAR* argv[])
{
int A[n][m]= {1,2,3,4,5,6};
int i, j, k;                         //Объявление индексов
int *pA, *B[6];                      //Объявление указателя и вектора указ.
cout<<"Matrix A"<<endl;
```

```
    for(i=0; i<n; i++)                  //Внешний цикл вывода элементов
                                        //матрицы A
      {                                 //Начало составного оператора для
                                        //внешнего цикла for
      for(j=0; j<m; j++)                //Вложенный цикл для вывода элементов
                                        //исходной матрицы A
        cout<<A[i][j]<<"   ";           //Вывод элемента матрицы A
        cout<<endl;
      }                                 //Конец составного оператора для
                                        //внешнего цикла for
  cout<<endl;
  pA=&A[0][0];                          //Инициализация указателя 1-го элемента
  cout<<pA<<"    "<<*pA<<endl;          //Вывод указателя 1-го элемента и его
                                        //разыменованного значения
    for(i=0; i<n; i++)                  //Внешний цикл для определения
элементов
                                        //вектора указателей B
      for(j=0; j<m; j++)                //Внутренний цикл для элементо вектора B
        B[i*m+j]=pA+i*m+j;              //Определение указателей на элементы
                                        //матрицы A
  cout<<endl;
  cout<<"Vectors of pointers and of elements"<<endl;
    for(i=0; i<6; i++)                  //Цикл для вывода указателей и элементов
      cout<<B[i]<<"   "<<*B[i]<<endl;   //Вывод элемента вектора указателей
  cout<<endl;
  cout<<"Matrix A from pointers"<<endl;
    for(i=0; i<n; i++)                  //Внешний цикл вывода элементов
                                        //матрицы A через указатель
      {                                 //Начало составного оператора для
                                        //внешнего цикла for
      for(j=0; j<m; j++)                //Вложенный цикл для вывода элементов
                                        //матрицы A
        cout<<*(pA+i*m+j)<<"   ";       //Вывод элемента исходной матрицы A
                                        //через указатель
        cout<<endl;
      }
  getch();
  return 0;
  }
```

В результате выполнения программы получен следующий результат:

```
Matrix A
1    2    3
4    5    6

002AFBF0      1

Vectors of pointers and of elements
002AFBF0    1
002AFBF4    2
002AFBF8    3
002AFBFC    4
002AFC00    5
002AFC04    6

Matrix A from pointers
1    2    3
4    5    6
```

Задание 11.11. Разработать программу для вывода на экран значений элементов целочисленной матрицы **M[3][4]** с учетом вышеприведенных зависимостей (с. 10) с использованием указателей. В начале программы выведите на экран исходную матрицу **M**.

Задание 11.12. Разработать программу умножения целочисленных матриц с использованием указателей.

Условием перемножения двух матриц **A** и **B** является равенство числа столбцов матрицы **A** и числа строк матрицы **B**. Если первая матрица **A** имеет размер $\mathbf{n} \times \mathbf{k}$, то вторая матрица **B** должна иметь размер $\mathbf{k} \times \mathbf{m}$. В результате перемножения получим матрицу **C** размера $\mathbf{n} \times \mathbf{m}$.

Поэлементное перемножение двух матриц в стандартной математической форме имеет вид

$$C_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, m.$$

С учетом синтаксиса формирования массивов в языке **C** индексация должна начинаться с нуля, поэтому формулу перепишем в следующем виде:

$$C_{ij} = \sum_{k=0}^{n-1} a_{ik} b_{kj}, \quad i = 0, 1, 2, \ldots, n-1, \quad j = 0, 1, 2, \ldots, m-1.$$

Программный код для решения данной задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include <iostream>
using namespace std;
#include <stdio.h>
#include <conio.h>
#define n 2                                    //Инициализация константы n
#define k 3
```

```
#define m 4
int _tmain(int argc, _TCHAR* argv[])
{
   int i, j, q;
   int A[n*k], B[k*m];
   long int *C[n*m];
            //Заполнение матриц целыми числами
for (i = 0; i < n; i++)
   for (j = 0; j < k; j++)
      A[i*k + j] = i + j-7 ;
   for (i = 0; i < n; i++)
   {
      for (j = 0; j < k; j++)
         cout<<"   "<<A[i,j];
         cout<<endl;
   }
            //Формирование вектора указателей B
   for (i = 0; i < k; ++i)
      for (j = 0; j < m; ++j)
         B[i*m + j] =  i + j-12;
            //Циклы перемножения с накоплением суммы
for (i = 0; i < n; ++i)
   for (j = 0; j < m; ++j)
   {
      C[i*m + j] = 0;
      for (q = 0; q < k; ++q)          //Цикл для накопления суммы при
вычислении
                                       //элемента Bij

            C[i*m + j] += (A[i*k + q] * B[q*m + j]);
   }
printf("\n\t The result of multiplying the two matrices:\n\t\t     C = A*B (%dx%d
= %dx%d * %dx%d)\n", n, m, n, k, k, m);

   for (i = 0; i < n; ++i)
   {
      printf("\n\t");
      for (j = 0; j < m; ++j)
         printf("%6ld", C[i*m + j]);
   }
//cout<<"A["<<i<<"]="<<A[i]<<endl;
//cout<<endl;
getch();
return 0;
}
```

Для вывода результата перемножения предусмотрен спецификатор **ld**.

Результат выполнения программы показан ниже.

```
-7   -6   -5
-7   -6   -5

      The result of multiplying the two matrices:
                C = A*B (2x4 = 2x3 * 3x4)

        800   728   656   584
        668   608   548   488_
```

Очевидно, что определено произведение двух матриц при помощи указателей на их элементы.

Задание 11.13. Разработать программу сложения целочисленных матриц с использованием указателей. За основу взять программы из **Заданий 3.11** и **3.12**. Осуществить максимально информативный вывод результатов.

Матрица **A[2][2]= {{1,2,3,}{4,5,6}};**

Матрица **B[2][2]= {{7,8,9,}{10,11,12}};**

Задание 3.14. Разработать программу на базе программы из **Задания 3.12** Промежуточный размер **k** примите за 5X, где X – номер компьютера, на котором выполняется лабораторная работа. Предусмотрите корректный вывод результата перемножения двух матриц.

Вывод результата перемножения выполните на основе операции разыменования.

Добавьте программу умножения матриц с обычной индексацией элементов массивов, т. е. без использования указателей.

Включите при выводе нумерацию строк и столбцов (слева от матрицы и вверху над ней).

3. Указатели на массивы символьных переменных

Рассмотрим инициализацию указателей типа **char**:

char *ptr = "hello, world".

Переменная ***ptr** является указателем, а не массивом. Поэтому строковая константа **"hello, world"** не может храниться в нем. Тогда возникает вопрос, где она хранится?

Когда компилятор встречает строковую константу, он создает так называемую **таблицу строк**, где сохраняет строковые константы, которые попадаются ему по ходу чтения текста программы. Следовательно, когда

встречается объявление с инициализацией, то компилятор сохраняет **"hello, world"** в таблице строк, а указатель ***ptr** записывает ее адрес.

Задание 11.15. Разработать программу считывания строк разной длины с использованием указателей и операций с ними.

Программный код для решения задания следующий:

```
#include "stdafx.h"
#include "stdio.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, n;
//Инициализация символьного
char *ptr[ ] = {"one", "two", "three", "four", "five",\     //Инициализация
"six", "seven", "eight", "nine", "ten"};        //символьного массива
n = sizeof(ptr)/sizeof(ptr[0]);                 //Определение числа
                                //адресов элементов
printf("\n\t Strings of various length:\n");
for (i = 0; i < n; ++i)
printf("\n%12d) %s", i+1, ptr[i]);
getch();
return 0;
  }
```

В программе использован одномерный массив указателей. Функция **printf()** и спецификатор преобразования **%s** допускают применение в качестве параметра указателя на строку. При этом на дисплей выводится не значение указателя, а содержимое адресуемой им строки.

Обратный слэш (****) служит для переноса содержимого операторной строки на новую строку.

Оператор **sizeof()** вычисляется во время компиляции программы, превращаясь обычно в целую константу, значение которой равно размеру типа или объекта, в данном случае соответствует размеру массива указателей.

Следует обратить внимание на инициализацию массива указателей. Содержимое, заключенное в фигурные скобки, представляет собой строки, для каждой из которых служит указатель, входящий в массив указателей.

Результат выполнения программы следующий:

```
Strings of various length:

  1) one
  2) two
  3) three
  4) four
  5) five
  6) six
  7) seven
  8) eight
  9) nine
 10) ten
```

Задание 11.16. Разработать программу поиска подстроки в строке, сформированной по случайному закону из 15 букв латинского алфавита, с помощью функции **strstr()**. В качестве подстроки принять первые три буквы своей фамилии.

Для работы с функцией **strstr()** требуется подключение заголовочного файла **string.h**.

Программный код для решения задания следующий:

```
#include "stdio.h"
#include <conio.h>
#include "iostream"
#include <string.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
   int i, k = 0, n, in;
   int numSTR;
   int N = 1000;
   int numA, numZ;
   char str[16];
   char sub[4];
   char *ptr;
      srand((unsigned)time(NULL));
   numA = (int)'a';
   numZ = (int)'z';

   printf("\n Enter the three letters: ");
   in = scanf_s("%s", sub, sizeof(sub));
      if (in == 0)
      {
         printf("\n Error input. Press any key: ");
      getch();
```

```
    exit(1);
    }
  printf("\n\t substring is \"%s\"\n", sub);
  for (n = 0; n < N; n++)
  {
    for (i = 0; i < 15; i++)
      str[i] = numA + rand() % (numZ - numA) + 1;
  str[i] = '\0'; // завершение строки нулевым символом
  ptr = strstr(str, sub);
    if (ptr != NULL)
    {
    numSTR = (int)(ptr - str +1);
    k++;
    break;
    }
  }
   if (k == 0)
    printf("\n\t Substring \"%s\" not found", sub);
    else
  printf("\n\t Substring \"%s\" found at positions %d, %d, %d", \
  sub, numSTR, numSTR+1, numSTR+2);
  puts("\n");
    for (i = 0; i < 15; i++)
      printf(" %3d)", i+1);
  puts("");
    for (i = 0; i < 15; i++)
      printf(" %3c ", str[i]);
  printf("\n\n ... Press any key: ");
  getch();
  return 0;
}
```

Результат выполнения программы следующий:

```
Enter the three letters: s d f

        substring is "s"

        Substring "s" found at positions 4, 5, 6

  1)   2)   3)   4)   5)   6)   7)   8)   9)  10)  11)  12)  13)  14)  15)
  d    t    d    s    r    r    u    p    i    f    k    x    f    g    s
```

Контрольные вопросы

1. Как рассматривает имя массива компилятор языка **C**?
2. Как инициализируется указатель **pA** адресом массива **A**?
3. На какое место в памяти компьютера указывает имя массива?
4. Какая связь между указателями и массивами в языке **C**?
5. Как формируется массив указателей в языке **C**?
6. Запишите выражение для прямой адресации массива.
7. На какое место в памяти компьютера указывает имя массива без индекса?
8. Если имя массива **A**, то какое выражение для i-го элемента по правилам работы с указателями с операцией разыменования?
9. Запишите оператор для вывода на экран i-го элемента вектора **A** при помощи указателя.
10. Запишите выражение для того, чтобы указатель ***ptr** ссылался на элемент вектора, находящийся на расстоянии **n** от того, на который ранее ссылался указатель.
11. Запишите два выражения для обращения ко второму элементу вектора **A.**

Вывод элементов вектора с использованием указателей.

12. Запишите оператор для вывода на экран i-го элемента вектора **A** при помощи инкремента указателя.
13. Как следует организовать посимвольное формирование строки символов с помощью указателя?
14. В каком порядке указатели содержат адреса элементов многомерных массивов?
15. Запишите выражение для обращения к элементам матрицы через указатели.

