5557

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. Уткина

АЛГОРИТМИЧЕСКИЕ ЯЗЫКИ И ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам

Рязань 2020

УДК 681.3.06
Алгоритмические языки и программирование: методические указания к лабораторным работам / Рязан. гос. радиотехн. ун-т;
Сост.: С.И. Лаврентьев, С.Ю. Жулева. Рязань, 2020. 50 с.

Содержат описание операторов, алгоритмов и набор заданий по темам: “Проектирование программ простейшей структуры”, “Реализация разветвляющихся алгоритмов”, “Проектирование итерационных циклических процессов”, “Циклов со счетчиком и вложенных циклических процессов”, “Работа с массивами”, “Знакомство с пользовательскими функциями”.

Предназначены для обучающихся на образовательных программах в городской школе программистов РГРТУ

Рецензент: кафедра вычислительной и прикладной математики Рязанского государственного радиотехнического университета (зав. кафедрой д-р техн. наук, проф. Г.В. Овечкин).

Алгоритмические языки и программирование:
методические указания к лабораторным работам
Составители Лаврентьев Сергей Иванович,
Жулева Светлана Юрьевна

Рязанский государственный радиотехнический университет
имени В.Ф. Уткина
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТУ.

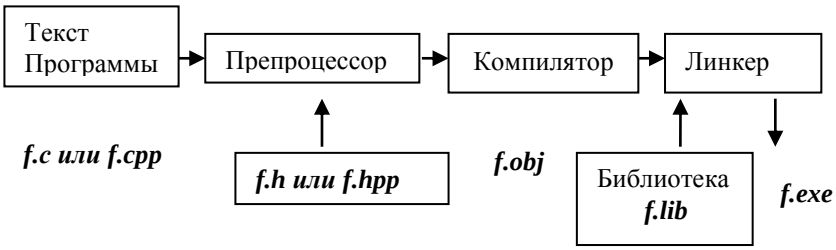
Лабораторная работа № 1

ПРОЕКТИРОВАНИЕ ПРОГРАММ ПРОСТЕЙШЕЙ СТРУКТУРЫ

Цель работы: изучение базовых элементов текста программы, основных типов данных и операций с ними.

Методические указания

Схема создания программы на языке С++



Препроцессор – программа предварительной обработки текста нашей программы перед компиляцией. На этапе компиляции компилятор преобразует код программы в машинный код. Линкер - программа, которая производит компоновку и принимает на вход один или несколько файлов и собирает по ним исполняемый файл, подключая реализации стандартных функций из библиотек. Библиотеки- это сборники подпрограмм ***f.lib***.
f.h и ***f.hpp*** - заголовочные файлы (в них хранятся заголовки стандартных функций).

Структура программы на С++

```
#include
#define          //директивы препроцессора
void main ( ) //- заголовок, главная программа
{
Операторы    //- тело программы
}
```

Директивы препроцессора

Директива **#include “имя файла”** позволяет включить в текст программы содержимое указанного файла. Имя файла может быть указано в кавычках, тогда он должен находиться в текущем каталоге (#include “anyfile.h”), иначе указывают путь к нему. Например, #include “c:\borlandc\bin\file.h”. Если имя файла указано < >, то директива использует файлы библиотечных функций из специального каталога **INCLUDE** (#include <iostream.h>).

Директива #include <iostream.h> используется для организации объектного ввода и вывода данных, а директива #include <stdio.h> используется для организации форматированного ввода/вывода данных.

Директива #include <math.h> использует заголовки математических функций, находящихся в файле math.h.

Директива **#define ИМЯ Текст замены** позволяет задать имя для некоторого фрагмента текста, такое обозначение называется макроопределением. Перед компиляцией программы **ИМЯ** во всем тексте файла заменяется на **Текст замены**.

```
#define PI 3.14
#define PodNalog 13%
```

Идентификаторы

Для обозначения имен (идентификаторов) применяются латинские буквы, цифры и знак подчёркивания. В написании имен необходимо учитывать регистр, т.к. заглавные и строчные буквы воспринимаются как разные (язык чувствителен к регистру символов). Длина идентификатора не больше 32 символов. При описании переменной задаётся её тип и имя. При необходимости после имени можно задать начальное значение переменной, это называется инициализацией переменной.

```
int i, j = 0, k;
float x = 1.0,y, z;
char c;
```

