МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний автомобільно-дорожній університет
Кафедра інформаційних технологій і мехатроніки

МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт

з дисципліни “Інформаційні технології ”
(розділ 2 “Програмування на С++”)
для студентів 2-го курсу ЦОП
галузі знань 14 “Електрична інженерія”
спеціальність 141 “Електроенергетика, електротехніка та електромеханіка” (для російськомовних студентів)

Конспект лекцій розроблений доцентом кафедри інформатики та прикладної математики кандидатом технічних наук Г. Д. Симбірським

Харків, 2018

Лабораторная работа № 1 (два занятия)

ИССЛЕДОВАНИЕ ИНТЕРФЕЙСА СРЕДЫ VISUAL STUDIO 2010. РАЗРАБОТКА ПРОГРАММ С ЛИНЕЙНОЙ СТРУКТУРОЙ В СРЕДЕ VISUAL C++ 2010. ИССЛЕДОВАНИЕ ПРОЦЕДУР 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;** объявлет 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 C++ будет сказано ниже.

Переменную, не меняющую своего значения в программе, можно использовать как константу. В C++ 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 C++ 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 C++ записываются в строчку.

Например, формула

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

в языке C++ 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 C++ 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 С++
Сложение	+
Вычитание	-
Умножение	*
Деление	/

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

В языке С++ 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 $\mathbf{y = \sin^3(x^4)\dfrac{e^x + z^5 - 4.5\cdot 10^2\sqrt{x}}{tg(a)(z^x + b)}}$ 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.

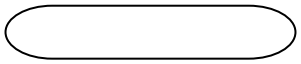
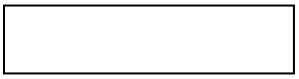
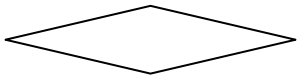
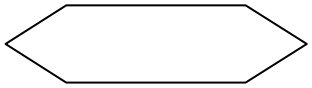
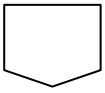
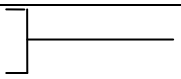
В схемах алгоритмов операционные блоки соединяются между собой линиями потока в виде стрелок. Допускается линии потока, которые идут сверху вниз и слева направо изображать без стрелок. При необходимости блок-схему могут сопровождать комментарии.

Алгоритмизацию решаемых 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 следующих лабораторных работ.

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

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

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 алгорит-мов, подпрограмм, функций
7		Печать документа	Вывод данных на печать
8		Соединитель внутристраничный	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

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).

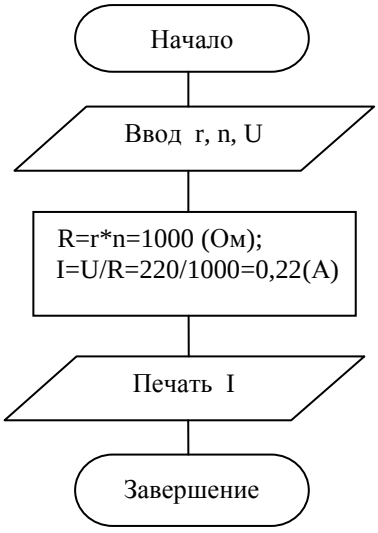


Рис. 1.1. Блок-схема алгоритма определения силы тока в новогодней электрогирлянде

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 C++ 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 C++ 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 файловом менеджере **Unreal Commander**.

Задание 1.1. Создайте проект **Lr1-1** для разработки программы, вычисляющей сумму трех целых чисел с использованием операторов консольных ввода и вывода данных. Проанализируйте код (текст) программы с комментариями, приведенный ниже.

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:\РЕ-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-приглашения  vvedite b . Здесь
                                      //endl – признак перемещения курсора на новую строку
            cin>>b;                   // Ввод значения переменной  b
    a=b+c+d;                          //Вычисление суммы переменных  b, c и 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!! 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.

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!

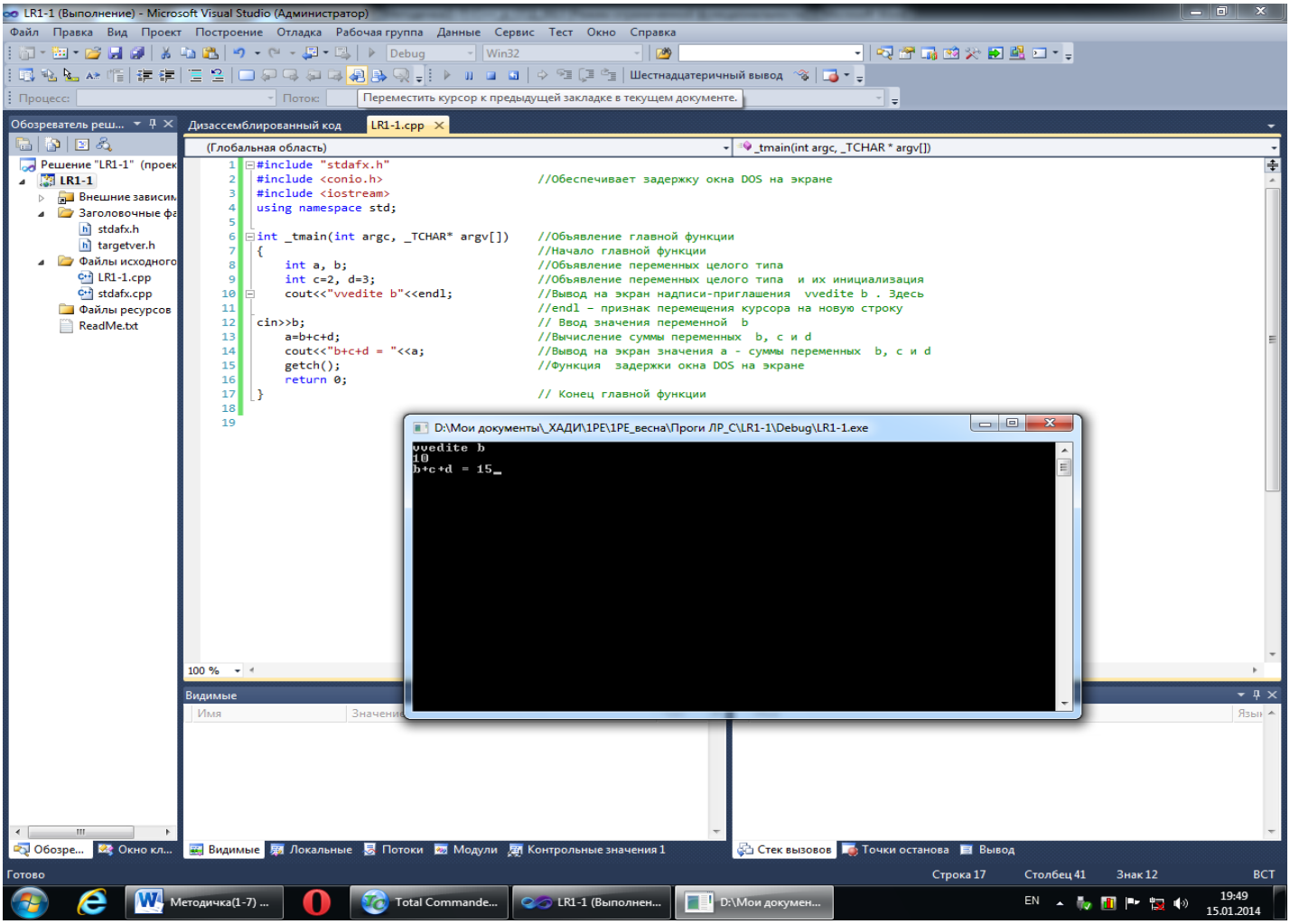


Рис. 1.2. Результаты выполнения **Задания 1**. Копия экрана с результатами выполнения проекта **Lr1-1**.

Задание 1.2. Создайте проект **Lr1-2** для вычисления переменной **a** по формуле $\mathbf{a = b\frac{c + 2d - cd}{d(5c + 4b)}}$ 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 заголовочный файл 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-приглашения 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 переменной 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

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

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

$$\mathbf{V = \pi\, r^2\, h}\ ,$$

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 оператора возведения в степень, то для вычисления $\mathbf{r^2}$ следует воспользоваться оператором умножения, менее трудоемким, чем функция **pow**.

Число $\boldsymbol{\pi}$ 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 $\mathbf{Y = x^5\,\frac{\sin^4 x^3 + 2e^{3a} - tg(\frac{1-x}{1+x})}{\cos 4(x+a)}}$ 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.