Лабораторная работа № 12 (2 занятия)

УКАЗАТЕЛИ И ФУНКЦИИ В СРЕДЕ VISUAL C++ 2010

Цель работы: изучить программирования функций, аргументами которых могут быть указатели, а также функций, возвращающих значения через указатели, в среде Visual C++ 2010.

1. Назначение и объявление функций в среде Visual C++ 2010

Как правило, основную часть программного кода в C++ составляют функции. Функции позволяют разбивать программу на отдельные автономные блоки. Любая программа содержит, по крайней мере, одну функцию (главную) - **tmain()**.

Для создания правильного кода компилятору необходимо сообщить в начале программы имя функции, тип возвращаемого результата, а также количество и типы аргументов. Для этой цели в C++ используется так называемый **прототип функции.**

Прототип функции задается следующим образом:

ТипРезультата ИмяФункции (ТипПараметра1 [ИмяПараметра1], …);

Использование прототипа функции является **объявлением функции**. Чаще всего прототип функции совпадает с заглавием функции. В отличие от заглавия функции в ее описании, прототип заканчивается (!) **точкой с запятой.**

Имена формальных параметров функции при ее объявлении не играют роли. Поэтому прототип функции может выглядеть следующим образом:

int function(int a, float b, float c)

или

int function(int, float, float).

Два этих объявления функции **function** равносильны.

2. Описание и вызов функций в Visual C++ 2010

Описание или **программный код** функции находится в программе после основной функции и имеет следующий вид:

```
Тип ИмяФункции (ТипПараметра1 ИмяПараметра1, …)
{
Тело функции
}
```

В заглавии **Тип** перед именем функции определяет тип значения, которое возвращает функция. Если тип не указан, то по умолчанию предусматривается, что функция возвращает целое значение (тип **int**).

Список параметров состоит из перечня типов и имен параметров, разделенных запятыми. Функция может не иметь параметров, но круглые скобки необходимы всегда.

В списке параметров для каждого параметра должен быть указан тип. Например,

function (int x, int в, float z) - правильный список параметров;
function (int x, в, float z) - неправильный список параметров.

В теле функции обязательно должен присутствовать оператор **return** с параметром того же типа, что и возвращаемое значение.

Оператор **return** имеет два варианта использования.

1. Вызывает немедленный выход из функции и возвращение в программу, которая ее вызвала.

2. Используется для возвращения значения функции.

Если возвращаемое значение не используется в дальнейшем в программе, то оператор **return** следует без параметра или **вообще может быть опущен**. В этом случае возвращение в программу осуществляется после достижения закрывающейся скобки "**}**".

В случае, когда оператора **return** в теле функции нет или за ним нет значения, то значение, возвращаемое функцией, неизвестно (не определено). Если функция должна возвращать значение, но не делает этого, компилятор выдает предупреждение. Все функции, которые возвращают значение, могут использоваться в выражениях языка С++.

Функция может вызывать другие функции (одну или несколько). А те, в свою очередь, проводить вызов третьих и т.д. Кроме того, функция может вызывать саму себя. Это явление в программировании называется **рекурсией.**

Для того чтобы функция выполняла определенные действия в программе, она должна быть вызвана. Функция выполняется только при обращении к ней. По окончании работы функция возвращает в основную программу в качестве результата значение некоторой переменной и т. п.

Вызов функции осуществляется путем указания в программе ее имени (идентификатора), за которым в круглых скобках следует список аргументов, разделенных запятыми:

ИмяФункции(аргумент 1, аргумент 2, … аргумент N) .

Каждый аргумент функции является **переменной, выражением или константой**. Они передаются в тело функции для последующего использования в вычислительном процессе. Список аргументов может быть пустым.

3. Использование указателей в качестве параметров при вызове функций

Использование указателей в качестве аргументов вызываемой функции позволяет обращаться к объектам в вызвавшей ее функции (главной функции), в том числе модифицировать их.

Задание 12.1. Разработать программный код функции **change()**, в задачу которой входит обмен двух переменных **a** и **b** местами. Для решения такой задачи необходимо передать из вызывающей программы (например, из главной функции **main()**) в функцию **change()** указатели на переменные, которые нужно изменить.

Программный код решения задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int change(int*, int*);                    //Объявление (прототип) ф-ии change()
int _tmain(int argc, _TCHAR* argv[])
{
int a = 10, b = -20;
cout<<"\n  a = "<<a<<"   b = "<<b;         //Вывод на экран значений перем-х
   change(&a, &b);                         // Вызов функции change() с адресами a
и
                                           //b в качестве фактических параметров
cout<<"\n\n   After change \n";
cout<<"  a = "<<a<<"   b = "<<b;           // Результат после обращения функции
change()
getch();
return 0;
}
int change(int *pa, int *pb)               //Описание функции change()
{
int temp;
   temp = *pa;
   *pa = *pb;
   *pb = temp;
return *pa, *pb;                           //Возврат в основную программу новых
}                                          //значений *pa и *pb
```

В программе в качестве фактических параметров функции **change()** выступают адреса заданных переменных. Можно было в главной функции определить указатели и инициализировать их адресами заданных переменных, а потом передать эти указатели в функцию **change()**.

Результат выполнения программы следующий:

```
a = 10    b = -20

 After change
a = -20    b = 10
```

Очевидно, что при помощи указателей был произведен обмен значений исходных переменных и результат передан в основную программу.

Задание 12.2. Разработать программный код на базе программы из **Задания 12.1** для вывода на экран адреса переменных **a** и **b** до и после вызова функции **change()**.

Проанализируйте полученные результаты письменно в отчете.

Указатели, передаваемые в функцию, могут ссылаться на указатели, обозначать начало какого-либо массива и т.д. Кроме этого, они могут применяться для защиты массивов, над которыми необходимо произвести некоторые вычисления или преобразования. Особым свойством указателей можно считать возможность использования их в качестве возвращаемых значений функций. Поскольку функции возвращают только одно значение, то несколько значений одного типа можно поместить в массив, а затем указатель на этот массив рассматривать в качестве возвращаемого значения.

Общая форма определения функции, которая возвращает указатель, следующая:

```
тип *имя_функции ( аргументы функции )
{
тип *имя_указателя;
…..
return  имя_указателя;
}
```

Задание 12.3. Разработать программный код функции для сложения двух одномерных массивов (векторов) и возвращения результата в основную программу через указатель.

Программный код решения задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int *sum(int A[], int B[], int);
int _tmain(int argc, _TCHAR* argv[])
{
int i, n;
int A[] = {1,2,3,4,5};
```

```
int B[] = {2,2,2,2,2};
int *sumAB;
n = (sizeof(A)/sizeof(A[0]));                  //Определение размера матрицы A
cout<<"\n  Matrix A \n";
   for (i = 0; i < n; i++)                     //Цикл для вывода матрицы  A
      cout<<"   "<<A[i];
cout<<"\n  Matrix B \n";
   for (i = 0; i < n; i++)                     //Цикл для вывода матрицы  B

      cout<<"   "<<B[i];
sumAB = sum(A, B, n);                          //Вызов функции суммирования
матриц
puts("\n\n Result from function: ");
   for (i = 0; i < n; i++)                     //Цикл для вывода матрицы-результата

      cout<<"   "<<sumAB[i];
getch();
return 0;
}

int *sum(int A[], int B[], int n)              //Описание ф-ии суммирования
матриц
{
int i;
int *C= (int *)calloc(n, sizeof(int));         //Выделение памяти для матрицы C
   for (i = 0; i < n; i++)                     //Цикл для суммирования элементов
      C[i] = A[i] + B[i];
return ptr;
}
```

В программном коде для функции **sum()** суммирования двух матриц через указатели вводится понятие функции **calloc()** для выделения блока памяти для массива (вектора).

Ее прототип записывается как

```
void* calloc(number, size);
```

где **number** – количество элементов массива, под который выделяется память, а **size** – размер одного элемента в байтах.

В результате обращения к этой функции вида

int *C= (int *)calloc(n, sizeof(int)),

для целого массива **C** выделяется блок памяти размером **number*size** байт, причём весь блок заполнен нулями.

Возвращаемое значение – это указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда **`void`***, поэтому этот тип данных может быть приведен к желаемому типом данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Функция **`calloc`** выделяет столько динамической памяти, сколько необходимо программе, не больше и не меньше.

Результат выполнения программы следующий:

```
Matrix A
 1   2   3   4   5
Matrix B
 2   2   2   2   2

Result from function:
 3   4   5   6   7_
```

Очевидно, что при помощи указателей было произведено правильное сложение исходных матриц и результат передан в основную программу.

Задание 12.4. Разработать программный код на базе программы из **Задания 12.3**, где для более полного понимания действий с указателями вывести на экран адреса элементов исходных массивов и нового массива до и после вызова функции **change()**.

Добавьте также оператор вывода результатов непосредственно в программный код функции **sum()** в оператор цикла для полного анализа данной программы.

Проанализируйте полученные результаты письменно в отчете.

4. Особенности возврата указателей в основную программу

Отметим, что никогда не следует возвращать адрес переменной, определенной в теле функции, так как переменные функции являются локальными и существуют только во время работы функции. Чтобы вернуть указатель, функция должна объявить его тип в качестве типа возвращаемого значения.

Таким образом, если функция возвращает указатель, то значение, используемое в ее инструкции **return**, также должно быть указателем. В частности, многие библиотечные функции, предназначенные для обработки строк, возвращают указатели на символы.

5. Указатель на функцию

В языке **C** существует такой механизм, как **указатель на функцию.** Допустим, существует несколько функций для различных операций с данными. В этом случае оказывается удобным определить указатель на функцию и использовать его там, где требуется производить расчет для различных функций.

Указатель на функцию – это переменная, содержащая в памяти адрес, по которому расположена функция.

Имя функции – это адрес начала ее программного кода. Указатели на функции могут передаваться функциям в качестве аргументов, возвращаться функциями, сохраняться в массивах и присваиваться другим указателям на функции.

Типичное определение указателя на функцию следующее:

тип_возвращаемый_функцией(*имя_указателя_на_функцию)(аргументы);

В круглых скобках определяется указатель на функцию, которая возвращает тот или иной тип – **тип_возвращаемый_функцией**. Хотя знак ***** обозначает префиксную операцию, он имеет более низкий приоритет, чем функции, заключенные в круглые скобки, поэтому для правильного комбинирования частей объявления необходимы дополнительные скобки. При этом **аргументы** – это аргументы той или иной функции с заданным типом возвращаемого значения, для которой определяется указатель ***имя_указателя-_на_функцию**. Очевидно, что возможны сложные объявления функций.

Указатели на функции часто используются в системах, управляемых меню. Пользователь выбирает команду меню (одну из нескольких), обслуживающуюся своей функцией. Указатели на каждую функцию находятся в массиве указателей. Выбор пользователя служит индексом, по которому из массива выбирается указатель на нужную функцию.

Часто указатели на функции применяются при реализации обобщенных алгоритмов, например алгоритмов сортировки и поиска. В этом случае критерии сортировки и поиска приобретают вид отдельных функций и передаются при помощи указателей на функции в качестве параметра реализации основного алгоритма.

Задание 12.5. Разработать программный код функции с пузырьковой сортировкой целого числового массива по возрастанию с вызовом такой функции по ссылке.

Вызов по ссылке означает, что в качестве фактических параметров функций будут использоваться адреса переменных, и в этом случае прототип таких функций будет содержать указатели на соответствующие типы.