Стандартные типы данных

Целые типы данных

Знаковый тип		
тип	**объем**	**диапазон значений**
char	1 байт	-128 ÷ 127
short	2 байта	-32768 ÷ 32767
int	4 байта	-2147483648 ÷
long	4 байта	-2147483648 ÷
Беззнаковый тип		
тип	**объем**	**диапазон значений**
unsigned	1 байт	0 ÷ 255
unsigned	2 байта	0 ÷ 65535
unsigned int	4 байта	0 ÷ 4294967295
unsigned	4 байта	0 ÷ 4294967295

Вещественные типы данных

тип	**объе**	**диапазон**	**точность**
float	4	$3.4 \cdot 10^{-38}$ ÷	7 значащих
double	8	$1.7 \cdot 10^{-308}$ ÷	16 значащих
long	10	$3.4 \cdot 10^{-4932}$ ÷	20 значащих

Константы

Описание констант от описания переменных отличается ключевым словом *const* и обязательной инициализацией значения выражения:

const double pi = 3.14;

Представление констант

Значения данных задают с помощью константы. Целые константы делятся на десятичные, восьмеричные, шестнадцатеричные и длинные.

Десятичная целая константа (простое число) состоит из цифр от 0 до 9 и не начинается с нуля (например, 26, 11, 1997), а длинная заканчивается буквой *l* (например, 26l).

Восьмеричная целая константа состоит из цифр от 0 до 7 и начинается с нуля, например, 0347, ее десятичное значение равно

$3 \cdot 8^2 + 4 \cdot 8^1 + 7 \cdot 8^0 = 231$.

Шестнадцатеричная целая константа состоит из цифр от 0 до 9 и от A до F (или от a до f) (символы от A до F или от a до f обозначают цифрами от 10 до 15). Она начинается парой символов 0X или 0x, например: число 0x 1b 2f, значение которого равно

$1 \cdot 16^3 + 11 \cdot 16^2 + 2 \cdot 16^1 + 15 \cdot 16^0 = 6959$.

Вещественные числа представляются так:

5.92, 3.1415926, .358E-6.

Символьная константа состоит из одного символа, заключенного в апострофы:’a’, ’ж’, ’2’, ‘?’.

char c=’a’ (в с записывается код символа).

Для специальных символов, в частности не имеющих графического представления, определены следующие константы:

’\a’ - звуковой сигнал;

’\n’- переход в начало следующей строки;

’\r’ - переход в начало текущей строки;

’\t’ - горизонтальная табуляция;

’\v’ - вертикальная табуляция;

’\\’ - обратный слеш;

’\’’ - апостроф;

’\’’’ - кавычка;

’\0’- признак окончания строки.

Строковая константа состоит из последовательности символов, заключенных в кавычки (например, ″a b c d 2 5″). Даже один символ в кавычках - это строка, т.к. в конце неявно вставляется ’\0’ – символ.

Стандартные математические функции

Заголовки большинства математических функций находятся в файле **<math.h> ,** который необходимо включить в программу.

Имя функции	**Тип аргумента**	**Тип результата**	**Описание**
abs(x)	int	int	возвращает абсолютную величину
acos(x)	double	double	арккосинус ($-1 \le x \le 1$)
asin(x)	double	double	арксинус (($-1 \le x \le 1$)
atan(x)	double	double	арктангенс (($-1 \le x \le 1$)
atan2(x,y)	double	double	арктангенс от значения x/y ($-1 \le x \le 1$)
ceil(x)	double	double	округление до ближайшего большего целого числа
cos(x)	double	double	косинус (x - в радианах)
exp(x)	double	double	вычисление экспоненты e^x
fabs(x)	double	double	абсолютная величина (числа с

			плавающей точкой)
floor(x)	double	double	округление до ближайшего меньшего целого числа
fmod(x,y)	double	double	вычисление остатка от деления нацело x/y
ldexp(x,e)	double int	double	умножение числа с плавающей точкой на целую степень двух (**x*2^{e}**)
modf(x,p)	double	double	извлекает целую и дробную части (с учетом знака) из числа с плавающей точкой
pow(x,y)	double	double	результат x^y
pow10(x)	int	double	результат 10^x
sin(x)	double	double	синус (x - в радианах)
log(x)	double	double	результат ln(x) логарифм по основанию e=2.71
log10(x)	double	double	результат lg(x) логарифм по основанию 10
sinh(x)	double	double	гиперболический синус
cosh(x)	double	double	гиперболический косинус
sqrt(x)	double	double	квадратный корень
tan(x)	double	double	тангенс (x - в радианах)
tanh(x)	double	double	гиперболический тангенс