// **Lr1-4.cpp:** определяет точку входа для консольного приложения.

```
#include "stdafx.h"
#include <conio.h>                  // Файл conio.h обеспечивает задержку окна DOS на экране дисплея
#include <iostream>                 //Директива include подключает файл заголовка iostream
using namespace std;                //Подключает все имена из пространства имен 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));   //Вычисление Y
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 С++.

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

№ 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	$Y = \frac{\sin\frac{x+1}{4}}{\sin^2 5x + e^{3a}}$	**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	$Y = \frac{tg^3(x+a) - \arccos^2(x+a)}{(x+a)^4}$	**x=0,68; a=5,55**
5	$Y = ctg\frac{1-3x}{1+2x} + \cos^2 5x + e^{7a}$	**x=0,35; a=4,8**
6	$Y = \frac{\cos^3(x+a) - 7(x+a)}{tg(x+a)^4}$	**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	$Y = \frac{tg\frac{4a^2+1}{4}}{\cos^3 2x + e^{2a}}$	**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	$Y = \frac{\arcsin^3 \frac{4x+1}{4}}{\operatorname{ctg}^2 3x + e^{3a}}$	**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	$Y = \frac{\operatorname{arcctg} \frac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	**x=0,37; a=2,75**
18	$Y = \frac{\operatorname{tg}^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	**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 С++.

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

№ варианта	Формула для расчета **Y**	Значения **x, a, в**
1	$Y = \operatorname{tg} \frac{1-x}{1+x} + \sin^2 5x + e^{5a}$	**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	$Y = \frac{\sin^3(a+x^2)}{\sqrt{a + \frac{x^3+1}{2} + \left(\frac{x^2+1}{2}\right)^5}}$	**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	$Y = \dfrac{(a+x)^3 \cdot \ln\dfrac{a+x}{2}}{\sqrt[3]{1+\dfrac{(a+x^3)^2}{4}}}$	**x=1,573; a=1,775**
10	$Y = 2x^3 + \dfrac{2a^4}{\sqrt[3]{1+x^3}} - e^{-\frac{a^2+1}{2}}$	**x=1,573; a=1,775**
11	$Y = \sqrt[4]{\dfrac{1+tg^2\dfrac{x+1}{4}}{1+\dfrac{a^3+1}{4}}} - e^{-\frac{x+1}{4}}$	**x=1,573; a=1,775**
12	$Y = \dfrac{e^{\frac{5x^2}{2}} + e^{1+\frac{4a^3}{2}}}{1+\dfrac{x^2}{2}+\left(\dfrac{a^3}{2}\right)^2}$	**x=1,573; a=1,775**
13	$Y = \dfrac{\lg(1+x^4)}{\sqrt{1+\dfrac{x^2+a^3}{2}+\left(\dfrac{x^2+1}{2}\right)^2}}$	**x=1,573; a=1,775**
14	$Y = \dfrac{1+\left(\dfrac{x}{a}\right)^{2,5} + \sqrt{1+\left(\dfrac{x}{a}\right)^{2,5}}}{1+e^{x^3} + \sqrt{1+\left(\dfrac{x}{a}\right)^{2,5}}}$	**x=1,573; a=1,775**
15	$Y = \left(x^3 + \dfrac{a}{a^2+x^2}\right) - \sqrt{1+\dfrac{a}{a^2+x^4}}$	**x=1,573; a=1,775**
16	$Y = \sqrt[4]{\dfrac{1+tg^2\dfrac{a^3+1}{4}}{1+\dfrac{x+1}{4}}} - e^{-\frac{a^2+1}{4}}$	**x=1,573; a=1,775**
17	$Y = \dfrac{1+\left(\dfrac{x^3}{a}\right)^{2,5} + \sqrt{1+\left(\dfrac{x}{a}\right)^{2,5}}}{1+e^{x} + \sqrt{1+\left(\dfrac{x}{a^3}\right)^{2,5}}}$	**x=1,573; a=1,775**
18	$Y = \ln\left(1+\sqrt{1+x^2}\right) + \dfrac{1+x^2}{\sqrt[3]{1+\dfrac{a^3}{2}}}$	**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 заголовка **io stream**?

Лабораторная работа № 21 (два занятия)

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

1.1. 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.**

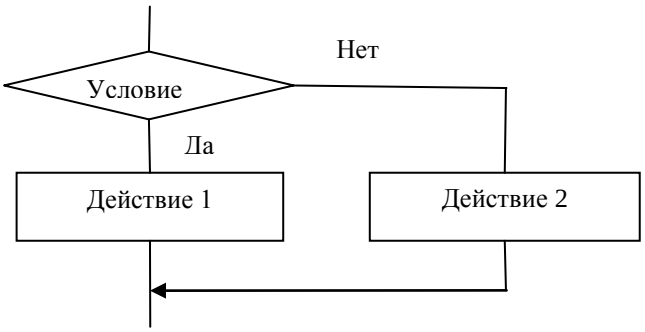


Рис.2.1. Структура ветвления

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 C++ 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:

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

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 1 представлена на рис.2.2. Начертите ее в отчете по лабораторной работе.

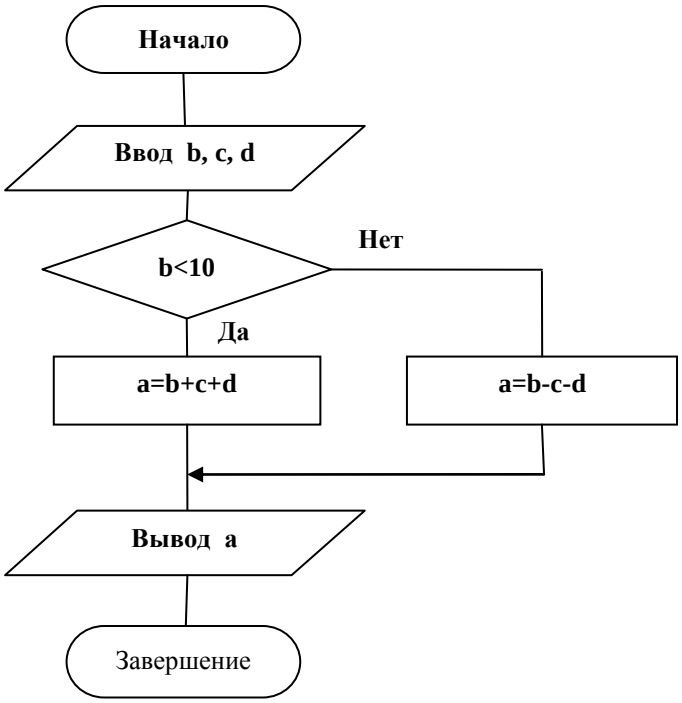


Рис. 2.2. Блок-схема алгоритма определения переменной **a** (простые операторы)

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 имен  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-приглашения  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_Лр6**. 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 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.

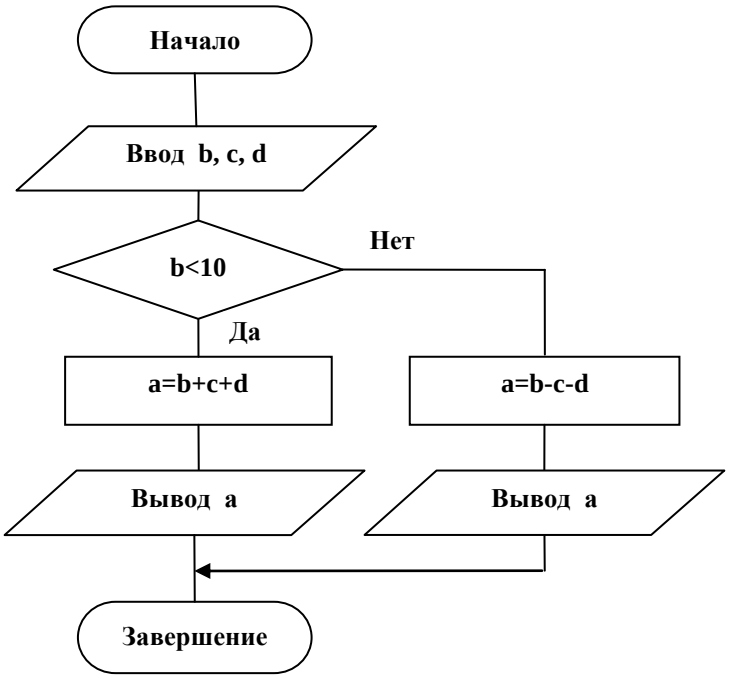


Рис. 2.3. Блок-схема алгоритма определения переменной **a** (сложные операторы)

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>                 //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
    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-приглашения  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 переменной 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:\PE-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:

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

Используйте условный оператор **if…else** и оператор консольных ввода и вывода данных аналогично **Заданию 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 **Результат_Фамилия_Лр2**.

Задание 2.4. Самостоятельно разработать алгоритм и программу для вычисления переменной **Y** (использовать оператор **if…else** и консольный ввод-вывод переменных). Определите свой номер варианта как номер компьютера.

Таблица 2.1 Исходные данные и формулы для расчета Y (Задание 2.4)

№ варианта	Формула для расчета **Y**	Значение **a**
1	$\begin{cases} Y = tg\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если} \quad x < 0{,}5; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если} \quad x \geq 0{,}5; \end{cases}$	**a=3,5**
2	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если} \quad x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если} \quad x \geq 0{,}7; \end{cases}$	**a=2,55**
3	$\begin{cases} Y = tg\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если} \quad x < 0{,}5; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если} \quad x \geq 0{,}5; \end{cases}$	**a=1,5**
4	$\begin{cases} Y = ctg\dfrac{1-x}{1+x} + \cos^2 5x, & \text{если} \quad x < 0{,}3; \\ Y = \sin^3(x+a) - tg^2(x+a), & \text{если} \quad x \geq 0{,}3; \end{cases}$	**a=1,44**
5	$\begin{cases} Y = \sin\dfrac{1-x}{1+x} + ctg^2 5x, & \text{если} \quad x < 0{,}48; \\ Y = \cos^3(x+a) - tg^2(x+a), & \text{если} \quad x \geq 0{,}48; \end{cases}$	**a=2,4**
6	$\begin{cases} Y = ctg^3(x+a) - \sin^2(x+a), & \text{если} \quad x < 0{,}7; \\ Y = tg^3(x+a) - \cos^2(x+a), & \text{если} \quad x \geq 0{,}7; \end{cases}$	**a=1,1**
7	$\begin{cases} Y = tg\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если} \quad x < 0{,}5; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если} \quad x \geq 0{,}5; \end{cases}$	**a=2,6**
8	$\begin{cases} Y = tg\dfrac{1-x}{1+x} + \sin^2 5x, & \text{если} \quad x < 0{,}5; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если} \quad x \geq 0{,}5; \end{cases}$	**a=1,9**
9	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если} \quad x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если} \quad x \geq 0{,}7; \end{cases}$	**a=3,1**
10	$\begin{cases} Y = ctg\dfrac{1-x}{1+x} + \cos^2 5x, & \text{если} \quad x < 0{,}3; \\ Y = \sin^3(x+a) - tg^2(x+a), & \text{если} \quad x \geq 0{,}3; \end{cases}$	**a=2,2**

11	$\begin{cases} Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x < 0{,}37; \\ Y = ctg^2 3x + tg^3(x+a)e^{3a}, & \text{если } x \geq 0{,}37; \end{cases}$	**a=1,1**
12	$\begin{cases} Y = ctg^2 3x + e^{3a}, & \text{если } x < 0{,}8; \\ Y = \sin^3(x+a) - tg^2(x+a), & \text{если } x \geq 0{,}8; \end{cases}$	**a=3,2**
13	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \geq 0{,}7; \end{cases}$	**a=1,2**
14	$\begin{cases} Y = tg\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}44; \\ Y = \cos^3(x+a) - tg^2(x+a), & \text{если } x \geq 0{,}44; \end{cases}$	**a=2,8**
15	$\begin{cases} Y = tg\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}5; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если } x \geq 0{,}5; \end{cases}$	**a=3,5**
16	$\begin{cases} Y = ctg^2 3x + e^{3a}, & \text{если } x < 0{,}8; \\ Y = \sin^3(x+a) - tg^2(x+a), & \text{если } x \geq 0{,}8; \end{cases}$	**a=1,3**
17	$\begin{cases} Y = \cos^3(x+a) - 7(x+a), & \text{если } x < 0{,}7; \\ Y = \sin^3(x+a) - \cos^2(x+a), & \text{если } x \geq 0{,}7; \end{cases}$	**a=1,7**
18	$\begin{cases} Y = tg\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}9; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если } x \geq 0{,}9; \end{cases}$	**a=2,6**
19	$\begin{cases} Y = ctg\frac{1-x}{1+x} + \cos^2 5x, & \text{если } x < 0{,}4; \\ Y = \sin^3(x+a) - tg^2(x+a), & \text{если } x \geq 0{,}4; \end{cases}$	**a=1,9**
20	$\begin{cases} Y = tg\frac{1-x}{1+x} + \sin^2 5x, & \text{если } x < 0{,}2; \\ Y = tg^3(x+a) - \arccos^2(x+a), & \text{если } x \geq 0{,}2; \end{cases}$	**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 число **a** делит числовую ось **x** на два интервала (рис. 2.4).

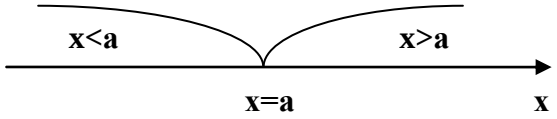


Рис. 2.4. Графическая интерпретация логического выражения для двух интервалов значений **x**

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**.

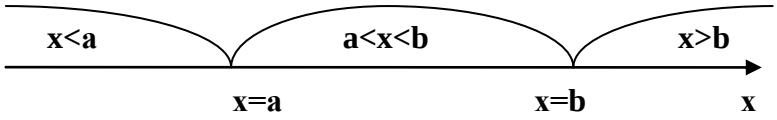


Рис. 2.5. Графическая интерпретация логического выражения для трех интервалов значений **x**

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).

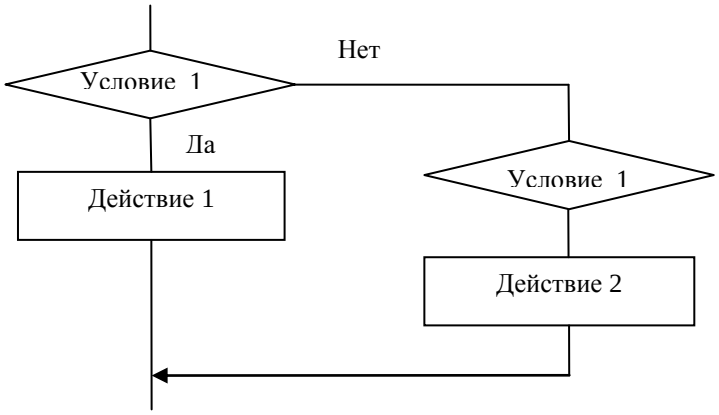


Рис.2.6. Структура вложенного условного оператора **if…else**

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-сема 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).

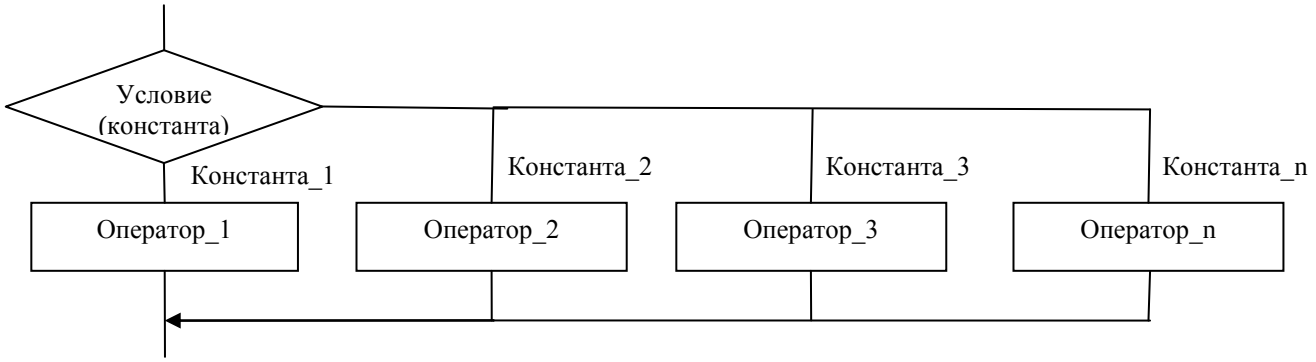


Рис. 2.7. Блок-схема алгоритма работы оператора **switch**

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.