Программный код решения задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <conio.h>
using namespace std;
int sortmax(int*, const int);
```

```
int _tmain(int argc, _TCHAR* argv[])
{
int A[]={56, 34, 2, 0, 1, -21, 6, 8, 7};
int i, n;
n=sizeof(A)/sizeof(A[0]);                    //Размерность исходного массива
cout<<"\n Data items in original order:";
   for(i=0; i<n; i++)                        //Цикл для вывода исходного массива
      cout<<"   "<< A[i];
sortmax(A, n);                               //Вызов функции сортировки
cout<<"\n\n Data items in ascending order:";
   for(i=0; i<n; i++)
      cout<<"   "<< A[i];
getch();
return 0;
}

int change(int *pa, int *pb)                 //Описание функции  change
{
int temp;
temp=*pa;
*pa=*pb;
*pb=temp;
return 0;
}

int sortmax(int *vektor, const int size)// Описание функции  sortmax
{
int i, j;                                    //Счетчик проходов и счетчик сравнений

   for(i=0; i<size-1; i++)                   /Цикл для контроля проходов
      for(j=0; j<size-1; j++)                //Цикл для контроля сравнений на i-ом
                                             //проходе
         if(vektor[j]> vektor[j+1])          //Обмен значений переменных при
                                             //нарушении порядка возрастания
         change(&vektor[j], &vektor[j+1]);
return 0;
}
```

В функции сортировки **sortmax()** в качестве формального параметра используется константный указатель, который указывает на первый элемент заданного массива. Второй формальный параметр также константный, так подчеркивается его неизменность в теле функции **sortmax()**.

Передача функции размера массива в качестве параметра имеет два преимущества: это хороший стиль программирования и, кроме того, такую функцию можно использовать многократно.

Результат выполнения программы следующий:

```
Data items in original order:   56   34   2   0   1   -21   6   8   7
Data items in ascending order:   -21   0   1   2   6   7   8   34   56,
```

Очевидно, что в результате выполнения сортировки с использованием указателей на функцию элементы исходного массива выстроены по возрастанию.

Задание 12.6. Разработать программный код на базе программы из **Задания 12.5**, где для более полного понимания действий с указателями вывести на экран адреса элементов исходного и отсортированного массивов.

Проанализируйте, на каком этапе происходит изменение значений элементов массива и опишите полученные результаты письменно в отчете.

Задание 12.7. Разработать программный код функции с пузырьковой сортировкой целого числового массива по убыванию с вызовом такой функции по ссылке, т.е. через указатели. Предусмотрите вывод на консоль исходного массива, потом отсортированного нового массива после вызова функции сортировки, и снова – для контроля – исходного массива. При этом аргументы функции **sortmax()** оставьте без изменения.

Задание 12.8. Разработать программный код функции по обработке вектора результатов измерений: расчету среднего значения (среднего арифметического) одномерного числового массива, его дисперсии и среднего квадратичного отклонения (стандартного отклонения). Эти значения должны быть выведены на консоль в главной функции программы **main()**.

Приведем расчетные формулы.

Среднее выборочное значение $\bar{a}$ элементов одномерного массива размерностью **N**

$$\bar{a} = \frac{1}{N}\sum_{i=1}^{N} a_i,$$

где a_i – элементы массива.

Дисперсия **D** одномерного массива показывает квадрат среднего отклонения значений элементов массива от их среднего значения:

$$D = \frac{1}{N-1}\sum_{i=1}^{N} (a_i - \bar{a})^2.$$

Среднеквадратичное отклонение есть плюс корень квадратный из дисперсии.

Программный код решения задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <conio.h>
#include <stdlib.h>
//#include <math.h>
using namespace std;
double *statistic(int arr[], int n);                    //Прототип функции для
расчета
                                                        //статистических показателей
int _tmain(int argc, _TCHAR* argv[])
 {
int A[] = {2, -3, 5, 6, 7, 8, 9,-1};
int i, n;
double *R;
n = sizeof(A)/sizeof(A[0]);                             //Определение размера вектора
cout<<"\n Data vektor:";
   for (i = 0; i < n; i++)
      cout<<"   "<<A[i];
R=statistic(A, n);                                      //Вызов функции

          //Вывод расчетных характеристик массива
cout<<"\n\n The average value of an array: "<<R[0];
cout<<"\n The dispersion of the array: "<<R[1];
cout<<"\n The standard deviation of the array: "<<R[2];
free(R);                                                //Освобождение памяти
getch();
return 0;
}

double *statistic(int arr[], int N)                     //Описание функции
{
int j;
double *vektor=(double*)calloc(N, sizeof(double));   //Выделение памяти для
                                                        //вектора расчетов vektor
double D, S,SR;
SR = D = 0.0;
   for (j = 0; j < N; j++)                              //Цикл для расчета среднего
   SR= SR+arr[j];                                       // арифметического
SR=SR/N;                                                //Расчет среднего
арифметического
```

```
    for (j = 0; j < N; j++)                    //Цикл для расчета дисперсии
    D=D+(arr[j]-SR)*(arr[j]-SR);
D=D/(N-1);                                     //Расчет дисперсии
S=sqrt(D);                                     //Среднеквадратическое откл-ние
vektor[0]=SR;                                  //Вектор результатов расчетов
vektor[1]=D;
vektor[2]=S;
return vektor;
}
```

Расчетные характеристики одномерного массива размещаются последовательно друг за другом в выделенной памяти для указателя **vektor**. Сформированный указатель функция возвращает в точку вызова функции **statistic()**.

Результат выполнения программы следующий:

```
Data vektor:   2   -3   5   6   7   8   9   -1

The average value of an array: 4.125
The dispersion of the array: 18.9821
The standard deviation of the array: 4.35685_
```

Задание 12.9. Разработать программный код на базе программы для **Задания 12.8**, где в качестве первого аргумента **statistic()** используйте указатель на числовой массив.

Воспользуйтесь справкой по математическим функциям и в программе примените функцию, которая осуществляет возведение в степень.

Дополните возврат функцией **statistic()** исходного массива, поэлементно возведенного в квадрат. В главной функции **main()** результаты выведите на консоль.

Задание 12.10. Разработать программный код для построения на экране дисплея графика функции

$$y = \sin(3x)\, e^{-x/3}.$$

Предусмотреть возможность записи в текстовый файл графика данной функции.

Для решения примера используем средства вывода на печать форматированных данных без применения специальных графических библиотек.

Внимание!!! Студенты самостоятельно, в том числе с использованием сети Интернет, должны разобраться с назаначением каждого оператора программы и записать это в комментариях программного кода, что должно найти свое отражение в файле **Результат ЛР4**.

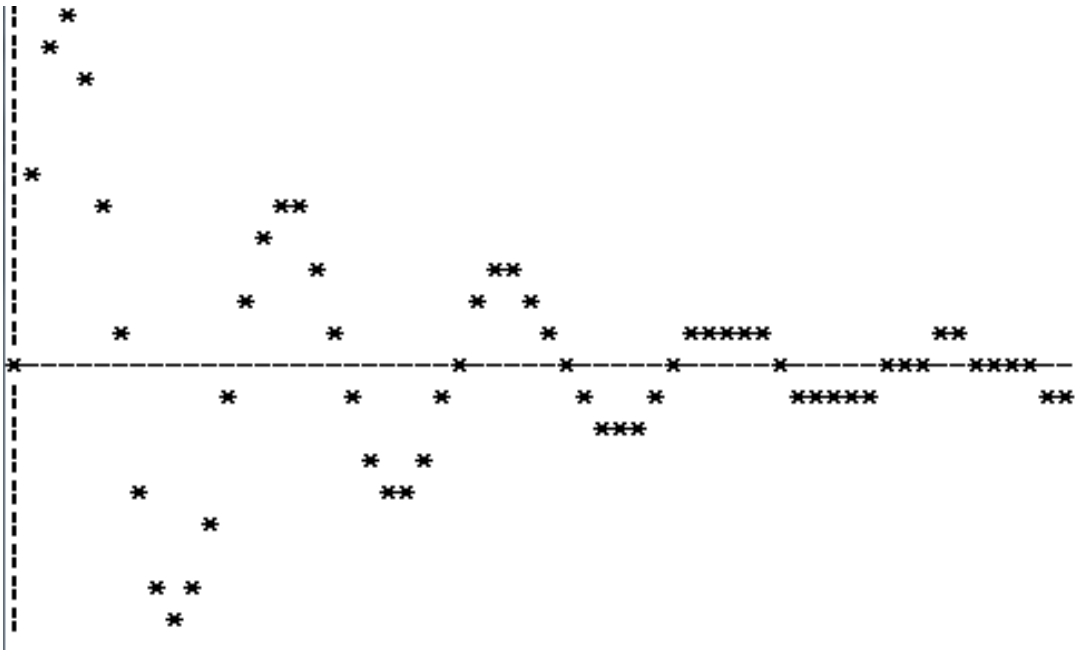
Кроме этого, необходимо подобрать параметры **SCREENW** и **SCREENH**, чтобы график занял всю видимую область окна **DOS**.

Программный код решения задачи следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <conio.h>
#include <stdlib.h>
#include <stdio.h>
using namespace std;
                    //Размеры диаграммы по ширине и высоте экрана
#define SCREENW  60
#define SCREENH  20
double f (double x);
void plot (FILE *fout, double a, double b, double (*f) (double));
int _tmain(int argc, _TCHAR* argv[])
{
plot (stdout, 0.0, 10.0, f);          //Обращение к функции вывода графика в
                                      //стандартный поток (консоль)
getch();
return 0;
}
        //Функция построения графика заданной функции
void plot (FILE *fout, double a, double b, double (*f) (double))
{
        //Формальные параметры функции plot
        //FILE *fout – указатель на поток вывода
        //double a – левая граница оси абсцисс
        //double b – правая граница оси абсцисс
        //double (*f) (double) – указатель на функцию
char screen[SCREENW][SCREENH];
double x, y[SCREENW];
double hx, hy, ymin = 0, ymax = 0;
int i, j, xz, yz;                          // hx – шаг по оси абсцисс
hx = (b - a) / (SCREENW - 1);
   for (i = 0, x = a; i < SCREENW; ++i, x += hx)
   {
   y[i]=f(x);                              //Вызов вычисляемого значения функции
      if (y[i] < ymin) ymin = y[i];        //Определение минимального и
         if (y[i] > ymax) ymax = y[i];     //максимальное значения ординаты
   }
hy = (ymax - ymin) / (SCREENH - 1);
yz = (int)floor (ymax / hy + 0.5);
xz = (int)floor (-a / hx + 0.5);
        //Рисование осей координат
   for (j = 0; j < SCREENH; ++j)
   {
      for (i = 0; i < SCREENW; ++i)
```