Ввод-вывод данных

Для организации ввода данных используются стандартный поток ввода ***cin*** и операция извлечения из потока >>. Для вывода данных используются стандартный поток ***cout*** и операция помещения в поток <<. Использование этих средств возможно после директивы **#include <iostream.h>**.

```
#include <iostream.h>
void main()
{ int i;
  cout<<”Введите целое число 4 ”;
// Строка “Введите целое число 4 ” появится на экране
  cin>>i;// Ввод с клавиатуры
```

```
  cout<<”Вы ввели число”<<i<<’\n’;
// на экране “Вы ввели число 4” и переводится строка
}
```

Для форматированного вывода применяется функция ***printf***; для её использования надо включить директиву # include <stdio.h>.

printf(“%5d”,i) – для вывода числа i используется 5 позиций.

printf(“%ш.тf “,x); **т** – точность (количество цифр после точки), **ш** - ширина поля (общее количество позиций, отводимых на число). Ширина поля должна быть больше точности. Для вывода числа могут использоваться следующие управляющие символы (спецификаторы преобразования):

%d - десятичное целое;
%o – восьмеричное число;
%x - шестнадцатеричное число;
%e - в научной нотации;
%f - вещественное число;
%g -в научной нотации либо в естественном представлении;
%c – символ;
%s – строка символов;
%u – беззнаковое целое.

Для форматированного ввода используется функция ***scanf,*** для её использования надо включить директиву # include <stdio.h>. Например, ***scanf(“%d”,&i),*** где ***&*** представляет физический адрес переменной ***i.***

Приведение типов данных

Если в операции присутствует вещественный операнд, результат приводится к вещественному значению; если все операнды целые, то результат приводится к целому значению.

```
float x;
x=3/7; // При выводе получим 0.

float x;
x=3.0/7; // При выводе получим ≈ 0.42

float x;
x=3.0/7*5; // При выводе получим ≈ 2.14
```

Для явного преобразования к нужному типу используется операция (тип) операнд.

```
float x;
x=(float)3/7; // При выводе получим ≈ 0.42
```

Пример.

Составить программу для вычисления значения целочисленной переменной y по формуле $y = x^2 + 0{,}08$ *при* $x = 1$.

Программа имеет следующий вид:

```
# include <stdio.h>
 void main ( )
{
  const  int x = 1;
  float y;
  printf (″\n Исходный параметр x = % u″, x);
  y = x*x + 0.08;
  printf (″\n Результат y = % f ″, y);
}
```

В операторах вызова функции ***printf*** спецификаторы преобразования ***%f*** и ***%u*** означают форматы вывода действительного и целого числа. Выполнение программы завершается выдачей на экран следующих строк:

Исходный параметр x = 1

Результат y = 1.08

Контрольные вопросы

1. Какова структура программы на языке C++?
2. Каковы правила записи имен программных объектов?
3. Какие типы констант вам известны?
4. Какие стандартные типы значений вам известны?
5. Какие арифметические операции вы знаете?
6. Каков порядок выполнения арифметических операций?
7. Назовите простейшие виды операторов ввода/вывода.
8. Каков порядок автоматического преобразования типов?
9. Запишите примеры операторов форматированного ввода и вывода.

Варианты заданий

Составить программу вычисления значения функции $y = f(x, a, b)$ В программе реализовать ввод исходных данных с клавиатуры и вывод значения функции и промежуточных данных на экран.

1. $y = \sqrt{x + \frac{a \cdot \cos(bx^5)}{b + \sin(ax^{\frac{1}{2}})}}$, где $= \sqrt{\frac{1}{2}x + 4}, b = (-x + 3)^3$
2. $y = \frac{a^5\sqrt{x^3 + 2}}{2\cos(b)}$, где $a = \frac{1}{\cos(x)}, b = \frac{x}{1 + x^2}$
3. $y = 1 - e^{-ax}\sin(ax + b)$, где $a = e^{-x}, b = \frac{x}{12}$

4. $y = \lg_{10}(arctg(bx) - \sin(ax))$, где $a = 10\sqrt{x}, b = x^2 + 0{,}567 \cdot 10^{-5}$