```
#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-приглашения Vvedite a :
cin >> a;                                   //Ввод значения переменной a
cout << "Vvedite operation : "<<endl;       //Вывод на экран надписи-приглашения Vvedite operation :
cin >> operation;                           //Ввод знака операции
cout << "Vvedite b : "<<endl;               //Вывод на экран надписи-приглашения 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 переходи з 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 А.

Задание 2.6. Проанализируем работу оператора **goto** на примере программы из задания 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 ";**

```
#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
    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-приглашения  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 разработать 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 (два занятия)
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.

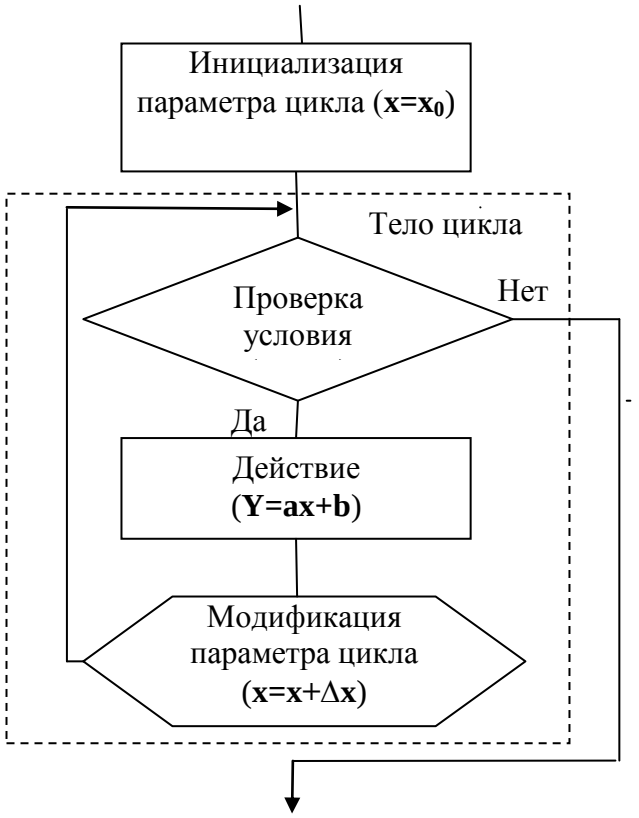


Рис. 3.1. Блок-схема алгоритма циклического вычислительного процесса

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 C++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 x_0 до x_k с шагом Δx, где x_0=1, 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);

x_0 – переменная, обозначим ее как **xn**, тип **double**;

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.

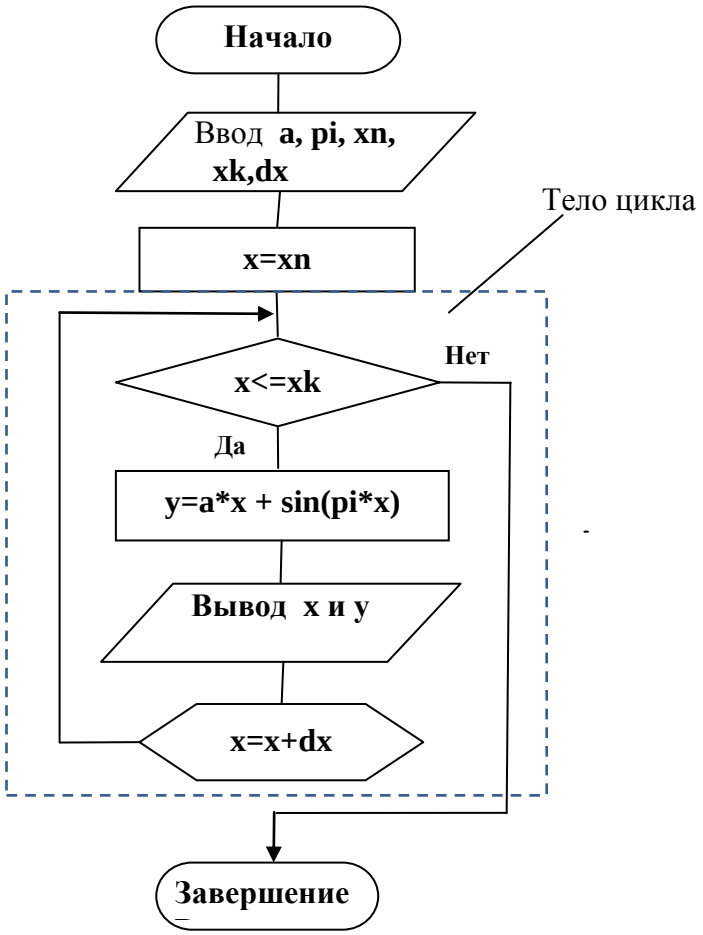


Рис. 3.2. Блок-схема алгоритма решения задания 3.1

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>                  //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
{
    const int a=3;                  //Инициализация целой константы a
    const  double  pi=3.14;         //Инициализация действительной константы  π
    double x, y, xn, xk, dx;        //переменных и параметра цикла: Y, x, x0,  xk  и Δx
    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 – предоставление 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 действительной 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 **x, a, в**
1	$Y = ctg\frac{1-3x}{1+2x} + \cos^2 5x + e^{7a}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
2	$Y = \frac{\cos^3(x+a) - 7(x+a)}{tg(x+a)^4}$	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	$Y = \frac{tg\frac{4a^2+1}{4}}{\cos^3 2x + e^{2a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
6	$Y = \sin\frac{1-x}{1+x} + tg^4 5x + e^{5a}$	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	$Y = \dfrac{ctg\dfrac{x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
9	$Y = \dfrac{ctg^3(3x+a) - \sin^2(x+7a)}{(5x+a)^3}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
10	$Y = \dfrac{\arcsin^3\dfrac{4x+1}{4}}{ctg^2 3x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
11	$Y = \dfrac{\arccos^3\dfrac{4x+1}{4}}{tg^2 3x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
12	$Y = \dfrac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
13	$Y = \dfrac{arcctg\dfrac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
14	$Y = \dfrac{tg^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	x_0=0; x_k=1,0; Δx=0,1; a=1,77
15	$Y = tg\dfrac{1-x}{1+x} + \sin^2 5x + e^{5a}$	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	$Y = \dfrac{tg^3(x+a) - \arccos^2(x+a)}{(x+a)^4}$	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 x_0=0 до x_k=1 с шагом Δx=0,2 (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);

x_0 – переменная, обозначим ее как **xn**, тип **double**;

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.

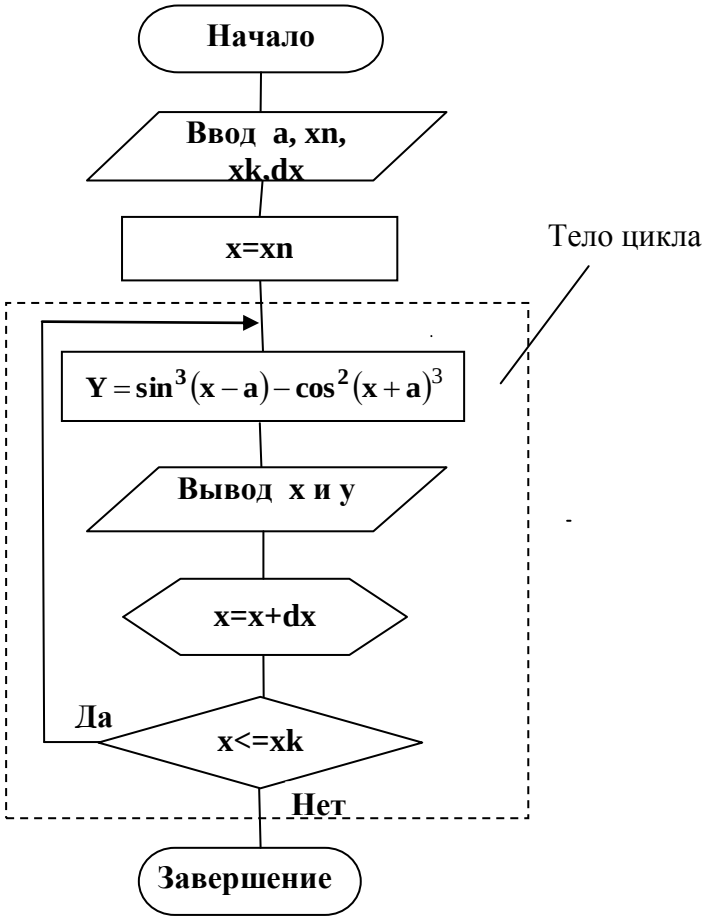


Рис. 3.3. Блок-схема алгоритма решения задания 3.3

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>                          //Файл conio.h для задержки 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
    {
    double a=0.3;                           //Инициализация константы a
    double x, y, xn, xk, dx;                //переменных и параметра цикла: Y, x, x0,  xk  и Δx
    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 – предоставление начального
                                            //значения параметру цикла - переменной 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 действ. переменной y на данном шаге
            cout<<x<<"   "<<y<<endl;       //Вывод на экран действительной переменной y
            x=x+dx;                         //Вычисление  следующего значения  параметра цикла  x
        }                                   //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 x_0, x_k, Δx и **a**,