```
      {
      if (j == yz && i == xz)
         screen[i][j] = '+';                    //Символы для осей  '.', '|', '+'
      else if (j == yz)
         screen[i][j] = '-';
      else if (i == xz)
         screen[i][j] = '|';
      else
         screen[i][j] = ' ';
      }
}
         //Рисование графика функции
   for (i = 0; i < SCREENW; ++i)
   {
      j = (int)floor ((ymax - y[i]) / hy + 0.5);
      screen[i][j] = '*';                       //Ссимвол начертания графика функции
   }
         // Вывод результата в файл или в стандартный поток stdout
   for (j = 0; j < SCREENH; ++j)
   {
      for (i = 0; i < SCREENW; ++i)
      cout<<screen[i][j];
      cout<<endl;
   }
}
double f (double x)
{
return sin (3.0*x) * exp (-x / 3.0);        //Описание функции для вычисления y
}
```

Результат выполнения программы следующий:



Контрольные вопросы

1. Каким образом можно вернуть из функции несколько значений?
2. Каким образом определяется тип функции?
3. Как выглядит описание функции, которая возвращает указатель на заданный тип, например **char**?
4. Можно ли использовать многоуровневую адресацию для функции, которая возвращает указатель на заданный тип? Если можно, то как происходит определение такой функции?
5. В каком месте программы можно определить указатель на функцию?
6. Имеет ли указатель на функцию прототип и определение?
7. Как осуществляется вызов функции с помощью указателя?
8. Как взаимосвязаны между собой объявление функции, ее определение и вызов?

Лабораторная работа № 13 (2 занятия)

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ В ЯЗЫКЕ С++

Цель работы: рассматриваются вопросы динамического распределения памяти, изучаются его функции и их применение для числовых и символьных массивов, хранения данных.

1. Динамическая память – основные понятия

Динамическая память – это оперативная память компьютера, предоставляемая программе при ее работе.

Динамическое размещение данных означает использование этой памяти при работе программы. Статическое размещение, например, явное определение массива переменных заданного типа осуществляется компилятором в процессе компиляции (запуска) программы.

Для динамического выделения памяти компьютера при хранении данных используются указатели. **Динамическое распределение** означает, что программа подготавливает память во время ее выполнения.

Память, выделяемая в **C++** функциями динамического распределения данных, находится в так называемой **динамически распределяемой области памяти** (англ. **heap – «куча»).**

Динамически распределяемая область памяти – это свободная область памяти, не используемая программой, операционной системой или другими программами. Ее размер заранее неизвестен, но, как правило, в ней достаточно места для размещения данных программы. Хотя размер динамически распределяемой области памяти очень большой, все же она конечна и может быть исчерпана.

Основу системы динамического распределения памяти в **C++** составляют библиотечные функции **calloc()**, **malloc()**, **realloc()** и **free()**.

2. Библиотечные функции динамического распределения памяти в C++

2. Функция calloc()

Прототип функции с необходимой подключаемой библиотекой имеет вид

```
#include <stdlib.h>
void *calloc(size_t num, size_t size);
```

Функция **calloc()** выделяет память, размер которой равен значению выражения **num×size**, т. е. память, достаточную для размещения массива, содержащего **num** объектов размером **size**. Выделенная область памяти обнуляется.

Функция **calloc()** возвращает указатель на первый байт выделенной области памяти для массива **num** объектов, каждый из которых имеет размер **size** или **NULL**, если запрос на память выполнить нельзя. Если для удовлетворения запроса нет достаточного объема памяти, возвращается нулевой указатель. Перед попыткой использовать распределенную память важно проверить, что возвращаемое значение не равно нулю. Тип **void** может быть переопределен для требуемого типа, т. е. для **char**, **int**, **float**, **double**.

Пример фрагмента программного кода динамического распределения памяти для действительных массивов (например) заданного размера:

```
double *ptr;
ptr=(double*)(calloc(10, sizeof(double)));
   if(!ptr)                                  //Условие логического отрицания
   {
   cout<<"  Out of memory. Press any key:\n");
   getch();
   exit(1);
   }
```

Здесь число **10** – размер одномерного массива с действительными данными (типа **double**). В случае выделения памяти для двухмерного массива размера **NM** строчка с функцией **calloc()** перепишется так:

```
ptr=(double*)(calloc(N*M, sizeof(double)));
```

При этом двухмерный массив рассматривается как аналог одномерного массива размера **N * M**. Использование явного приведения типов (**double**) сделано для того, чтобы гарантировать переносимость программы, в первую очередь для обеспечения совместимости с языком программирования **C++**.

3. Функция malloc()

Прототип функции с необходимой библиотекой имеет вид

```
#include <stdlib.h>
void *malloc(size_t size);
```

Функция **malloc()** возвращает указатель на первый байт области памяти размера **size**, которая была выделена из динамически распределяемой области памяти. Если для удовлетворения запроса в динамически распределяемой области памяти нет достаточного объема, возвращается нулевой указатель **NULL**. При этом следует иметь в виду, что попытка использовать нулевой указатель обычно приводит к полному отказу системы. Выделенная область памяти не инициализируется.

Приведем фрагмент программного кода динамического распределения памяти для массивов заданного размера:

```
double *ptr;
ptr=(double*)(malloc(10*sizeof(double)));
   if (!ptr)                                  //Условие логического отрицания
   {                                          //Выход за пределы памяти
   cout<<"  Out of memory. Press any key: ";
   getch();
   exit(1);
   }
```

4. Функция realloc()

Прототип функции с необходимой библиотекой имеет вид

```
#include <stdlib.h>
void *realloc(void *ptr, size_t size);
```

Функция **realloc()** изменяет размер блока ранее выделенной памяти, адресуемой указателем ***ptr** в соответствии с заданным размером **size**. Значение параметра **size** может быть больше или меньше, чем перераспределяемая область. Функция **realloc()** возвращает указатель на блок памяти, поскольку не исключена необходимость перемещения этого блока. В таком случае содержимое прежнего блока (до **size** байтов) копируется в новый блок. Если новый размер памяти больше старого, дополнительное пространство не инициализируется.

Если запрос невыполним, то функция распределения памяти **realloc()** возвращает нулевой указатель **NULL**. Данная функция позволяет перераспределить ранее выделенную память. При этом новый размер массива может быть как меньше, так и больше предыдущего. Если система выделит память в новом месте, то все значения, к которым программа обращалась по указателю ***ptr**, будут переписаны на новое место автоматически.

5. Функция free()

Прототип функции с необходимой библиотекой имеет вид

```
#include <stdlib.h>
void free(void *ptr);
```

Функция **free()** возвращает в динамически распределяемую область памяти блок, адресуемый указателем ***ptr**, после чего эта память становится доступной для выделения в будущем.

!!!Вызов функции **free()** должен выполняться только с указателем, который был получен ранее в результате вызова одной из функций динамического

распределения памяти. Использование недопустимого указателя при вызове, скорее всего, приведет к разрушению механизма управления памятью и, возможно, вызовет крах системы.

Задание 13.1. Исследовать программу считывания строк разной длины с использованием массива указателей, когда строки вводятся с клавиатуры, и вывода считанных строк на дисплей.

Примем количество символов в строке 79, что позволит разместить ее в окне **DOS** на экране дисплея.

Программный код для решения данного задания следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
#define N 79
int _tmain(int argc, _TCHAR* argv[])
{
int i, m = 3;
char *str[N+1];
char *str2[] = {"st", "nd", "rd"};
   for (i = 0; i < m; ++i)
      str[i] = (char *) calloc((N+1), sizeof(char));
cout<<"\n Dynamic reading strings of different lengths\n\n";
   for (i = 0; i < m; ++i)
   {
      if (str[i] == NULL)
      {
         cout<<"\n\t Error memory allocation.\n";
         cout<<"\n Press any key: ";
         getch();
         exit(1);
      }
   cout<<"\t Enter  string: ";   //<< i+1<< str2[i];
   gets_s(str[i], sizeof(str)/sizeof(char));
   }
cout<<"\n\t The strings are:\n";
   for (i = 0; i < m; ++i)
      cout<<"\t   \n"<<str[i];
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
Dynamic reading strings of different lengths

        Enter  string: ddddddddddddddddd
        Enter  string: xzccxcxc  dfvxcvv
        Enter  string: xcxcc  cvxczccv  dczc

        The strings are:

ddddddddddddddddd
xzccxcxc  dfvxcvv
xcxcc  cvxczccv  dczc_
```

Приведенные выше результаты выполнения исследуемой программы подтверждают, что функция **calloc()** действительно обеспечивает выделение участков памяти для массивов (в данном случае символьных). Ведь иным способом память в данной задаче невыделялась.

Задание 13.2. Самостоятельно разработать программу, где:

1. Вывод символьного массива осуществите на основе его разыменования.
2. Вместо функции **calloc()** примените функцию **malloc()** и введите (а потом выведите) свои фамилию, имя, номер группы, специальность (буквами латинского алфавита).
3. Видоизмените программу для ввода одной строки с несколькими словами различной длины (с различным количеством символов), а затем сформируйте массив строк из заданных слов. Предусмотрите вывод строк сформированного символьного массива.
4. Отсортируйте символьный массив по убыванию длин введенных слов, считая, что прописные буквы имеют приоритет над строчными. Сделайте вывод отсортированного массива на дисплей.

Задание 13.3. Исследовать программу заполнения одномерного массива случайными числами, распределенными по стандартному нормальному закону. Размерность массива вводится с клавиатуры пользователем.

Для решения примера выберем метод Марсальи – Брея. Его этапы:

1) генерируются два равномерно распределенных случайных числа $R1$, $R2$ из интервала [0; 1];
2) формируются два соотношения:

$$V1=-1+2R1,\ \ V2=-1+2R2;$$

3) составляется сумма:

$$S = V1^2 + V2^2;$$

4) если $S \geq 1$, то пункты 1 – 3 повторяются;

5) если S<1, то вычисляется первая пара случайных чисел z1, z2:

$$z1 = V1\sqrt{\frac{-2\ln S}{S}}; \quad z2 = V2\sqrt{\frac{-2\ln S}{S}}.$$

Примечание 1. Нормальный закон характеризуется двумя параметрами: математическим ожиданием и среднеквадратическим отклонением. Эти параметры соответственно равны 0 и 1 для стандартного нормального закона.

Для оценки математического ожидания используется среднее значение данного объема *n* выборки случайных чисел. Для оценки дисперсии *D* могут быть использованы следующие формулы:

$$D = \frac{1}{n-1}\sum_{i=1}^{n}(z_i - m)^2;$$

$$m = \frac{1}{n}\sum_{i=1}^{n} z_i,$$

где *m* – среднее значение заданного массива.

Программный код для решения данного задания следующий:

```
#include "stdafx.h"
#include "iostream"
#include <conio.h>
#include <stdlib.h>
#include <time.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double *Norm1, *Norm2, R1, R2, z1, z2, V1, V2, S;
int i, j, n;
time_t t;
srand((unsigned) time(&t));                    //Функция-генератор случ. чисел
cout<<"\n\t Enter the size of the array: ";
cin>>n;
Norm1=(double*)malloc(n*sizeof(double));   //Выделение
Norm2=(double*)malloc(n*sizeof(double));   //памяти
S = 1.0;                                       //Реализация алгоритма Марсальи-Брея
   for(i=j=0; i<n; ++i, ++j)
   {
      while(S>=1.0)
      {
         R1=(double) rand()/RAND_MAX;          //Получение случайного
числа
```

```
        R2=(double) rand()/RAND_MAX;          //Получение случайного
числа

        V1=2.0*R1-1.0;
        V2=2.0*R2-1.0;
        S=(V1*V1+V2*V2);
      }
   z1=V1*sqrt(-2.0*log(S)/S);
   z2=V2*sqrt(-2.0*log(S)/S);
   Norm1[i]=z1;
   Norm2[j]=z2;
   S=1.0;
   }
        //Вывод нормально распределенных случайных чисел
cout<<"\n\t Normally distributed random numbers:\n";
   for(i=j=0; i<n; i++, j++)
   {
      cout<<"\n\t"<<Norm1[i];
      cout<<"\n\t"<<Norm2[j];
   }
        //Освобождение памяти
free(Norm1);
free(Norm2);
getch();
return 0;
}
```

Примечание 2. **time_t** - это тип данных, который используется для представления целого количества секунд, прошедших после полуночи (00:00 часов) 1 января 1970 года в формате GMT. Т.е. обычный целочисленный тип данных. Считается, что это случайное число.

Примечание 3. Запись **srand((unsigned) time(&t))** означает, что генератор случайных чисел инициализируется неким числом (то есть числа будут выбрасываться в случайном порядке в пределе от нуля до этого числа), которое возвращает функция **time()**.

Результат выполнения программы следующий:

```
Enter the size of the array: 10

Normally distributed random numbers:

0.950125
0.734338
0.251343
-0.124698
-0.154211
-1.29208
-0.800168
0.488684
-0.12152
0.877688
0.064686
-1.60629
-1.32642
1.53636
-0.191808
-0.421441
0.719499
0.515966
0.344827
0.47012_
```

Очевидно, что в результате выполнения программы с использованием функций, резервирующих и освобождающих динамическую память, получен массив нормально распределенных случайных чисел.