5. $y = \frac{a}{\cos(x)} + \ln(\frac{b}{\sin(x)})$, где $a = 10^{x+\ln(x)}, b = \frac{x}{1+x}$

6. $y = \frac{x^2 + bx + a}{1 + \frac{bx^2 + a}{\sin(x)}}$, где $a = tg\sqrt[20]{x}, b = \cos(\frac{x}{1+x})$

7. $y = tg(ax) + \frac{\sin^3(x)}{b}$, где $a = \sqrt{(x+b)^2 - 0{,}25}, b = \frac{\pi + x}{x^2}$

8. $y = x^2 - b - \sqrt[a]{\pi - x}$, где $a = \ln(x), b = \frac{1}{\cos(x)}$

9. $y = \cos(\frac{\sin(1{,}25a + x) + tg(x)}{tg^2 bx})$, где $a = e^x, b = \ln\frac{x}{30}$

10. $y = \frac{tg(\pi ax) + \cos(b)}{x^2 + ax + b}$, где $a = \frac{x^3}{x^4 + 1}, b = tg(x)$

11. $y = \frac{b + tg^2 ax}{(bx + a)^7}$, где $a = \frac{x^5}{x^6 + 1}, b = e^{2{,}5x}$

12. $y = \frac{a + \sqrt{x^3 + b}}{1 - e^{bx}}$, где $a = 1 - \sin(x), b = \frac{x^3 - 2x}{0{,}625x^2}$

13. $y = \cos(\frac{x}{a} + tg^2\frac{x}{bx + 3})$, где $a = e^{0{,}25x}, b = \frac{x}{x^3 - 1}$

14. $y = \frac{\sqrt{(x+a)^2 - 6}}{e^{bx}}$, где $a = x^{\cos(x)}, b = \frac{x^2}{2x + 1}$

15. $y = \frac{\ln(a)}{5(bx+1)^2}$, где $a = \frac{2}{(x+1)^2}, b = \frac{x^6}{x^5 + 1}$

16. $y = \frac{a + \sqrt{x^3 - bx}}{\sin\frac{x}{x - b}}$, где $a = \sqrt[4]{x}, b = tg\frac{1}{x^5}$

17. $y = \sin ax^3 - \sqrt{\cos(bx)}$, где $a = e^{2x}, b = \sin(x)$

18. $y = (a + \sqrt{x^3 - 1} + bx)^{ab}$, где $a = \cos(tg(x)), b = \frac{x^6}{x^5 - 1}$

19. $y = \frac{bx^2}{2\ln(ax^2 + \frac{1}{b})}$, где $a = \cos(5x), b = \sin(7x)$

20. $y = \frac{a \cdot tg(\frac{x}{b})}{\sin(x^2 - b)}$, где $a = e^{0{,}567x}, b = \cos(x)$

Лабораторная работа № 2

РАЗРАБОТКА И РЕАЛИЗАЦИЯ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ

Цель работы: изучение правил построения разветвляющихся алгоритмов и конструкций языка С++, реализующих эти алгоритмы.

Методические указания

Понятие разветвляющегося вычислительного процесса

Разветвляющимся называется такой вычислительный процесс, который в зависимости от выполнения условия идет по той либо иной заранее предусмотренной ветви.

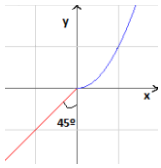
Для реализации разветвлений используется оператор ***if***. Общая форма записи оператора имеет такой вид:

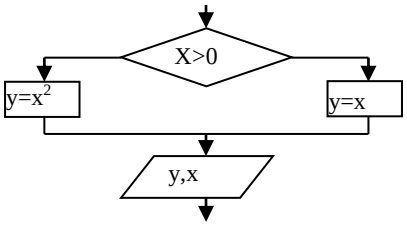
if (выражение)
< оператор 1 >
[else < оператор 2>;]

При выражении отличающемся от нуля, выполняется оператор 1 в противном случае – оператор 2.

По каждой из ветвей может стоять один оператор, если необходимо использовать более, их надо заключить в фигурные скобки.