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	$Y = 2a^3 + \frac{2x^4}{\sqrt{1 + a^2}} - x e^{-\frac{a^2+1}{2}}$	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	$Y = \frac{1+\left(\frac{x}{a}\right)^{2,5}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x^3}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
11	$Y = \left(x^3+\frac{a}{a^2+x^2}\right)-\sqrt{1+\frac{a}{a^2+x^4}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
12	$Y = \sqrt[4]{\frac{1+tg^2\frac{a^3+1}{4}}{1+\frac{x+1}{4}}}-e^{-\frac{a^2+1}{4}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
13	$Y = \frac{1+\left(\frac{x^3}{a}\right)^{2,5}+\sqrt{1+\left(\frac{x}{a}\right)^{2,5}}}{1+e^{x}+\sqrt{1+\left(\frac{x}{a^3}\right)^{2,5}}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
14	$Y = \ln\left(1+\sqrt{1+x^2}\right)+\frac{1+x^2}{\sqrt[3]{1+\frac{a^3}{2}}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
15	$Y = tg\frac{1-x}{1+x}+\sin^2 5x+e^{5a}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
16	$Y = \frac{1+\cos\frac{1-x^2}{1+x^2}}{\sqrt{1+\left(\frac{1-a^3}{1+a^4}\right)^2}}-e^{-\frac{x^2}{2}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
17	$Y = \ln\left(1+\sqrt{1+x^2}\right)+\frac{1+x^2}{\sqrt{1+\frac{a^4}{2}}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$
18	$Y = \frac{\sin^3\left(a+x^2\right)}{\sqrt{a+\frac{x^3+1}{2}+\left(\frac{x^2+1}{2}\right)^5}}$	$x_0=0;\quad x_k=1,0;$ $\Delta x=0,1;\quad a=1,77$

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**.

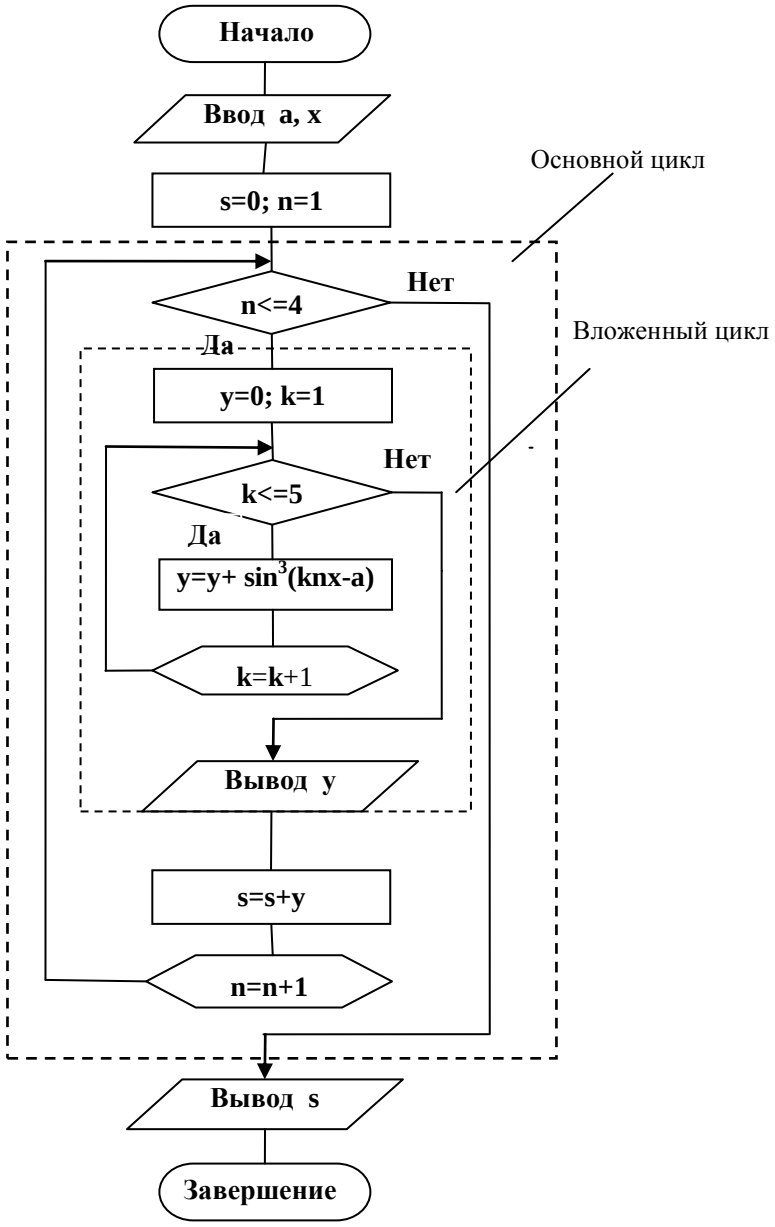


Рис. 3.4. Блок-схема алгоритма решения задания 3.5 с использованием вложенного цикла

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>                  //Файл conio.h обеспечивает задержку окна DOS на экране
#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
{
```

```
const double a=1.4;                    //Инициализация константы a
double x, y, n, k, s;                  //Инициализация переменных и параметров цикла: y,x, s, n  и k
cout<<" Vvedite x "<<endl;             //Вопрос для ввода  x
cin>>x;                                //Ввод  переменной x
cout<<"x= "<<x<<"  a= "<<a<<endl;      //Вывод исходных данных
n=1;                                   //Подготовка к циклу – предоставление 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 – вложенного цикла
    y=0;                               //Начальное значение переменной y для суммирования
    k=1;                               //Подготовка к циклу – предоставление начального
                                       //значения параметру цикла - переменной k
    while (k<=5)                       //Вложенный оператор while (условие цикла k ≤ 5)
    {                                  //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 переменной 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 переменной s на k-ом шаге
        n=n+1;                         //Вычисление  следующего значения  переменной 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**
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	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}tg^4\left(knx^3-an\right)$	**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 (два занятия)
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.

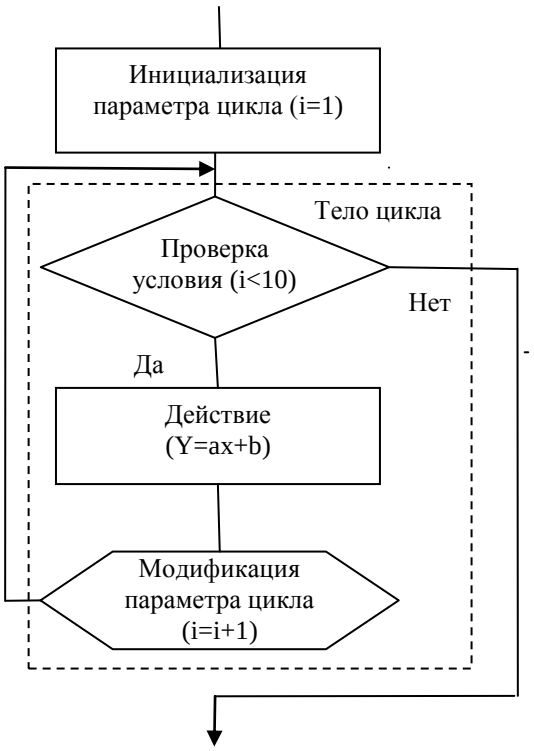


Рис. 4.1. Блок-схема алгоритма циклического вычислительного процесса

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.

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 цикал **for** отражается символом **модификация** (рис. 4.2):

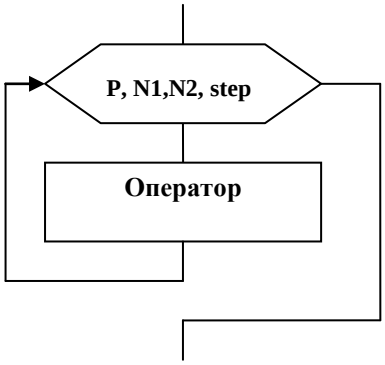


Рис. 4.2. Блок-схема оператора цикла **for**

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.

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 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.

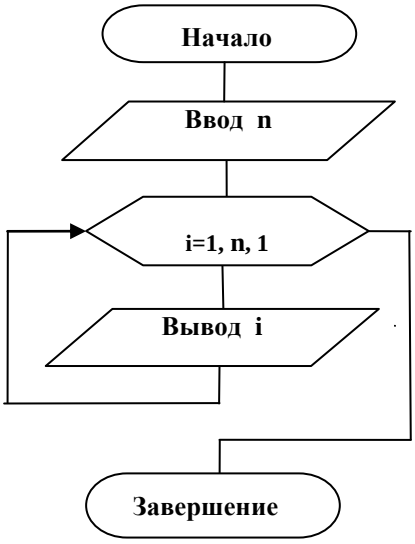


Рис. 4.3. Блок-схема алгоритма для печати в столбец **n** чисел

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. Потом 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**.

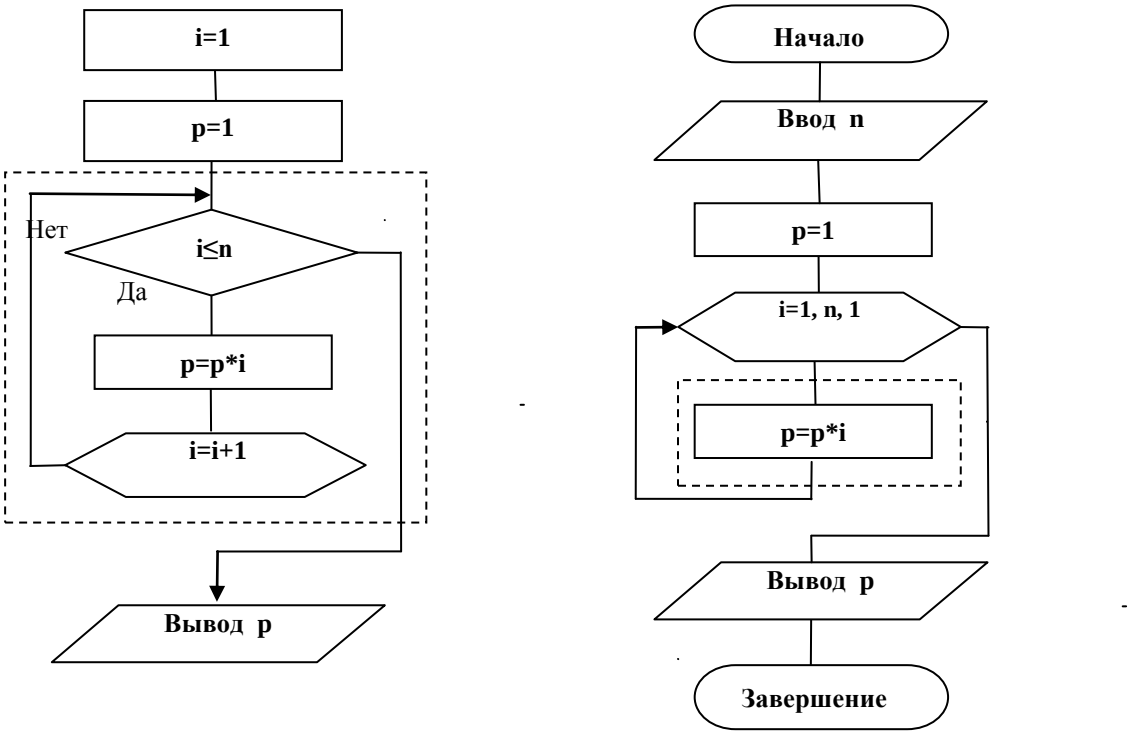
!Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

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:

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

При вычислении факториала начальному значению произведения Π необходимо присвоить 1. При каждом выполнении тела цикла Π_{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 1 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.

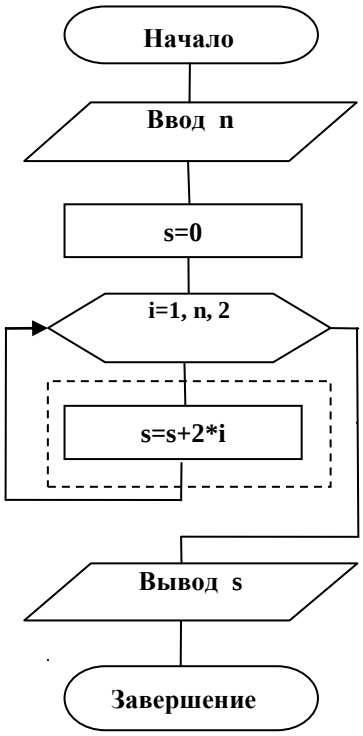


Рис. 4.5. Блок-схема алгоритма для вычисления суммы **n** четных чисел

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 0 переменной s для суммирования
        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 C++ имеет следующий вид:

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

Тогда программа для расчета функции **Y** примет вид:

```
#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 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
```

Рис.4.6. Результаты вычислений для Задания 4.4

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, 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	$Y=\sum_{k=0}^{7}\frac{\text{arcctg}\,\frac{2kx^3+1}{4}}{\cos^2 5x+e^{3a}}$	**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.

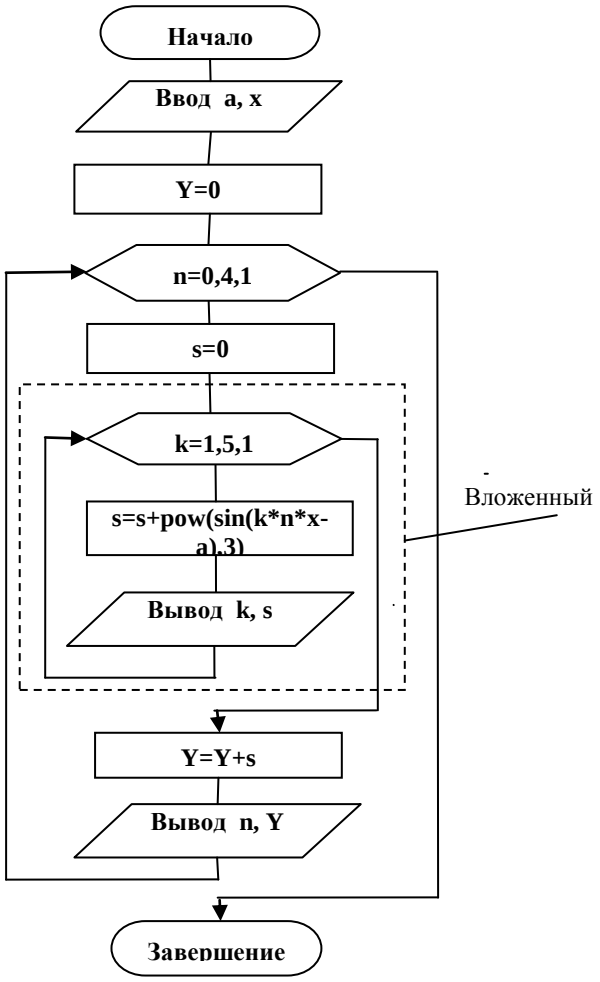


Рис. 4.7. Блок-схема алгоритма решения задания 4.6 с использованием вложенного цикла **for**

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>                          //Файл conio.h обеспечивает задержку окна DOS на экране
#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
    {
    const double x=1.4, a=2.3;              //Инициализация констант a и x
    double Y, n, k, s;                      //Объявление переменных и параметров цикла: y, s, n  и k
    cout<<"x= "<<x<<"  a= "<<a<<endl;       //Вывод исходных данных
    Y=0;                                    //Начальное значение переменной Y для суммирования
       for(n=0;n<=4;n++)                    //Внешний оператор for (условие цикла n ≤ 4)
           {                                //Начало составного оператора – вложенного цикла
           s=0;                             //Начальное значение переменной s для суммирования
              for(k=1;k<=5;k++)             //Вложенный оператор for (условие цикла k ≤ 5)
              {
                 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
```

Результаты вычислений приведены на рис. 4.7. 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
```

Рис. 4.8. Результаты вычислений для Задания 4.6

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 С++.

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

№ варианта	Формула для расчета **Y**	Значения **a** и **x**
1	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{knx^2-an}$	**a=1,2; x=1,32**
2	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}tg^4(knx^3-an)$	**a=1,3; x=1,46**
3	$s=\sum_{n=1}^{6}\sum_{k=1}^{5}\sin^4(knx^3-a)$	**a=1,4; x=1,7**
4	$s=\sum_{n=1}^{6}\sum_{k=1}^{3}\sqrt[4]{knx^3-ak}$	**a=1,5; x=1,4**
5	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^3(knx^5-a)$	**a=1,1; x=1,6**
6	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[3]{knx^2-a}$	**a=1,2; x=1,8**
7	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}tg^5(knx^3-an)$	**a=1,3; x=1,4**
8	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}\sqrt[3]{knx^4-ak}$	**a=1,4; x=1,5**
9	$s=\sum_{n=1}^{6}\sum_{k=1}^{4}tg^3(knx^5-ax)$	**a=1,5; x=1,6**
10	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{knx^3-an}$	**a=1,1; x=1,4**
11	$s=\sum_{n=1}^{4}\sum_{k=1}^{3}\sin^4(knx^6-ax)$	**a=1,2; x=1,8**
120	$s=\sum_{n=1}^{3}\sum_{k=1}^{6}tg^3(knx^2-a)$	**a=1,3; x=1,2**
13	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{knx^3-ax}$	**a=1,4; x=1,4**
14	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^4(knx^3-a)$	**a=1,5; x=1,3**
15	$s=\sum_{n=1}^{3}\sum_{k=1}^{7}\sqrt[3]{knx^5-an}$	**a=1,1; x=1,5**
16	$s=\sum_{n=1}^{5}\sum_{k=1}^{4}\sqrt[4]{knx^3-ak}$	**a=1,2; x=1,9**
17	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\sin^3(knx^6-an)$	**a=1,3; x=1,4**
18	$s=\sum_{n=1}^{4}\sum_{k=1}^{5}\cos^4(knx^3-ak)$	**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 цикла **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 (два занятия)
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. рисваивать 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.

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

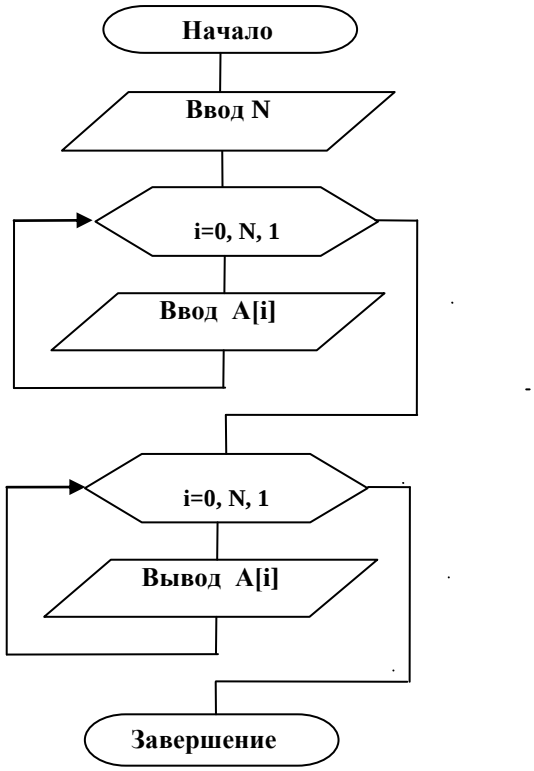


Рис. 5.1. Блок-схема алгоритма ввода и вывода элементов вектора с использованием цикла **for**

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 на экран
        cout<<"   A["<<i<<"]= "<<A[i];          //вывод i-го элемента массива  A на экран
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++ 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];

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 C++:

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[8]
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] на экран
        cout<<"  Y["<<i<<"]="<<Y[i];                       //Вывод элементов массива Y[8] на экран
        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	$Y_i = \frac{ctg^3(3x_i + a) - \sin^2(x_i + 7a)}{(5x_i + a)^3}$	**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	$Y_i = \frac{\operatorname{arcctg} \frac{x_i^3 + 1}{4}}{\cos^2 5x_i + e^{3a}}$	**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 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.

Решение. 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:

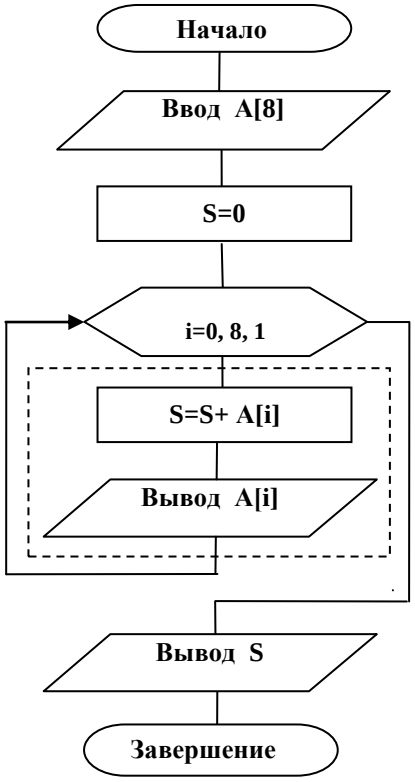


Рис. 5.2. Блок-схема алгоритма вычисления суммы элементов одномерного массива

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 переменной S для суммирования
    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] на экран
    }                                                     //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.

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

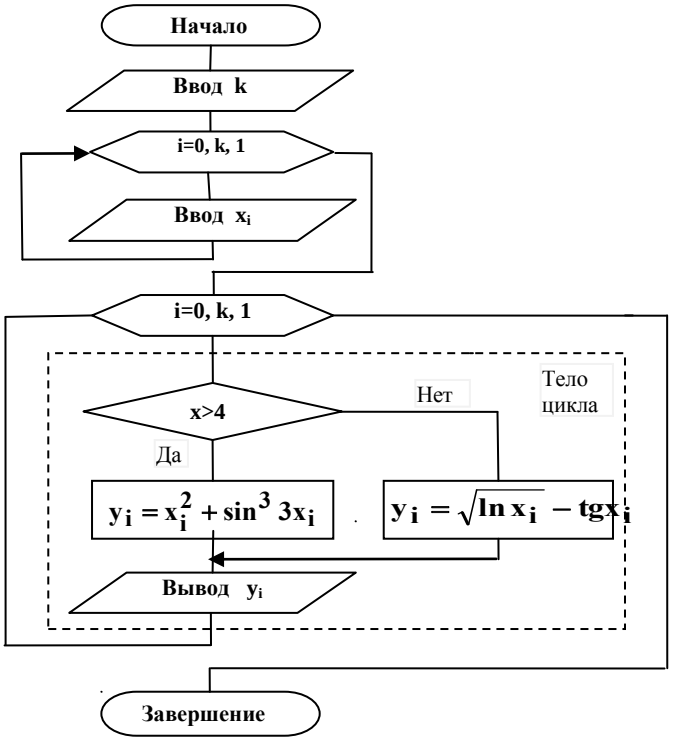


Рис. 5.3. Блок-схема алгоритма вычисления элементов одномерного массива в задании 5.6

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];                   //Исходный массив X и входной массив 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 на экран
        }                                   //Конец составного оператора
getch();
return 0;
}
```

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 C++.

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	$y_i = \begin{cases} 1.2^2 + 2\sqrt[3]{3x_i}, & \text{если } x<1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$
6	$y_i = \begin{cases} 1.2^2 + 2\sqrt[3]{3x_i}, & \text{если } x<1; \\ tg^3 x_i, & \text{если } x \ge 1; \end{cases}$	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.

Решение. 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.