Задание 13.4. Разработать программу на базе **Задания 13.3**, где рассчитать минимальное и максимальное значения сформированных случайных чисел.

Задание 13.5. Исследовать программу заполнения одномерного символьного массива заданным числом (вводимым с клавиатуры) символов с добавлением символа восклицательного знака **!** в конце массива.

Для решения примера используем функции динамического распределения памяти **malloc()** и **realloc()**.

Примечание. Функция **flushall()** записывает содержимое всех буферов, связанных с открытыми **input** потоками, в соответствующие файлы. После вызова функции **flushall**() все потоки остаются открытыми.

Программный код для решения данного задания следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int n, m;
char *ptr;
cout<<"\n Enter a dimention of character array: ";
```

```
cin>> n;                                    //Задание размерности массива
flushall();                                 //
ptr=(char*)malloc((n+1)*sizeof(char));      //Выделение памяти
   if(!ptr)
   {
      cout<<"\n\t 1st Error! ";
      cout<<"\n\n Press any key: ";
      getch();
      return -1;
   }
cout<<" Enter a character array of no more than  "<< n<<" characters: ";
gets_s(ptr, n+1);                           //Ввод строки символов
m=strlen(ptr);                              //Определение числа символов в
строке
cout<<"\n Start line:\n";
cout<<"   "<<ptr;
ptr=(cha *)realloc(ptr, (m+2)*sizeof(char));    //Перераспределение памяти
   if (!ptr)
   {
      cout<<"\n\t 2nd Error! ";
      cout<<"\n\n Press any key: ";
      getch();
      return -1;
   }
      //Присоединение к массиву символов еще одного символа
strcat_s(ptr, m+2,  "!");
cout<<"\n Start line and character  '!'";
cout<<"\n  "<<ptr;
free (ptr);                                     //Освобождение памяти
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
Enter a dimention of character array: 20
Enter a character array of no more than 20 characters: Hello, students

Start line:
Hello, students

Start line and character "!":
Hello, students!
```

Результаты работы анализируемой программы свидетельствуют о том, что, во-первых, выделяется динамическая память для вводимого символьного массива,

а, во-вторых, использование указателей позволяет добавлять к исходному массиву дополнительный символ.

Задание 13.6. Самостоятельно разработать программу на базе **Задания 5.5**, где двумя способами добавляется три восклицательных знака к исходной символьной строке.

Задание 13.7. Самостоятельно разработать программу на базе **Задания 5.5**, где:

1. Выведите сформированный массив символов в обратном порядке.
2. Осуществите вывод массива символов с дополнительным случайным символом без применения функции **strcat_s()**.

Задание 13.8. Исследовать программу транспонирования матрицы, размерность которой (количество строк и количество столбцов) вводится с клавиатуры, а элементы – вещественные случайные числа, распределенные по равномерному закону из интервала [0; 15].

Транспонированная матрица – это матрица A^T, полученная из исходной матрицы A заменой строк на столбцы.

Программный код для решения данного задания следующий:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
#include <time.h>
#include <locale.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j, n, m;
double *pA, *pB;                                    //Объявление указателей
srand((unsigned)time(NULL));                        //
setlocale(LC_ALL, "Russian");                       //Смена шрифта на русский
cout<<"\n   Введите размерность матрицы: ";
cout<<"\n   число строк и число столбцов через пробел: ";
cin>>n>>m;
   pA = (double *) calloc((n*m),sizeof(double));    //Выделение динамической
   pB = (double *) calloc((n*m),sizeof(double));    //памяти для 2-х массивов
   for (i = 0; i < n*m; ++i)                        //Цикл для получения
массива
                                                    //указателей случайных
чисел
```

```
      pA[i] = 15.0*rand()/RAND_MAX;          //Получение случайного
числа
setlocale(LC_NUMERIC, "English");            //Смена шрифта  для
                                             //десятичной точки
cout<<"\n Исходная матрица:\n";
   for (i = 0; i < n; ++i)                   //Цикл для вывода матрицы
A
   {
      cout<<endl;
         for(j = 0; j < m; ++j)
            cout<<"   "<<pA[i*m+j];
   }
   for (i = 0; i < n; ++i)                   //Цикл для транспонирования
                                             //матрицы А
      for (j = 0; j < m; ++j)
         pB[j*n+i]=pA[i*m+j];                //Смена индексов у
элементов
cout<<"\n\n Транспонированная матрица:\n";
   for (j = 0; j < m; ++j)
   {
      cout<<endl;
         for(i = 0; i < n; ++i)
            cout<<"   "<<pB[j*n+i];
   }
free(pA); free(pB);                          //Освобождение выделенной памяти
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
Введите размерность матрицы -
число строк и число столбцов через пробел: 4 5

Исходная матрица:

 10.8470  13.9457   8.8553   5.4096   7.6989
  0.0641   2.4990  12.4145  11.0709  12.2121
  3.0035   8.8351   1.6681  14.8531  14.7404
 10.7733   1.0387  11.2906  11.4536   3.5162

Транспонированная матрица:

 10.8470   0.0641   3.0035  10.7733
 13.9457   2.4990   8.8351   1.0387
  8.8553  12.4145   1.6681  11.2906
  5.4096  11.0709  14.8531  11.4536
  7.6989  12.2121  14.7404   3.5162_
```

Результаты работы анализируемой программы свидетельствуют о том, что, во-первых, выделяется динамическая память для вычисляемого действительного

массива, а, во-вторых, использование указателей позволяет несложным образом транспонировать и вывести на экран исходную матрицу.

Задание 13.9. Самостоятельно разработать программу для сложения двух матриц с использованием функций для выделения динамической памяти и указателей на эти матрицы.

Контрольные вопросы

1. Что такое динамическая память?
2. Какие средства языка **C++** используются для хранения данных с динамическим выделением памяти компьютера?
3. Какие основные библиотечные функции языка **C++** используются для динамического распределения памяти?
4. Каково различие в действии функций **malloc()** и **calloc()**?
5. Как осуществляется перераспределение динамической памяти?
6. Для каких типов данных возможно динамическое распределение памяти?
7. В каком случае функции динамического распределения памяти возвращают нулевой указатель?

Лабораторная работа № 14 (3 занятия)

ФАЙЛОВЫЕ ВВОД И ВЫВОД В СРЕДЕ VISUAL C++ 2010

Цель работы: изучение базовых функций файловой системы языка программирования C++ и приемов создания, чтения, записи и модификации файлов.

1. Понятие файла и потока в C++

При решении большинства задач на любом языке программирования возникает необходимость записывать, хранить и получать информацию, используя файлы.

Файл (file) – это именованный объект, хранящий данные (программа или любая другая информация) на каком-либо носителе (винчестер, флешь-память, CD и др.).

Поток (stream) - это виртуальное логическое устройство, связывающее программу с физическим устройством ввода-вывода (терминалом, дисководом и др.). Поскольку потоки не зависят от физических устройств, то одна и та же функция может записывать информацию на диск или на другое устройство.

Поток связывают с определенным файлом, выполняя операцию **открытия.** Как только файл открыт, можно проводить обмен информацией между ним и программой.

Файл отсоединяется от определенного потока (т.е. разрывается связь между файлом и потоком) с помощью операции **закрытия**. При закрытии файла, открытого с целью вывода, содержимое (если оно есть) связанного с ним потока записывается на внешнее устройство. Этот процесс, который обычно называют дозаписью потока, гарантирует, что никакая информация случайно не останется в буфере диска. Если программа завершает работу нормально, то все файлы закрываются автоматически. В случае аварийного завершения программы, файлы не закрываются.

В языке C++ существует два типа потоков:

- текстовый (**text**);
- двоичный (**binary**).

Текстовый поток – это последовательность символов. В **C++** считается, что текстовый поток организован в виде строк, каждая из которых заканчивается символом новой строки. Среди символов в потоке может быть символ возврата каретки, перехода на новую строку и др.

Двоичный поток – это последовательность байтов, однозначно соответствующих информации, находящейся на внешнем носителе. Причем никакого преобразования символов не происходит.

Двоичный файл это последовательность данных не текстового формата. Поэтому редактировать файл в текстовом редакторе невозможно. С другой стороны такой файл может иметь меньший объем, а для редактирования могут быть предназначены специальные редакторы.

Доступ к информации в потоках и в файлах неодинаков. Например, из файла на диске можно выбрать 3-ю запись или заменить 6-ю запись. В то же время в файл, связанный с печатью, информация может передаваться только последовательно. Это иллюстрирует самое главное отличие между потоками и файлами.

Потоки, использующиеся в программах, делятся на:

- входные, из которых читается информация;
- выходные, в которые вводится информация;
- двунаправленные, допускающие как чтение, так и запись.

В соответствии с особенностями «устройства», к которому «присоединен» поток, потоки принято разделять на:

- стандартные;
- консольные;
- строковые;
- файловые.

Стандартные и консольные потоки соответствуют передаче данных от клавиатуры до дисплея.

Если символы потока в совокупности образуют символьный массив в основной памяти, то это **строковый поток.**

Если информация размещается на внешнем носителе данных, то это **файловый поток** или просто **файл.**

До сих пор во всех программах курса выполнялся обмен со стандартными потоками:

cin – стандартный входной поток, связанный с клавиатурой;

cout – стандартный выходной поток, связанный с экраном дисплея.

Используя операции включения (записи) данных в поток << и извлечения данных из потока >> выполнялся обмен данными с дисплеем и с клавиатурой ПК. Для этих целей в программу необходимо было включить заголовочный файл **iostream.h**.

2. Создание, открытие и закрытие файлов в Visual C++ 2010. Ввод и вывод данных в эти файлы

Библиотека ввода и вывода данных в файлы подключается в заголовочном файле **stdio.h** и включает средства для работы с последовательными файлами. **Последовательный файл** можно представить как именованную цепочку (ленту, строку) байтов, имеющую начало и конец. Последовательный файл отличается от файла с другой организацией тем свойством, что чтение из файла (или запись в него) ведется байт за байтом от начала до конца.

В каждый момент времени позиция в файле, из которого выполняется чтение (или запись), определяется значением **указателя позиции записи и чтения файла.**

Установка указателя записи (чтения) на нужные байты выполняется либо автоматически, либо за счет управления их положением. В библиотеке ввода-вывода есть соответствующие средства.

При работе с файлами используются следующие операции:

- создание файла;
- удаление файла;
- поиск файла на внешнем носителе;
- открытие файла;
- чтение из файла или запись данных в файл;
- позиционирование файла;
- закрытие файла.

Все перечисленные действия могут быть выполнены с помощью средств библиотеки ввода-вывода.

Операция **открытия файла** связывает поток с определенным файлом. Операция **закрытия** файла разрывает эту связь.

Запись или чтение из файла осуществляются с помощью указателя файла. **Указатель файла** – это указатель на структуру типа **FILE.**

Для объявления переменной–указателя на файл **F**, например, **pF**, используется следующий оператор:

FILE *pF;

Указатель файла указывает на структуру, содержащую различные сведения о файле, его имя, статус и указатель текущей позиции в начало файла

При обработке данных, хранящихся в файле, программе необходимо иметь доступ к данному файлу. С помощью переменной ***fp** ведется в дальнейшем вся работа с файлом в программе.

В файле **stdio.h** определены функции для работы с файлами. Основные из них приведены в таблице 6.1.

