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

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

“Інформаційні технології”

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

для студентів напряму підготовки 6.050702 ” Електромеханіка ”,
галузь знань 0507 ” Електротехніка та електромеханіка ”

Розроблено та надруковано доц. Симбірським Г.Д.

Харків, 2018

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

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

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

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.

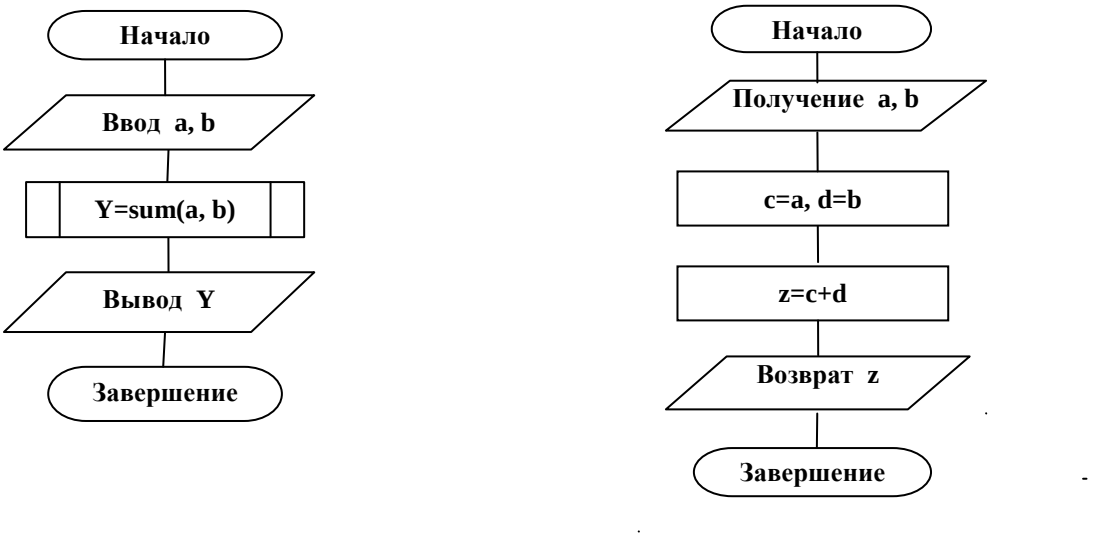
Когда происходит передача переменной-аргумента по значению, в функции создается локальная переменная с именем аргумента, в которую записывается его значение. Внутри функции может измениться значение этой переменной, но не самого аргумента.

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 С++ 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	$Y = \dfrac{\arcsin^3 \dfrac{4x+1}{4}}{\operatorname{ctg}^2 3x + e^{3a}}$	**x=0,49; a=3,7**
15	$Y = \dfrac{\sin^3(x+a) - \cos^2(x+a)}{(x+a)^4}$	**x=0,83; a=4,7**
16	$Y = \dfrac{\operatorname{arcctg} \dfrac{2x^3+1}{4}}{\cos^2 5x + e^{3a}}$	**x=0,37; a=2,75**
17	$Y = \dfrac{\operatorname{tg}^3(x+a) - 5(\sin x + a)}{\sin^3(x+a)^4}$	**x=0,13; a=0,7**
18	$Y = \operatorname{tg} \dfrac{1-x}{1+x} + \sin^2 5x + e^{5a}$	**x=0,5; a=3,5**

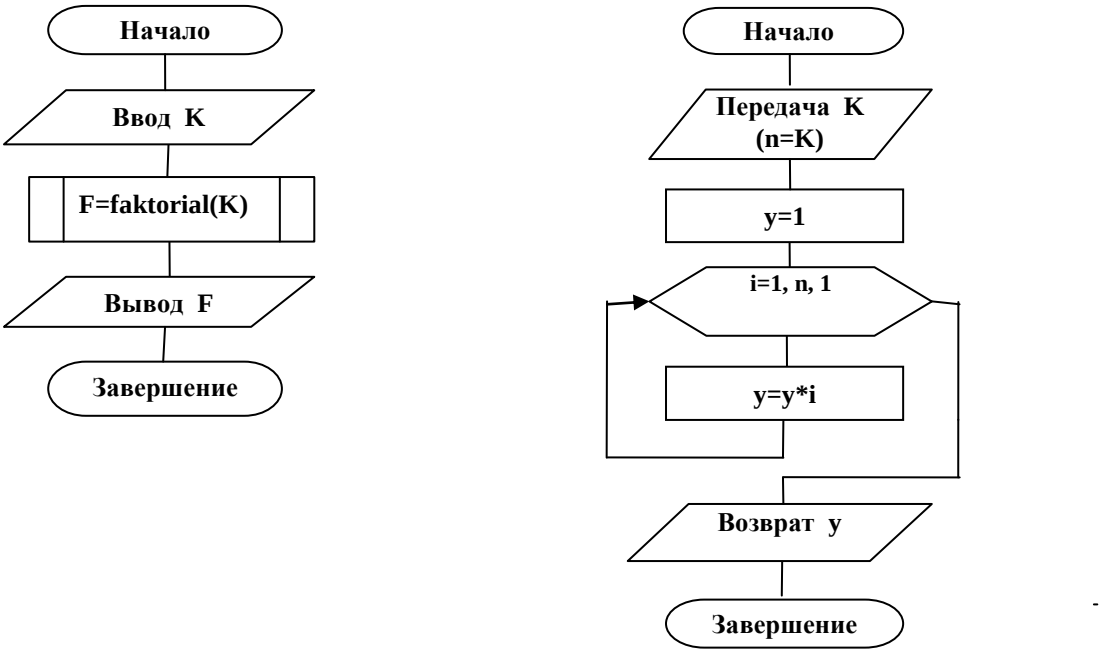
Задание 7.3. Исследовать программу, использующую функцию вычисления факториала числа:

$$F = n!\ .$$

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[][]);**