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 элементов 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 операторе
            X[k]=i;                             //Записываются номера элементов, больших q,  в массив X[k]
            k=k+1;
            cout<<" >q     ";                   //Вывод знака  >q , если элемент массива A[8] больше, чем q
            }                                   //Конец составного оператора в условном операторе
        }                                       //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;
}
```

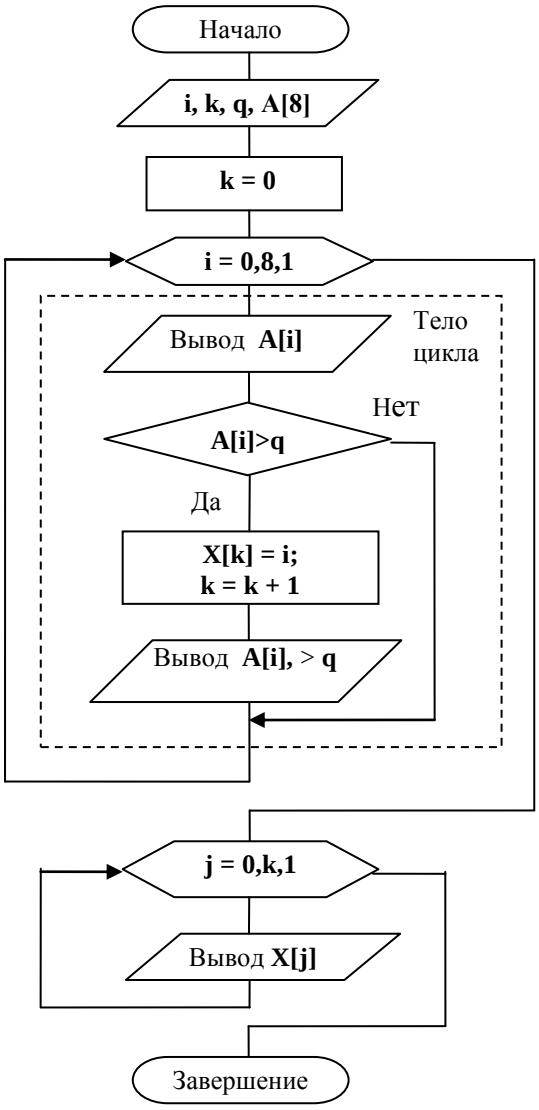


Рис. 5.4. Блок-схема алгоритма поиска элементов одномерного массива по образцу

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.

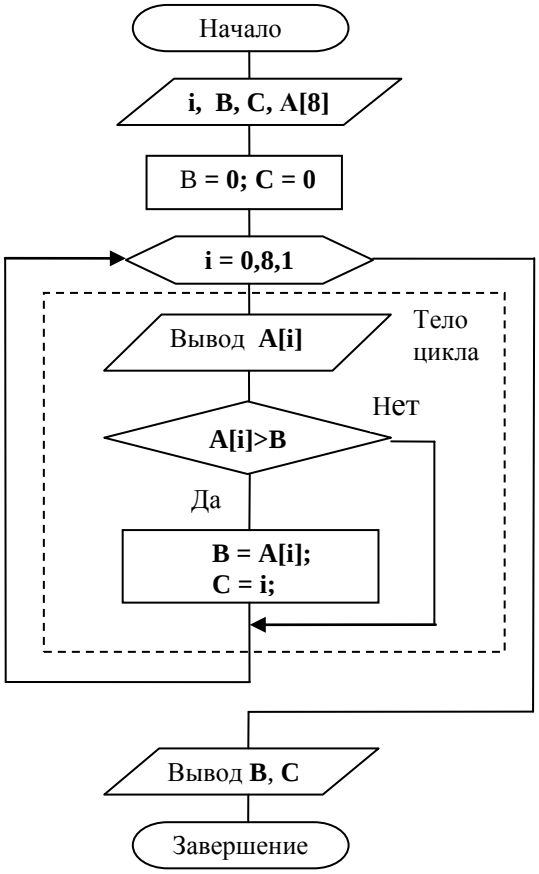


Рис. 5.5. Блок-схема алгоритма поиска максимального элемента одномерного массива

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 и индекса i
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 операторе
                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++.

Таблица 5.3. Исходные данные и формулы 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	$y_i=\begin{cases}1.2^2+2\sqrt[3]{3x_i}, & \text{если } x<1;\\ tg^3 x_i, & \text{если } x\ge 1;\end{cases}$
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 (два занятия)
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 из 24-х целых чисел по имени **A** размерностью 3×8. В памяти будет зарезервировано место для 24-х целочисленных элементов массива, которые располагаются в **непрерывном (!!!)** блоке памяти.

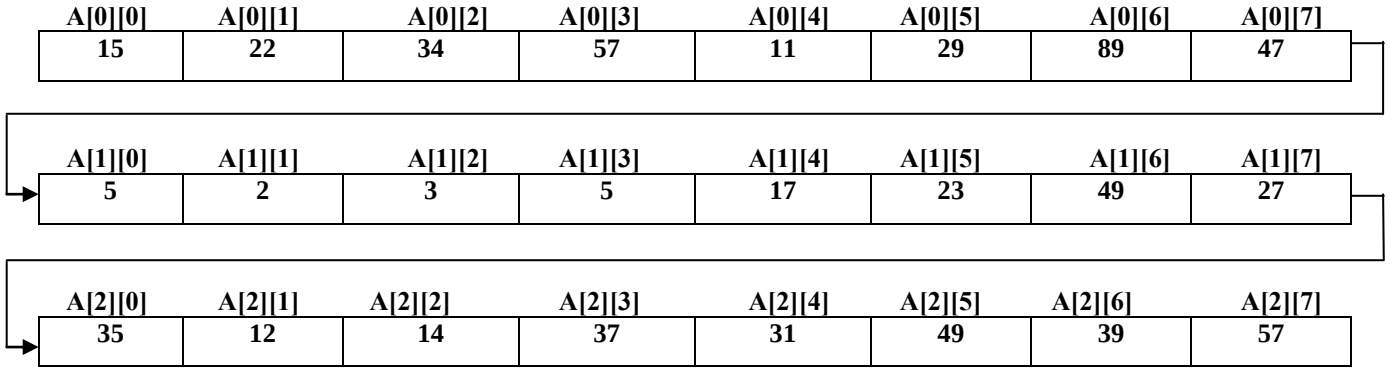


Рис. 6.1. Значения и расположение в памяти элементов массива **A[3][8]**

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

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

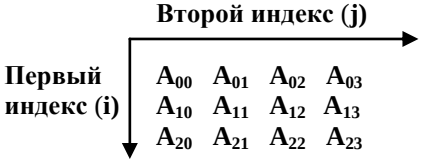
int A[2][3]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};

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, 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 двухмерных массивов в среде Visual C++ 2010

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).

Для консольного вывода двухмерного массива также надо с помощью оператора 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.

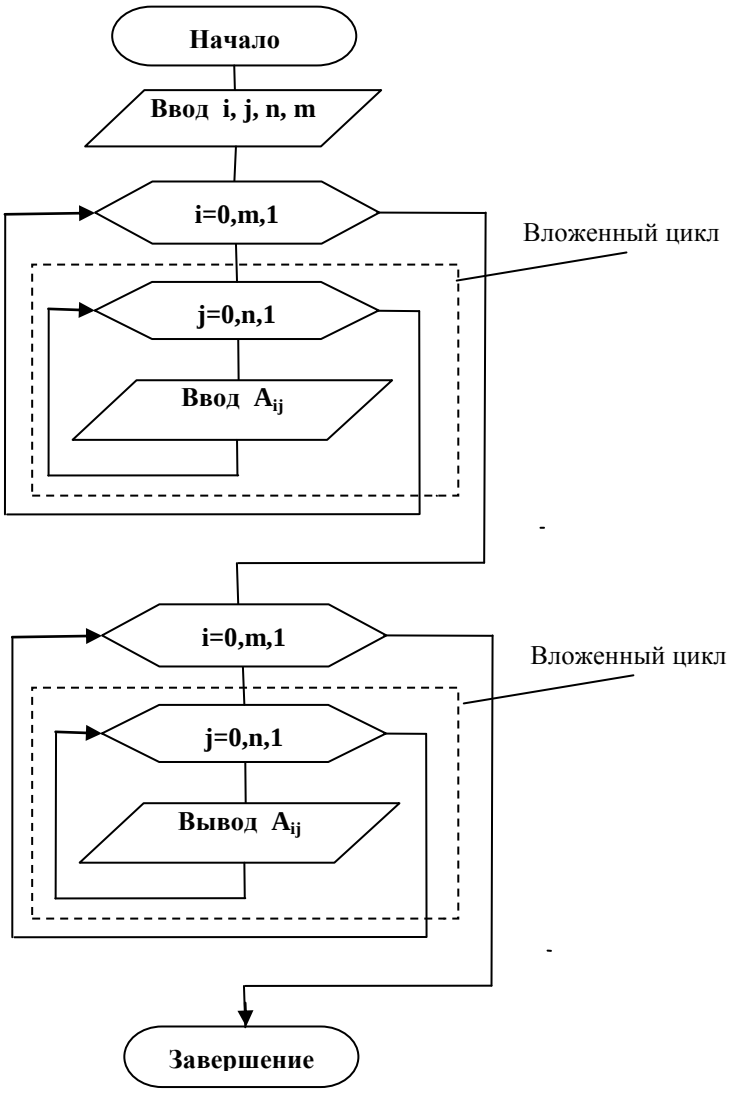


Рис. 6.1. Блок-схема алгоритма поэлементного ввода и вывода матрицы с использованием цикла **for**

Реализующая данный алгоритм программа имеет следующий вид:

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
    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 matricu:"<<endl;
    cin>>m>>n;                                    //Ввод числа строк и столбцов исходной матрицы A
    cout<<" Vvedite postrochno elementu matricu:"<<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;
    }
```

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.

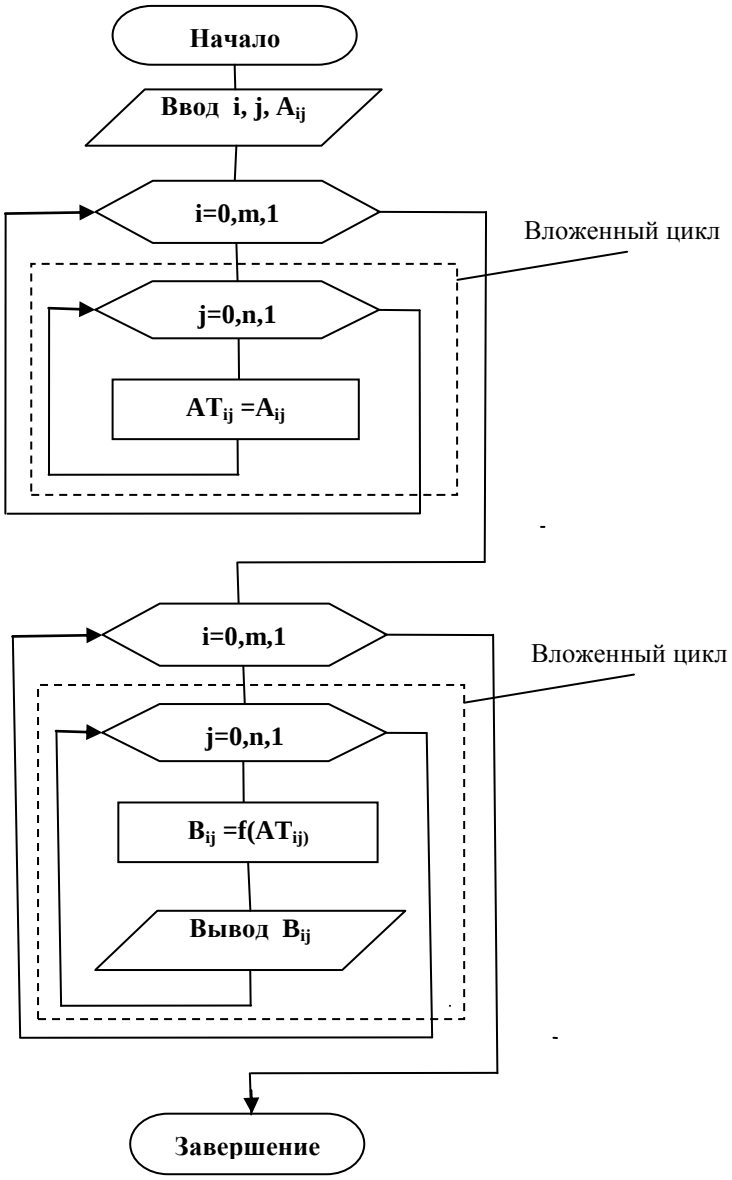


Рис. 6.2. Блок-схема алгоритма транспонирования и присваивания матрицы

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 переменных и матриц A 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 названия исходной 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;   //Вывод названия
    for(i=0; i<4; i++)                          //Начало цикла
    {
        for(j=0; j<3; j++)                      //Вложенный цикл для вывода матрицы AT
            cout<<AT[i][j]<<"  ";               //Вывод элементов матрицы AT
            cout<<endl;
    }
cout<<"Matrica B = f(AT)"<<endl;                //Вывод названия

                        //Вычисление матрицы B
    for(i=0; i<4; i++)                      //Внешний цикл для поэлементного вычисление матрицы B
    {
        for(j=0; j<3; j++)                  // Вложенный цикл для поэлементного вычисление матрицы B
        {
            B[i][j]=5*pow(sin(pow(AT[i][j],3)),2)+pow((AT[i][j]*AT[i][j]),1./3.);   // Bij = f(ATji)
            cout<<B[i][j]<<"  ";                                                    //Вывод элементов 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, которые выполняют 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++.

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.