Таблица 6.1. **Функции для работы с файлами в Visual C++ 2010**

Функция	**Описание функции**
fopen()	Открыть файл
fclose()	Закрыть файл
fseek()	Переместить (установить) указатель позиции файла
feof()	Возвращает значение «истина», если достигнут конец файла
ferror()	Возвращает значение «ложь», если найдена ошибка
fread()	Читает блок данных из потока.
fwrite()	Записывает блок данных в поток
rewind()	Устанавливает указатель позиции файла на начало
remove()	Удаляет файл

При открытии файла функция **fopen()** выполняет два действия:
- открывает файл и связывает его с потоком;
- возвращает указатель, ассоциируемый с этим файлом.

Прототип функции **fopen()** имеет следующий вид:

FILE *fopen (char *filename, char mode);

где **char *filename** – строка, содержащая полное имя файла на диске; **char *mode** – строка, определяющая режим открываемого файла.

Возможны следующие режимы открытия файла:

“**r**” – открыть файл для чтения;
“**w**” – создать файл для записи;
“**a**” – открыть для добавления в существующий файл;
“**rb**” – открыть двоичный файл для чтения;
“**wt**” – создать текстовый файл для записи;
“**at**” – открыть текстовый файл для добавления;
“**r+t**” – открыть текстовый файл для чтения и записи;
“**w+t**” – создать текстовый файл для чтения и записи;
“**a+t**” – открыть текстовый файл для добавления.

Например, для создания файла для записи данных с именем **C:\\Inform.dat** необходимо записать:

```
FILE *F1;
F1=fopen(“C:\\Inform.dat”, “w”);
```

При открытии файла для записи данных в языке C++ в программном коде запись

("D:\\Inform.dat", "w"))==NULL

применяется для проверки того, открылся ли файл. Индикатором является значение **NULL**. В случае, когда файл по каким-то причинам не был открыт, происходит выход из программы с выдачей соответствующего сообщения.

3. Функции записи в файл и чтения из файла

Запись потока байтов в файл выполняет функция **fwrite**:

fwrite(void *p, size, n, FILE *f),

где **void *p** – указатель на буфер некоторого типа, который записывается в файл; **size** – число байт в одном элементе; **n** – число записываемых элементов; **FILE *f** – указатель на файл, куда ведется запись.

Возвращает функция **fwrite** число записанных элементов.

Чтение и запись потока байтов выполняют функции fread и fwrite.

fread(void *buffer, size, count, FILE *f);

где **void * buffer** – указатель на буфер некоторого типа, который считывается из файла; **size** – число байт в одном элементе; **n** – число считываемых элементов; **FILE *f** – указатель на файл, из которого считываются элементы.

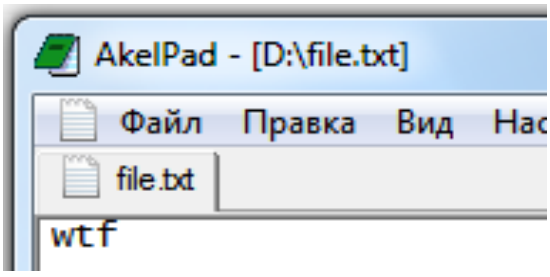
Возвращает функция **fread** число считанных элементов, которое может быть меньше count, если при чтении произошла ошибка или встретился конец файла.

Задание 14.1. Необходимо исследовать программу для записи символьного массива **A** в открываемый файл на диске **D**.

Для этих целей разработан следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include <cstdio>
#include <iostream>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
char A[] = {'w', 't', 'f'};                  //Объявление символьного массива A
FILE* pF=fopen("D:\\file.txt", "wb");        //Объявление указателя pF файлового
                                             //типа и открытие файла F на диске D
fwrite(A, 1, sizeof(A), pF);                 //Запись в файл содержимого буфера
fclose(pF);                                  //Закрытие файла F
getch();
return 0;
}
```

Отлаженная программа дает следующий результат:



Очевидно, что в результате выполнения программы файл **file.txt** на диске **D** открыт и в него записаны заданные символы.

Задание 14.2. Необходимо разработать программу для записи символьного массива **A** в открываемый файл на диске **D**. Символы массива A должны составлять имя и фамилию студента и номер группы (кириллица отображается в файлах).

Задание 14.3. Необходимо исследовать программу для записи переменной **A** в открываемый файл, а затем чтения значения переменной **A** из этого файла и вывода этого значения на печать.

Для убедительности в программе записывать и считывать значение **A** будем с использованием двух разных указателей, чтобы быть полностью уверенными в считывании значения переменной именно из файла.

Для решения задачи разработан следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A, B, S1,S2;                    //Объявление переменных и буферов
памяти
double *pS1,*pS2;                      //Указатель на буфер обмена
pS1=&S1;                               //Присваив. адреса S1 указателю pS1
pS2=&S2;                               //Присваив. адреса S2 указателю pS2
cout<<"Vvedite A"<<endl;
cin>>A;                                //Ввод A
cout<<"A= "<<A<<endl;
   FILE*F1;                            //Объявление файла для записи
   F1=fopen("D:\\Inform.dat", "w");    //Открытие файла для записи
S1=A;                                  //Запись значения A в буфер
   fwrite(pS1,4,sizeof(pS1),F1);       //Запись из буфера в открытый файл
   fclose(F1);                         //Обязательное закрытие файла
   F1=fopen("D:\\Inform.dat", "r");    //Открытие файла для чтения
   fread(pS2,4,sizeof(pS2),F1);        //Чтение буфера pS из файла
B=S2;                                  //Запись из буфера в переменную B
cout<<"B= "<<B;
fclose(F1);                            //Обязательное закрытие файла
getch();
return 0;
}
```

Внимание!!! Если файл открывается для записи, то существующий файл удаляется и создается новый.

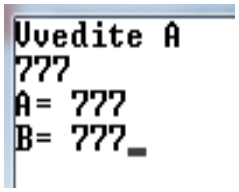
При открытии файла для чтения, требуется, чтобы он существовал.

В случае открытия файла для чтения и записи существующий файл не уничтожается, однако создается, если он не существовал ранее.

После чтения данных из файла (или записи данных в файл) он должен быть закрыт следующим образом:

fclose (F1);

После отладки программы должен быть получен, например, следующий результат:



Очевидно, что выполнено открытие файла **Inform.dat**, запись в него значения **A** и затем чтение этого значения в буфер памяти.

Задание 14.4. Необходимо исследовать программу для записи переменной **A** в открываемый файл, а затем чтения значения переменной A из этого файла и вывода этого значения на печать. Ввести блоки для проверки открытия файлов.

Программный код для решения данной задачи аналогичен коду из **Задания 6.3**, но после каждого открытия файла в программу добавлены операторы для проверки открытия файла с сообщением в случае невозможности такого открытия.

Для этих целей применяется следующий программный код:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A, B, S1,S2;                    //Объявление переменных и
буферов
                                       //памяти
double *pS1,*pS2;                      //Указатель на буфер обмена
pS1=&S1;                               //Присваив. адреса S1 указателю
pS1
pS2=&S2;                               //Присваив. адреса S2 указателю
pS2
cout<<"Vvedite A"<<endl;
cin>>A;                                //Ввод A
cout<<"A= "<<A<<endl;
   FILE*F1;                            //Объявление файла для записи
   F1=fopen("D:\\Inform.dat", "w");    //Открытие файла для записи
   if(F1==NULL)                        //Стандартная процедура
   {                                   //проверки открытия файла с
   cout<<"Couldn't open the file F1"<<endl; //сообщением  в случае ошибки
   exit(1);
   }
S1=A;                                  //Запись значения A в буфер
```

```
fwrite(pS1,4,sizeof(pS1),F1);          //Запись из буфера в открытый файл
fclose(F1);                            //Обязательное закрытие файла
F1=fopen("D:\\Inform.dat", "r");       //Открытие файла для чтения
   if(F1==NULL)                        //Стандартная процедура проверки
   {
   cout<<" Couldn't open the file F1"<<endl;
   exit(1);
   }
fread(pS2,4,sizeof(pS2),F1);           //Чтение буфера pS из файла
B=S2;                                  //Запись из буфера в переменную B
cout<<"B= "<<B;
fclose(F1);                            //Обязательное закрытие файла
getch();
return 0;
}
```

В данной программе используется метод определения ошибки при открытии создаваемого файла. Невозможность открыть файл приравнивается к константе **NULL**, которая определена в библиотеке **stdio.h**. Функция **exit(1)** определена в файле **stdlib.h** и прекращает выполнение программы, а единицу возвращает в операционную систему. Перед прекращением программы она закрывает все открытые файлы, освобождает буферы и выводит все необходимые сообщения на экран.

Отлаженная программа дает следующий результат:

```
Vvedite A
56.5
A= 56.5
B= 56.5
```

Таким образом, значение переменной было верно записано в специально открытый файл **F1** и прочитано из этого файла.

Кроме того, можно убедиться, что был открыт файл **Inform. dat** на диске **D**.

Задание 14.5. Необходимо разработать программу для чтения значения переменной **B** из открываемого файла и вывода этого значения на экран. Для этого удалите из программного кода **Задания 14.4** все операторы, отвечающие за объявление, ввод, запись в файл значения переменной A, за открытие для записи файла **F1**.

Отладьте программу и результат ее выполнения запишите в файле **Результат.**

После этого удалите файл **Inform.dat** с диска **D** и вновь запустите программу. Зафиксируйте результат выполнения в отчете и письменно объясните его.

Внимание!!! В дальнейшем не будем использовать блоки проверки открытия файла с сообщением в простых задачах, где низкая вероятность того, что файл может не открыться.

В более сложных задачах это необходимо выполнять обязательно, а также в случаях некорректной работы программ.

Задание 14.6. Разработать программу, которая вводит в память компьютера с клавиатуры одномерный массив целых чисел **A** произвольного размера **n**, создает и открывает для записи и чтения файл **D:\\Inform.dat**, после чего записывает массив **A** в данный файл. Затем данные из файла должны быть записаны в новый массив **B** и выведены на экран дисплея.

Такая программа для записи массива в файл, а затем чтения массива из файла и отображения на экране дисплея будет иметь следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                    //Максимальный размер массива
int A[N];                          //Объявление массива A
int i,n;                           //Параметр цикла i и реальный размер
                                   //массива n
int S1,S2;                         //Объявление переменных S1 и S2 в
                                   //качестве буферов обмена
int *pS1, *pS2;                    //Объявление указателей pS1 и pS2
pS1=&S1;                           //Присваивание адресов переменных S1
pS2=&S2;                           //и S2 указателям pS1 и pS2
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                            //Ввод реального размера массива A
cout<<"Vvedite massiv A:"<<endl;
   for(i=0;i<n;i++)                //Цикл для ввода массива A
      cin>>A[i];                   //Ввод i-го элемента массива A
cout<<" Massiv A: "<<endl;
   for(i=0;i<n;i++)                //Цикл для вывода массива A
      cout<<A[i]<<" ";             //Вывод i-го элемента массива A
FILE*F1;                           //Объявление переменной–указателя
                                   //файлового типа
F1=fopen("D:\\Inform.dat", "w");
   for(i=0;i<n;i++)                //Цикл для записи в открытый файл
                                   //массива A
   {
      S1=A[i];                     //Запись i-го элем. массива A в буфер S1
      fwrite(pS1,4,1,F1);          //Запись i-го элемента массива A из
                                   //буфера в файл
```

```
   }
fclose(F1);                          //Закрытие файла
int B[N];                            //Объявление нового массива B
FILE*F2;                             //Объявление указателя файлового типа
F2=fopen("D:\\Inform.dat","r");
   for(i=0;i<n;i++)                  //Цикл для чтения из файла в буфер S2
   {
      fread(pS2,4,1,F2);
      B[i]=S2;                       //Запись элемента данных из буфера S2 в
                                     //i-й элемент массива  B
   }
fclose(F2);                          //Закрытие файла
cout<<endl<<" New massiv B:"<<endl;
   for(i=0;i<n;i++)                  //Цикл для вывода массива B
   cout<<B[i]<<" ";                  //Вывод на экран i-го элемента массива
B
getch();
return 0;
}
```

Данная программа несколько усложнена - можно было обойтись одной переменной **S** вместо **S1** и **S2**, т.е. одним указателем и т.д. Этот приём использован для того, чтобы показать: массив B получает свои элементы именно из открываемого файла, а не из буфера **S**.