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;                                          // Объявление переменных
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];                                    //Подготовка к поиску максимального элемента
m=0; n=0;
    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 максимального элемента
            {
            B=A[i][j];                        //Запоминание  в B большего элемента Aij
            n=i;                              //Запоминание в  n номера строки i элемента Aij
            m=j;                              //Запоминание в m номера столбца j элемента Aij
            }
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;
}
```

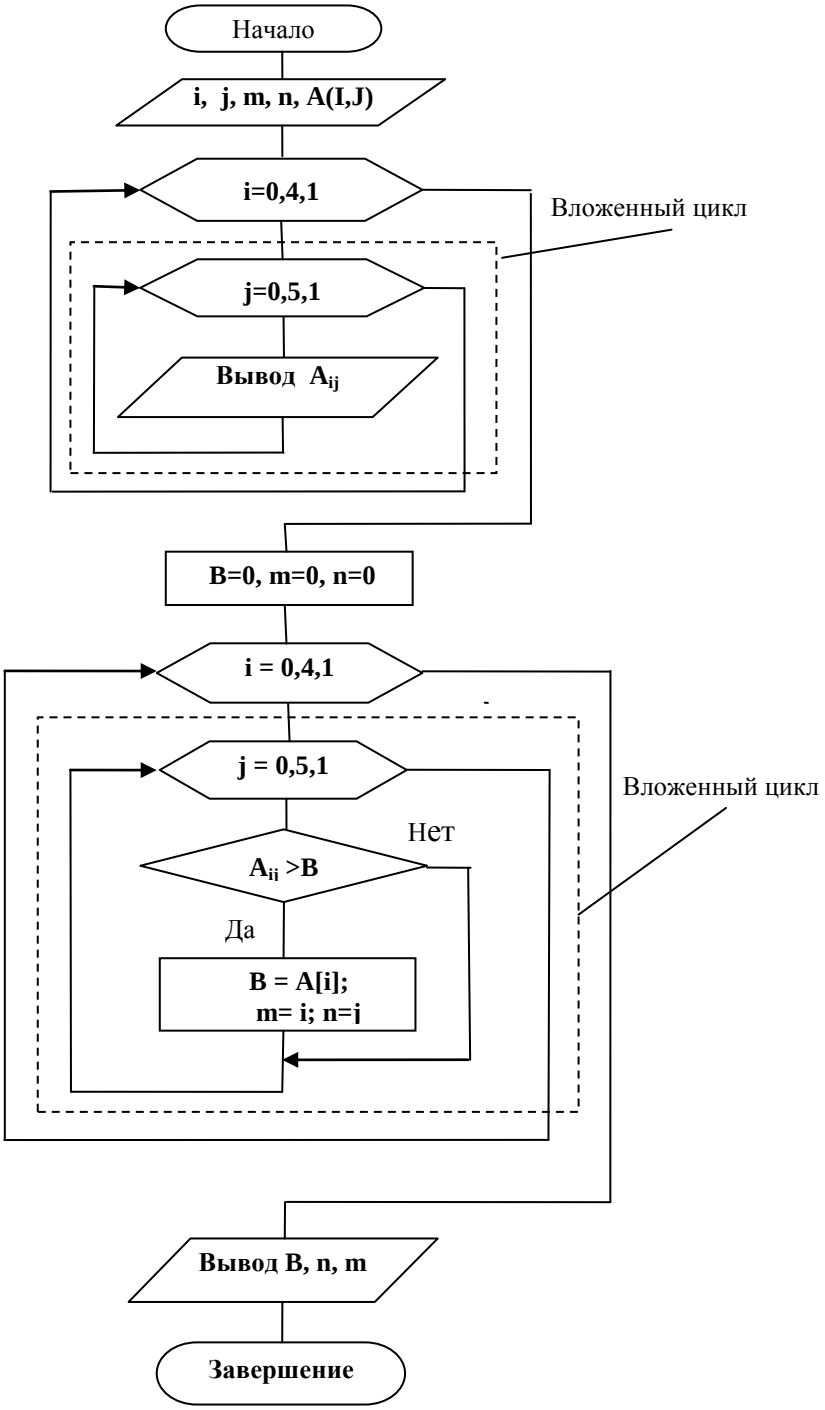


Рис. 6.3. Блок-схема алгоритма поиска максимального элемента матрицы

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 задачи 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;                    //Максимальное 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 вспомогательных переменных
cout<<"Vvedite chislo strok  i stolbcov matricu  A :"<<endl;
cin>>m>>k;                               //Ввод реальных размеров исходной матрицы A
cout<<"Vvedite postrochno elementu matricu  A :"<<endl;
    for(i=0; i<m; i++)       //Внешний цикл вывода элементов матрицы A
        for(j=0; j<k; j++)
            cin>>A[i][j];                //Ввод элементов 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 в цикле
S1=0;                                    //Задание начальных значений сумм четных
S2=0;                                    //и нечетных элементов вектора R
    //Операции по определению суммы 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 (четное или нет)
            S2=S2+R[i];                  //Определение суммы четных элементов вектора R
        else S1=S1+R[i];                 //Определение суммы нечетных элементов вектора R
    }                                    //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 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 4х4 (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. Раскрыть понятие многомерного массива?
2. Как объявляются многомерные массивы?
3. Как выполняются внешние и вложенные циклы **for**?
4. Как обратиться к элементу матрицы?
5. Как выполняются различные операции над матрицами?
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
6. Как будет выведен на экран двухмерный массив в фрагменте программы 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
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

Лабораторная работа № 7 (два занятия)
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

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) - t**main ( ).**

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 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 функции 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.

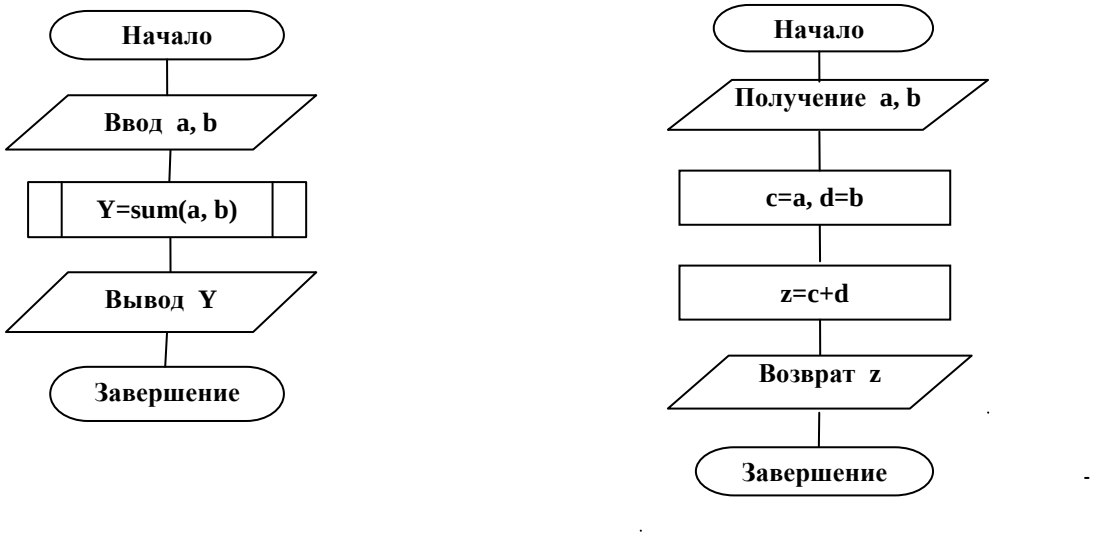
Когда происходит передача переменной-аргумента по значению, в функции создается локальная переменная с именем аргумента, в которую записывается его значение. Внутри функции может измениться значение этой переменной, но не самого аргумента.

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) функции вычисления Y
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 функции, 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 C++ и начертите блок-схему алгоритма решения поставленной задачи.

Таблица 7.1 Исходные данные и формулы для расчета Y (Задание 7.2)

№ варианта	Формула для расчета **Y**	Значения **x, a, в**
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	$Y = \frac{\arcsin^3 \frac{4x+1}{4}}{\operatorname{ctg}^2 3x + e^{3a}}$	**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	$Y = \frac{\operatorname{tg}^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	**x=0,13; a=0,7**
18	$Y = \operatorname{tg}\frac{1-x}{1+x} + \sin^2 5x + e^{5a}$	**x=0,5; a=3,5**

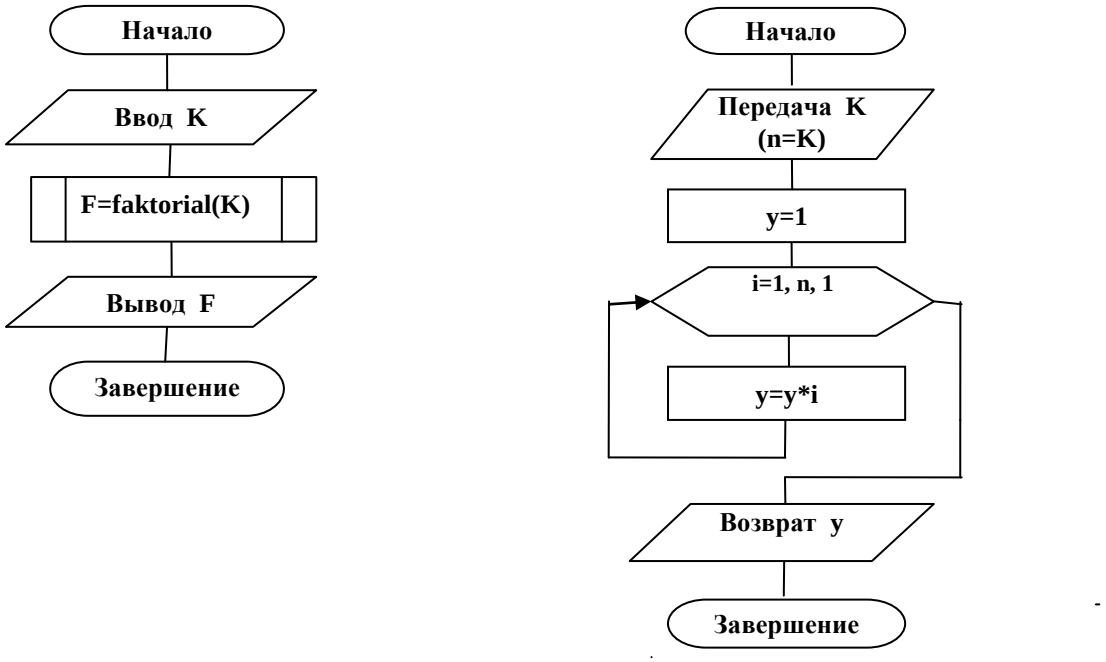
Задание 7.3. Исследовать программу, использующую функцию вычисления факториала числа:

$$F = n!\ .$$

Для составления программы данное выражение запишем следующим образом:

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

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 K
    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 числа
        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

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

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**:

```
#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 трехкратным 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 данные переменные **определены** (доступны для действий с ними 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: вычисление 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 для 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++)                     //Организация цикла 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 вычисления факториала числа
{
    int j,y;
    y=1;
    for(j=0; j<=x; j++)
    {
        if(j==0)
            y=1;
        else y=y*j;
    }
return y;                                     //Возвращение в основную программу значения  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 параметра  цикла и 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;             //Печать максимального  элемента исходного  массива
    cout<<" Novui massiv "<<endl;
    for(i=0; i<k; i++)                        //Цикл для расчета и печати 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 параметра цикла и реального размера массива
double A[N];                                    //Объявление исходного массива  A[N]
cout<<"Vvedite razmer massiva :"<<endl;
cin>>m;                                         //Ввод реального размера исходного 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;                         //Печать макс. элемента массива A (глобальная переменная)
cout<<" Nomer max "<<k<<endl;                   //Печать номера макс. элемента (глобальная переменная)
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 первый элемент
for(j=0; j<n;j++)                               //Цикл для поиска максимального элемента
    if(B[j]>C)
    {
    C=B[j];                                     //Определяем максимальный элемент
    k=j;                                        //Номер макс. элемента заносим в глобальную переменную k
```

```
   }
return 0;
}
```

После выполнения программы экран будет иметь следующий вид:

```
Uvedite razmer massiva :
8

Uvedite 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 (Задание 7.9)

№	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 (два занятия)
ИССЛЕДОВАНИЕ ВОЗМОЖНОСТЕЙ VISUAL C++ 2010 ПО ОБРАБОТКЕ МАССИВОВ СИМВОЛЬНЫХ ПЕРЕМЕННЫХ (СТРОК)

1. Описание строк в среде Visual C++ 2010

В языке C++ массив типа **char** - это одномерный массив, 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() РазмерПеременной** 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 (два занятия)
ИССЛЕДОВАНИЕ ФАЙЛОВОГО ВВОДА И ВЫВОДА ДАННЫХ В VISUAL C++ 2010. АДРЕСАЦИЯ ПЕРЕМЕННЫХ И УКАЗАТЕЛИ В СРЕДЕ VISUAL C++ 2010

Цель работы: изучение адресации памяти и исследование особенностей применения указателей в Visual C++ 2010; получение навыков разработки программ, в которых необходимая информация записывается и считывается из открываемых файлов.

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 переменной 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 коипьютера 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 **Lr9-1.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 – **9-1**. 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. 1 показана взаимосвязь между адресом переменной и указателем на этот адрес.

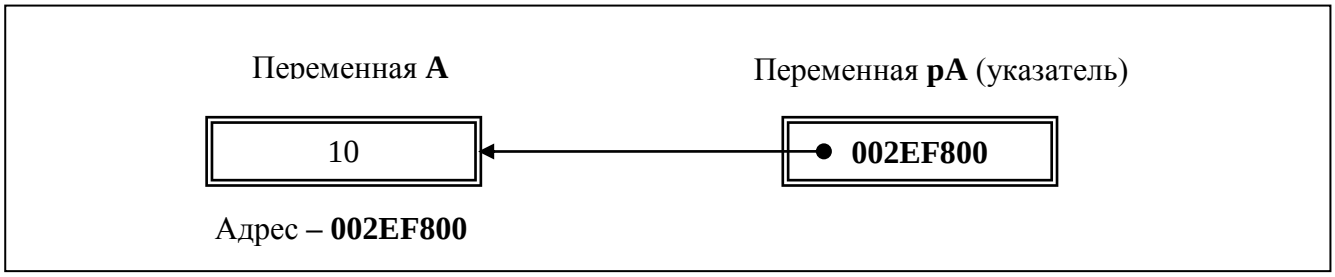


Рис. 1. Взаимосвязь между адресом переменной и указателем на этот адрес.

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**.

```
double B=340;
double *pB;                          //Объявление указателя uB
#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 адреса A
cout<<"      pA = "<<pA<<endl;       //Вывод указателя pA
cout<<"B= "<<B<<"   &B= "<<&B<<endl; //Вывод переменной B и адреса B
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 и другие операции с указателями в Visual C++ 2010

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.

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 С++ позволяет работать с указателями также, как и с переменными стандартных типов. Однако операции над указателями отличаются некоторыми особенностями.

Операция присваивания.

Указатели одного и того же типа могут использоваться в операциях присваивания, как и другие любые переменные.

Задание 9.4. 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 внесло погрешностей и значение разыменованного указателя **g** совпадает со значением переменной **x**.

3. Указатели и массивы

В языке С++ принято, что имя массива – это адрес ячейки памяти, начиная с которой располагается массив или иначе, имя массива – это адрес первого (нулевого) элемента массива.

Если объявлен массив

int A[12];

то **A** является указателем на массив, точнее на первый элемент массива.

Таким образом**,**

pA = A[0];

Для того чтобы получить значение 8-го элемента массива **A**, необходимо задать

A[8]=*(A+7) .

В С++ **n**-й элемент массива определяется как

A[n] =***(A[0]+n)**.

Задание 9.5. Исследовать программу, которая вводит в память компьютера с клавиатуры целочисленный массив, вычисляет сумму его элементов и выводит на экран эту информацию. При этом обращаться к элементам массива будем при помощи указателей, а не индексов.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
  {
  const int M=50;
    int i, m, S, Mas[M];
    int *pMas;
    pMas=Mas;
```

```
    cout<<"Vvedite razmer massiva m:"<<endl;
    cin>>m;
    cout<<"Vvedite massiv:"<<endl;
    for (i=0; i<m; i++, pMas++)
        cin>>*pMas;
    cout<<endl;
    S=0;
    pMas=pMas-m;
    for (i=0; i<m; i++, pMas++)
        S=S+*pMas;
        pMas=pMas-m;
    for (i=0; i<m; i++, pMas++)
        cout<<*pMas<<' ';
    cout<<endl<<"S= "<<S<<endl;
getch();
return 0;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
Vvedite razmer massiva m:
6
Vvedite massiv:
1 2 3 4 5 6

1 2 3 4 5 6
S= 21
```

4. Файлы и потоки ввода и вывода данных в Visual C++ 2010

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.

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;
- строчные;
- 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 **строчный 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**.

5. 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 файла, например, ***fp**, используется следующий оператор:

```
FILE  *fp;
```

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.

```
FILE *F1;
```

В файле **stdio.h** определены функции для работы с файлами. Основные из них приведены в таблице 15.1.

Таблица 9.1. 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 С++ 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.

Задание 9.6. Необходимо исследовать программу для записи переменной **A** в открываемый файл, чтения значения переменной A из этого файла и вывода значения **A** на печать.

```
#include <stdio.h>                          //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                         //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double A, B, S;
double *pS;                                 //Указатель на буфер обмена
pS=&S;                                      //Присваивание адреса S указателю pS
cout<<"Vvedite A"<<endl;
cin>>A;                                     //Ввод A
cout<<"A= "<<A<<endl;
FILE*F1;                                    //Объявление файла для записи
if((F1=fopen("D:\\Inform.DAT", "w"))==NULL) //Стандартная процедура создания и
    {                                       //открытия файла для записи данных с одно-
    cout<<"Ne mogu otkrit F1"<<endl;        //временной проверкой результата этих
    exit(1);                                //действий и сообщением в случае ошибки
    }

S=A;                                        //Запись A в буфер
fwrite(pS,4,1,F1);                          //Запись из буфера в открытый файл
fclose(F1);                                 //Обязательное закрытие файла

fread(pS,4,1,F1);                           //Чтение переменной pS из файла
B=S;                                        //Запись из буфера в переменную B
cout<<"B= "<<B;
getch();
return 0;
}
```

В данной программе используется метод определения ошибки при открытии создаваемого файла. Неоткрытие файла 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.

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);

Отлаженная программа дает следующий результат:

```
Vvedite A
45
A= 45
B= 45
```

Таким образом, значение переменной было верно записано в специально открытый файл **F1** и прочитано из этого файла.

Задание 9.7. 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.

Такая программа для записи массива в файл, а затем чтения массива из файла и отображения на экране дисплея будет иметь следующий вид:

```
#include <stdio.h>                          //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                         //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                             //Максимальный размер массива
int A[N];                                   //Объявление массива A[N]
int i,n;                                    //Параметр цикла i и реальный размер массива n
int S;                                      //Объявление переменной S в качестве буфера обмена
int *pS;                                    //Указатель pS на буфер обмена S
pS=&S;                                      //Присваивание адреса переменной S указателю pS
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                                     //Ввод реального размера массива A[N]
cout<<"Vvedite massiv A:"<<endl;
for(i=0;i<n;i++)                            //Цикл для ввода массива A[N]
cin>>A[i];                                  //Ввод i-го элемента массива A[N]
cout<<" Massiv A: "<<endl;
for(i=0;i<n;i++)                            //Цикл для вывода массива A[N]
cout<<A[i]<<" ";                            //Вывод i-го элемента массива A[N]
FILE*F1;                                    //Объявление переменной–указателя файла
    if((F1=fopen("D:\\Inform.dat", "w"))==NULL) //Открытие файла для записи с условием вывода на
    {                                       // экран предупреждения в случае сбоя при открытии
        cout<<"Ne mogu otkrit F1"<<endl;
        exit(1);
    }
for(i=0;i<n;i++)                            //Цикл для записи в открытый файл массива A[N]
{
    S=A[i];                                 //Запись i-го элемента массива A[N] в буфер S
    fwrite(pS,4,1,F1);                      //Запись i-го элемента массива A[N] из буфера в файл
}
fclose(F1);                                 //Закрытие файла
int B[N];                                   // Объявление нового массива B[N]
FILE*F2;                                    //Объявление переменной–указателя файла
if((F2=fopen("D:\\Inform.dat","r"))==NULL)  //Открытие файла для чтения с условием вывода на
{                                           //экран предупреждения в случае сбоя при открытии
    cout<<"Ne mogu otkrit F2"<<endl;
    exit(1);
}
for(i=0;i<n;i++)                            //Цикл для чтения из файла элементов массива B
{
    fread(pS,4,1,F2);
    B[i]=S;                                 //Запись единицы данных из буфера S 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[N]
    cout<<B[i]<<" ";                        //Вывод на экран i-го элемента массива B
cout<<endl;
getch();
return 0;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
Vvedite razmer n:
6
Vvedite 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.

При чтении данных из файла размер файла часто неизвестен. Для определения конца файла служит функция **feof()**. Она имеет следующий прототип:

int feof (FILE *F1);

Функция возвращает значение «истина», если достигнут конец файла и «ложь» - если нет.

Исследуем применение этой функции для файлового ввода и вывода

Задание 9.8. 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 осуществить 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 <stdio.h>                          //Подключение библиотеки программ файлового ввода-вывода
#include <stdlib.h>                         //Подключение библиотеки стандартных операций
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
const int N=30;                                 //Максимальный размер массива
int A[N];                                       //Объявление массива A[N]
int i,n;                                        //Параметр цикла i и реальный размер массива n
int S;                                          //Объявление переменной S в качестве буфера обмена
int *pS;                                        //Указатель pS на буфер обмена S
pS=&S;                                          //Присваивание адреса переменной S указателю pS
cout<<"Vvedite razmer n:"<<endl;
cin>>n;                                         //Ввод реального размера массива A[N]
cout<<"Vvedite Mas:"<<endl;
for(i=0;i<n;i++)                                //Цикл для ввода массива A[N]
    cin>>A[i];                                  //Ввод i-го элемента массива A[N]
cout<<" Massiv A: "<<endl;
for(i=0;i<n;i++)                                //Цикл для вывода массива A[N]
    cout<<A[i]<<" ";                            //Вывод i-го элемента массива A[N]
FILE*F1;                                        //Объявление переменной–указателя файла
    if((F1=fopen("D:\\ Inform.dat ", "w"))==NULL)  //Открытие файла для записи с условием вывода на экран
    {                                           //предупреждения в случае сбоя при открытии
        cout<<"Ne mogu otkrit F1"<<endl;
        exit(1);
    }
for(i=0;i<n;i++)                                //Цикл для записи в открытый файл массива A[N]
    {
    S=A[i];                                     //Запись i-го элемента массива A[N] в буфер S
    fwrite(pS,4,1,F1);                          //Запись i-го элемента массива A[N] из буфера в файл
     }
fclose(F1);                                     //Закрытие файла
float B[N];                                     // Объявление нового массива B[N]
FILE*F2;                                        //Объявление переменной–указателя файла
    if((F2=fopen("D:\\Inform.dat","r"))==NULL)   //Открытие файла для чтения с условием вывода на экран
    {                                           //предупреждения в случае сбоя при открытии
    cout<<"Ne mogu otkrit F2"<<endl;
    exit(1);
    }
i=0;
while(!feof(F2))                            //Условие проверки конца файла в цикле при чтении данных
{                                           //с применением функции определения конца файла feof()
fread(pS,4,1,F2);                           //Чтение единицы данных из файла в буфер S
    B[i]=S;                                 //Запись единицы данных из буфера S в i-й элемент массива B
    i=i+1;
 }
n=i-1;                                      //Вычисление количества прочитанных чисел
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;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
Vvedite razmer n:
6
Vvedite Mas:
2 7 4 9 6 5
 Massiv A:
2 7 4 9 6 5
 New massiv B:
2 7 4 9 6 5
```

Очевидно, что данные (элементы массива) были без искажений записаны и считаны из созданного для этого файла.

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. Как объявляется указатель с именем **pA** в языке С++?
    1. **int *pA** 2. **int «pA** 3. **int &A**
13. Как обозначается операция взятия адреса переменной **A** в языке С++?
    1. ***pA** 2. **«pA** 3. **&A**
14. Как обозначается операция разыменования указателя **pA** в языке С++?
    1. ***pA** 2. **«pA** 3. **&pA**
15. 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
16. 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 значение **X**
    3. Будет выдано сообщение об ошибке с указанием на последнюю строку

ОГЛАВЛЕНИЕ