После выполнения программы на экран будет выдана следующая информация:

```
Uvedite razmer n:
6
Uvedite Mas:
3 6 1 7 9 4
 Massiv A:
3 6 1 7 9 4
 New massiv B:
3 6 1 7 9 4
```

Очевидно, что данные (элементы массива) были без искажений записаны и считаны из созданного для этого файла.

Задание 14.7. Необходимо разработать программу только для чтения значений одномерного массива **B** из открываемого файла **Inform.dat** через буфер и вывода этого массива на экран. Для этого удалите из программного кода **Задания 14.6** все операторы, отвечающие за объявление, ввод, запись в файл массива A, а также за открытие для записи файла **F1.**

Отладьте программу и результат её выполнения запишите в файле **Результат.**

Сделайте попытку прочитать из файла **Inform.dat** элементы массива **B** без использования буфера. Письменно в отчёте опишите и объясните результат.

После этого удалите файл **Inform.dat** с диска **D** и вновь запустите программу. Зафиксируйте результат выполнения в отчёте и письменно объясните его.

Задание 14.8. Необходимо разработать программу для записи и чтения значений одномерного массива **B** из открываемого файла **Inform.dat** через буфер и вывода этого массива на экран (**Задание 14.6**).

Блоки операторов, отвечающие за открытие файла и запись в него массива A, а также за считывание из файла массива A, необходимо оформить в виде функций с помощью указателей на массив.

Отладьте программу и результаты её выполнения запишите в файле **Результат**.

4. Запись числовых данных в текстовый файл в языке C

Использование бинарных (двоичных) файлов не всегда является удобным. В некоторых случаях необходимо, например, просмотреть файл с числовыми данными в обычном текстовом редакторе.

Эту задачу можно решить с использованием операторов записи **fprintf()** и считывания **fscanf()** из языка C.

У функции **fprintf()** формат аналогичен формату исследованной ранее функции **printf()**. Например, для записи некоторого заголовка в текстовый файл необходимо записать

fprintf(F, "\r\n\t Matrix (%d x %d), initial number: %d\r\n", n, m, xi),

где **F** – переменная-указатель на файл, в который необходимо записать данную информацию.

Для записи некоторого числа в текстовый файл необходимо записать

fprintf(F, "%5d", a),

где **a** – некоторая числовая переменная или константа.

Задание 14.9. Необходимо разработать программу заполнения матрицы размера **n × m** нечетными целыми числами с выводом результата на консоль и в текстовый файл. Размеры матрицы и начальное нечетное число задаются пользователем с клавиатуры.

При разработке программы воспользуемся оператором **fprintf** из языка C, который позволяет записывать числовые данные в текстовый файл без кавычек.

Тогда программа для записи массива в текстовый файл и отображения на экране дисплея будет иметь следующий вид:

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

```
#include <stdlib.h>
#include <stdio.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j, x, xi, n, m, *A;
FILE *F;
F=fopen("D:\\data.txt","w");                    //Открытие файла F на диске D
под
                                                //названием data.txt
char str[]="D:\\data.txt";                      //Месторасположение файла F
   if((F=fopen(str, "w")) == NULL)              //Проверка, открыт ли файл с
      {                                         //сообщением в случае ошибки
      printf("\n\t The file could not be opened.\n ");
      printf("\n Press any key: ");
      getch();
      return -1;
      }
cout<<"\n\t Enter the number of lines and columns: ";
cin>>n>>m;
cout<<"\n\t Enter the odd number: ";
cin>>x;
xi=x;
A=(int *)calloc(n*m, sizeof(int));        //Резервирование динамической памяти
cout<<"\n\t Matrix  " <<n<<"*"<<m<<" initial number: "<<x;
fprintf(F, "\r\n\t Matrix (%d x %d), initial number: %d\r\n", n, m, xi);//Запись
                                          //заголовка в текстовый файл data.txt
   for (i=0; i<n; ++i)                    //Вложенные циклы для заполнения
      for (j=0; j<m; ++j)                 //матрицы A числами
      {
         A[i*m+j]=x;
         x+=2;                            //Получение очередного нечетного числа
      }
      //Организация записи элементов матрицы A в текстовый файл
   for (i = 0; i<n; ++i)
   {
      cout<<"\t"<<endl;
      fprintf(F, "\r\n ");
         for (j = 0; j<m; ++j)
         {
            cout<< A[i*m+j]<<"   ";
            fprintf(F, "%5d", A[i*m+j]);  //Запись очередного элемента матрицы
                                          //A в текстовый файл data.txt
         }
   }
```

```
fclose(F);                                  //Закрытие файла
cout<<"\n\n Result of record look in file  "<< str;
getch();
return 0;
}
```

Результат выполнения программы в части вывода на консоль следующий:

```
Enter the number of lines: 5
Enter the number of columns: 8
Enter the odd number: 33

Matrix (5 x 8), initial number: 33

 33   35   37   39   41   43   45   47
 49   51   53   55   57   59   61   63
 65   67   69   71   73   75   77   79
 81   83   85   87   89   91   93   95
 97   99  101  103  105  107  109  111

Result of record look in file D:\data.txt
```

Текстовый файл **data.txt** с заполненной матрицей представлен на рис. 14.1.

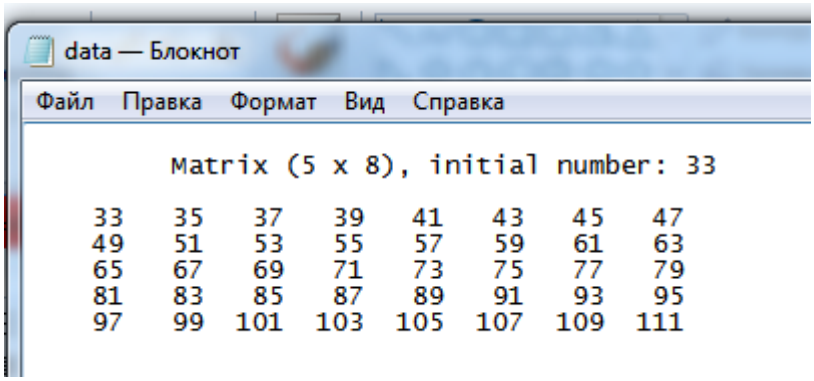


Рис. 14.1. Матрица нечетных чисел в текстовом файле **data.txt**

Задание 14.10. Необходимо на базе программы из **Задания 14.9** разработать программу заполнения матрицы размера **n** × **m** четными целыми числами с выводом результата на консоль и в текстовый файл, а также записи этой матрицы в бинарный файл и чтения из него с выводом на консоль.

Для резервирования динамической памяти для массива **A** используйте операторы **new** и **delete**. Размеры матрицы и начальное четное число задаются пользователем с клавиатуры.

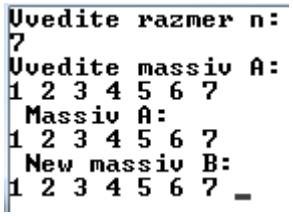
Задание 14.11. Разработать программу, которая вводит в память компьютера с клавиатуры одномерный массив целых чисел **A** произвольного размера **n**, создает и открывает для записи и чтения файл **D:\\Inform.dat**, после чего записывает массив **A** в данный файл. Затем данные из файла должны быть записаны в новый массив **B** и выведены на экран дисплея.

Отличие разрабатываемой программы от программы из **Задания 6.6** состоит в том, что массив данных **A** будет записываться в файл и считываться из него без организации циклического процесса, а через указатель на массив.

Такая программа для записи массива в файл, а затем чтения массива из файла и отображения на экране дисплея будет иметь следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                    //Максимальный размер массива
int A[N];                          //Объявление массива A
int i,n;                           //Параметр цикла i и реальный размер
                                   //массива n
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                            //Ввод реального размера массива A
cout<<"Vvedite massiv A:"<<endl;
   for(i=0;i<n;i++)                //Цикл для ввода массива A
      cin>>A[i];                   //Ввод i-го элемента массива A
cout<<" Massiv A: "<<endl;
   for(i=0;i<n;i++)                //Цикл для вывода массива A
      cout<<A[i]<<" ";             //Вывод i-го элемента массива A
FILE*F1;                           //Объявление переменной–указателя
                                   //файлового типа
F1=fopen("D:\\Inform.dat", "w");
   fwrite(A,sizeof A,1,F1);        //Запись массива A в файл
   fclose(F1);                     //Закрытие файла
int B[N];                          //Объявление нового массива B
FILE*F2;                           //Объявление указателя файлового типа
F2=fopen("D:\\Inform.dat","r");    //Открытие файла для чтения
      fread(B,sizeof B,1,F2);      //Считывание из файла массива A
      fclose(F2);                  //Закрытие файла
cout<<endl<<" New massiv B:"<<endl;
   for(i=0;i<n;i++)                //Цикл для вывода массива B
   cout<<B[i]<<" ";                //Вывод на экран i-го элемента массива
B
getch();
return 0;
}
```

Результат выполнения программы в части вывода на консоль следующий:



5. Функция конца файла feof()

При чтении данных из файла размер файла часто неизвестен. Для определения конца файла служит функция **feof() ("eof" – end of file**). Эта функция имеет следующий прототип:

int feof(FILE *F1);

Функция возвращает значение «истина», если достигнут конец файла, и «ложь» - если нет.

Исследуем применение этой функции при файловом вводе и выводе.

Задание 14.12. Исследовать программу, которая вводит в память компьютера с клавиатуры одномерный массив целых чисел **A** произвольного размера **n**, создает и открывает для записи и чтения файл **D:\\Inform.dat,** после чего записывает массив **A** в данный файл. Затем данные из файла должны быть записаны в новый массив **B** и выведены на экран дисплея. Чтение из файла осуществляется при помощи функция определения конца файла **feof()**.

Такая программа записи массива в файл, чтение массива из файла и отображения его на экране дисплея будет иметь следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                    //Максимальный размер массива
int A[N];                          //Объявление массива A
int i,n;                           //Параметр цикла i и реальный размер
                                   //массива n
int S1,S2;                         //Объявление переменных S1 и S2 в
                                   //качестве буферов обмена
int *pS1, *pS2;                    //Объявление указателей pS1 и pS2
pS1=&S1;                           //Присваивание адресов переменных S1
pS2=&S2;                           //и S2 указателям pS1 и pS2
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                            //Ввод реального размера массива A
cout<<"Vvedite massiv A:"<<endl;
   for(i=0;i<n;i++)                //Цикл для ввода массива A
```

```
      cin>>A[i];                    //Ввод i-го элемента массива A
cout<<" Massiv A: "<<endl;
   for(i=0;i<n;i++)                 //Цикл для вывода массива A
      cout<<A[i]<<" ";              //Вывод i-го элемента массива A
FILE*F1;                            //Объявление переменной–указателя
                                    //файлового типа
F1=fopen("D:\\Inform.dat", "w");
   for(i=0;i<n;i++)                 //Цикл для записи в открытый файл
                                    //массива A
   {
      S1=A[i];                      //Запись i-го элем. массива A в буфер S1
      fwrite(pS1,4,1,F1);           //Запись i-го элемента массива A из
                                    //буфера в файл
   }
fclose(F1);                         //Закрытие файла
int B[N];                           //Объявление нового массива B
FILE*F2;                            //Объявление указателя файлового типа
F2=fopen("D:\\Inform.dat","r");
if(F2==NULL)                        //Открытие файла для записи
                                    //с условием вывода на экран
{
cout<<" Don't open this file F2"<<endl;
exit(1);
}
i=0;
   while(!feof(F2))                 //Условие проверки конца файла в цикле и
                                    //при чтении данных с применением функции
   {                                //определения конца файла feof()
      fread(pS2,4,1,F2);            //Чтение единицы данных из файла в буфер S
      B[i]=S2;                      //Запись единицы данных из буфера S в i-й
                                    //элемент массива B
      i=i+1;
   }
n=i-1;                              //Вычисление количества прочитанных чисел
cout<<"\n Number of elements = "<<n;

fclose(F2);                         //Закрытие файла
cout<<endl<<"New massiv B:"<<endl;
   for(i=0;i<n;i++)                 //Цикл для вывода массива B[N]
   cout<<B[i]<<"  ";                //Вывод на экран i-го элемента массива B
getch();
return 0;
}
```

Примечание! При вводе элементов исходного массива с клавиатуры для выполнения последующего задания введите девять чисел от 1 до 9.

После выполнения программы на экран будет выдана следующая информация:

```
Uvedite razmer n:
9
Uvedite massiv A:
1 2 3 4 5 6 7 8 9
 Massiv A:
1 2 3 4 5 6 7 8 9
 Number of elements = 9
New massiv B:
1 2 3 4 5 6 7 8 9 _
```

Очевидно, что данные (элементы массива) были без искажений записаны и считаны из созданного для этого файла.

Задание 14.13. Исследовать программу, которая открывает для чтения файл **D:\\Inform.dat,** созданный в предыдущей задаче, после чего считывает данные из файла в массив **B** и выводит на экран дисплея. Чтение из файла осуществляется при помощи функции определения конца файла **feof()**.

Кроме этого, в данной программе считывание массива из файла оформлено в отдельную функцию с помощью указателя на массив

Такая программа чтения массива из файла и отображения его на экране дисплея будет иметь следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
#include <stdlib.h>
using namespace std;
double *ArrayReadFile(double *);
int n;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                    //Максимальный размер массива
double B[N];                       //Объявление массива B
int i;                             //Параметр цикла i
ArrayReadFile(B);
cout<<endl<<" New massiv B:"<<endl;
for(i=0;i<n;i++)                   //Цикл для вывода массива B
   cout<<B[i]<<" ";                //Вывод на экран i-го элемента массива  B
getch();
return 0;
}

double *ArrayReadFile(double *B)
{
int i;                             //Параметр цикла i размер массива n
```

```
int S1;                          //Объявление переменных S1
int *pS1;                        //Объявление указателей  pS1
pS1=&S1;                         //Присваивание адресов переменных S1
FILE*F1;                         //Объявление переменной–указателя
                                 //файлового типа
F1=fopen("D:\\Inform.dat", "r");
if(F1==NULL)                     //Стандартная процедура
   {                             //проверки открытия файла с
   cout<<"Couldn't open the file F1"<<endl; //сообщением  в случае ошибки
   exit(1);
   }
i=0;
   while(!feof(F1))              //Условие проверки конца файла в цикле и
                                 //при чтении данных с применением
   {                             // функции определения конца файла feof()
      fread(pS1,4,1,F1);         //Счит. единицы данных из файла в буфер S
      B[i]=S1;                   //Запись единицы данных из буфера S в i-й
                                 //элемент массива  B
      i=i+1;
   }
n=i-1;                           //Вычисление колич. прочитанных чисел
cout<<"\n Number of elements = "<<n;
fclose(F1);
return 0;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
 Number of elements = 9
 New massiv B:
1 2 3 4 5 6 7 8 9
```

Очевидно, что данные (элементы массива) были без искажений считаны из файла **Inform.dat**.

6. Позиционирование файла

Функция **rewind ()** устанавливает указатель позиции файла на начало файла. Прототип этой функции имеет следующий вид:

```
void rewind (FILE *fptr);
```

Обращение к функции имеет вид:

```
rewind (F2);
```

Чтение и запись в файл не обязательно делать последовательно. Можно получить доступ непосредственно к нужному байту. Для этих целей используется функция **fseek()**, устанавливающая указатель позиции в нужное место.

Прототип этой функции имеет вид:

int fseek(FILE *fptr, long num, int mode);

где: **fptr** – указатель на соответствующий файл;

num – количество байт от точки отсчета для установки текущей позиции указателя файла;

mode – определяет точку отсчета.

Точка отсчета	Значение mode
Начало файла	0
Текущая позиция	1
Конец файла	2

Задание 14.14. В файле **C:\\Inform.dat** записано 9 чисел: 1, 2, 3, 4, 5, 6, 7, 8 и 9. Необходимо прочитать данные из этого файла, начиная с 3-го числа (то есть с 3), определить сумму прочитанных чисел и добавить ее в файл **C:\\Inform.dat**. Затем прочитать полученный массив из файла и вывести его на экран дисплея.

Программа для решения задачи будет иметь следующий вид:

```
#include <stdio.h>
#include <stdlib.h>
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int n=30;                    //Максимальный размер массива
float S *pS;                       //Объявление буфера обмена и указателя на
                                   //буфер
float Mas1[n], Sum;                //Объявление массива и суммы
int i,k;
pS=&S;                             //Присваивание адреса буфера указателю
FILE *F1;                          //Объявление указателя файла F2
if((F1=fopen("C:\\Inform.dat", "r+b"))==NULL)    //Открытие файла
                                   //C:\\Inform.dat в режиме чтения и записи
{
cout<<" Don't open this file!"<<endl;
exit(1);
}
```

```
i=0;
fseek(F1, 8, 0);                    //Установка указателя позиции файла
                                    //через 8 байтов с начала файла
   while(!feof(F1))                 //Чтение данных из файла, пока не
                                    //наступит конец файла
   {
      fread(pS,  4, 1, F1);         //Чтение порции из 4-х байтов в буфер S
      Mas1[i]=S;                    //Запись из буфера S в массив
      i+=1;                         //Подсчет количества прочитанных чисел
   }
k=i-1;                              //Определение количества считанных чисел
fclose(F1);                         //Закрытие файла
cout<<endl;
   for(i=0; i<k; i++) cout<<Mas1[i]<<'  '; //Вывод массива на экран
      cout<<endl;
Sum=0;
   for(i=0; i<k; i++)               //Цикл для вычисления суммы
Sum=Sum+Mas1[i];                    //Вычисление суммы
cout<<"Sum= "<<Sum<<endl;           //Отображение суммы на экране
FILE *F2;                           //Объявление указателя файла  F3
F2=fopen("C:\\Inform.dat", "a");    //Открытие файла C:\\Inform.dat в
                                    //режиме добавления
S=Sum;                              //Пересылка суммы в буфер
fwrite(pS, 4, 1, F2);               //Запись суммы из буфера в файл
fclose(F2);                         //Закрытие файла F3
FILE *F3;                           //Объявление указателя файла  F4
if((F3=fopen("C:\\Inform.dat" "r+b"))==NULL) //Открытие файла в режиме
                                             //чтения и записи
{
cout Don’t open this file 2!"<<endl;
exit(1);
}
i=0;
while(!feof(F3))                    //Чтение данных из файла, пока не
                                    //наступит конец файла
{
fread(pS,  4, 1, F3);               //Чтение порции из 4-х байтов в буфер S
Mas1[i]=S;                          //Запись из буфера S в массив
i+=1;                               //Счет количества прочит. чисел из
файла
}
k=i-1;                              //Счет количества прочит. чисел из
файла
fclose(F3);                         //Закрытие файла F4
cout<<endl;
```

```
for(i=0; i<k; i++) cout<<Mas1[i]<<'  ';//Вывод массива на экран
cout<<endl;
}
```

Вид экрана после выполнения программы:

```
3 4 5 6 7 8 9
Sum= 42
1 2 3 4 5 6 7 8 9 42
```

Очевидно, что программа верно считала данные из файла и добавила в конец файла сумму элементов, начиная с третьего.

Контрольные вопросы

1. Что такое адрес переменной?
2. Что такое указатель на переменную?
3. Что такое файл данных?
4. Что такое потоки ввода и вывода данных?
5. Что такое разыменование указателя на переменную?
6. Что такое текстовый поток?
7. Что такое двоичный поток?
8. Для чего необходима библиотека **stdio.h**?
9. Для чего необходима функция **fopen()**?
10. Для чего необходима функция **fclose()**?
11. Для чего необходима функция **feof()**?
12. Для ввода данных в файл необходимо следующее объявление:
    1. **ofstream name_file**;
    2. **ofstream (“filename.txt”)**;
    3. **fwrite(void *p, size, n, FILE *f);**
13. Чтобы выполнить операцию вывода данных из файла необходимо следующее объявление:
    1. **ifstream name_file (“filename.dat”);**
    2. **fread(void *buffer, size, count, FILE *f);**
    3. **fstream name_file (“filename.dat”);**
14. При чтении данных из файла конец файла определяется функцией:
    1. **feol()**;
    2. **feof()**;
    3. **feok()**.
15. Функция **eof()** возвращает значение 0, когда…
    1. Конец файла еще не наступил;
    2. Конец файла наступил;
    3. Эта функция ничего не возвращает.
16. Для определения конца файла записан оператор **while (! name_file.eof()).** Цикл будет выполняться…
    1. Пока функция **eof()** возвращает ложь (0);

2. Когда функция **eof()** возвращает истину (1);
3. Такую конструкцию использовать нельзя.

17. При завершении работы с файлом его нужно закрыть функцией:
1. **close();**
2. **close.file_name;**
3. **file_name.close().**

18. При чтении массивов или структур из файла используется функция:
1. **read;**
2. **fread;**
3. **ifread.**

19. Из какого файла будет прочитана информация при выполнении оператора **ifstream koord (“koord1.dat”);**?
1. **koord**;
2. **koord1**;
3. Здесь имя файла не указано.

20. Функция **write** используется для…
1. Ввода массива (структуры) в файл;
2. Вывода массива (структуры) из файла;
3. Вывода массива (структуры) на экран.

ОГЛАВЛЕНИЕ

Лабораторная работа № 1. Исследование интерфейса программного пакета MICROSOFT VISUAL STUDIO 2010. Разработка программ с линейной структурой в среде VISUAL C++ 2010. Изучение процедур ввода и вывода данных …….……..…….. 2
Лабораторная работа № 2. Разработка и исследование разветвляющихся программ в VISUAL C++ 2010 ……... 19
Лабораторная работа № 3. Разработка и исследование циклических программ в VISUAL C++ 2010. Операторы цикла **WHILE** и **DO...WHILE**)…….**.....** 34
Лабораторная работа № 4. Разработка и исследование циклических программ в VISUAL C++ 2010. Оператор цикла **FOR)..**…… 49
Лабораторная работа № 5. Исследование операций с одномерными массивами в VISUAL C++ 2010 …….……..... 62
лабораторная работа №6. Исследование возможностей среды VISUAL C++ 2010 для операций с многомерными массивами данных ……. 80
Лабораторная работа № 7. Разработка и исследование программ с использованием функций в VISUAL C++ 2010 …….**.....** 96
Лабораторная работа № 8. Исследование возможностей VISUAL C++ 2010 по обработке массивов символьных переменных (строк)…….**.....** 112
Лабораторная работа № 9. Адресация переменных и указатели…….. 119
Лабораторная работа № 10. Настройка компилятора языка C++. Исследование операторов ввода и вывода языка C …… 133
Лабораторная работа № 11. Указатели и массивы в Visual C++ 2010 … 163
Лабораторная работа № 12. Указатели и функции в среде Visual C++ 2010 ……... 182
Лабораторная работа № 13. Динамическое распределение памяти в языке C++ …… 195
Лабораторная работа № 14. Файловые ввод и вывод в Visual C++ 2010 205