Пример1. *Вычислить значение функции, представленной на графике.*

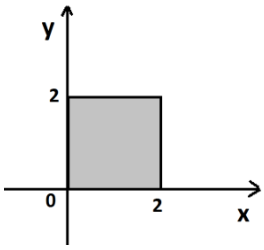




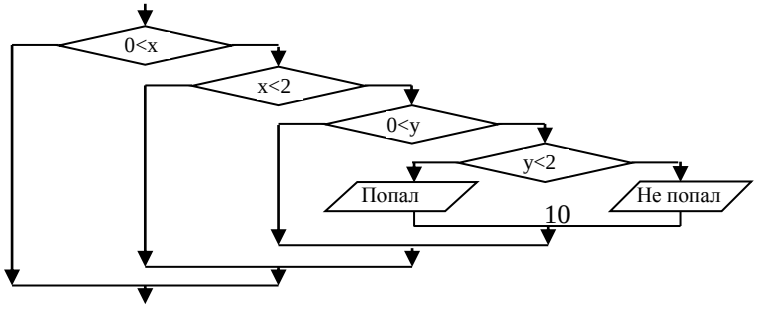

```
if (x>0)
    y=x*x;
      else y=x;
cout << “x= ”<<x<<“y=”<<y;
```

Пример2. *Определить попадание точки в закрашенную область.*

1-й способ решения задачи

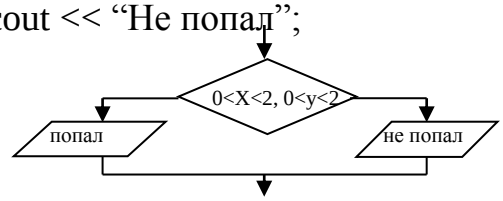



```
if (x>0)
 if (x<2)
  if (y>0)
   if (y<2)
    cout<< “Попал”;
    else cout << “Не попал”;
```



2-й способ решения задачи:

```
if (x>0 && x<2 && y>0 && y<2)
  cout<< “Попал”;
  else cout << “Не попал”;
```



В логических выражениях используются операции отношения и логические операции:

! - логическое НЕ

&& - логическое И

|| - логическое ИЛИ

В логическом выражении логические операции имеют приоритет ниже, чем операции отношения.

Приоритет логических операций, арифметических и операций отношения:

1) сначала выполняются операции !, ++, --
   ++ - операция инкремента C++ аналогично записи C = C+1.
   -- - операция декремента X++ аналогично записи X = X-1.
   ! – логическое НЕ.

Операции инкремента и декремента применяются в двух формах: префиксной и постфиксной. Префиксная форма подразумевает изменение переменной до ее использования например, ++X либо –C. Постфиксная форма применяется, когда в выражении переменная сначала используется для вычислений, а потом изменяется, например X++ либо C–;

затем

2) *(умножение), / (деление), % (остаток от деления);
3) + (сложение), - (вычитание);
4) >=, <=, <, > - операции сравнения;
5) ==, != - операции отношений (равенство и неравенство);
6) && - логическое И;

7) || - логическое И;

8) ? : - сравнение (тернарная операция - с тремя выражениями). Тернарная - условная операция используется в выражениях для получения одного из двух вариантов ответов в зависимости от условия.

Например, min = a>b ? b: a , здесь если выражение (a>b) истина, то результатом будет b, в противном случае a;

9) = - операция присваивания.

Так же могут использоваться операции продуктивного присваивания: +=, -=, *=.

Например,

S = S+f(x) S+=f(x),

S = S-f(x) S-=f(x),

S = S*f(x) S*=f(x).

В выражении допустимо использование многократного присвоения, которое выполняется справа налево, как и при инкрементации или декрементации, а в остальных случаях слева направо.

Например,

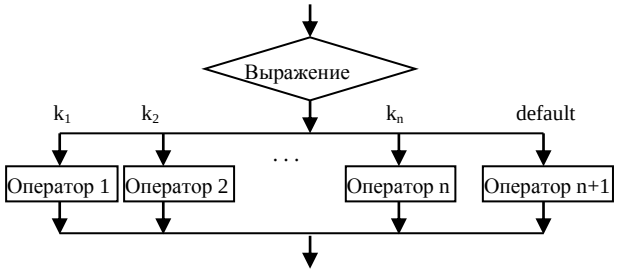
x=y=z=7.

При использовании выражения с операциями из одной группы их выполнение происходит по законам ассоциативности, а если из разных - то по приоритету.

Оператор выбора

Оператор выбора является очень удобной заменой множественного использования условного оператора. Оператор ***switch*** сравнивает значение селектора с несколькими константами k_1, k_2 и т.д., каждая из которых следует за ключевым словом ***case***. При совпадении выполняется оператор этой ветви, а затем оператор ***break,*** который осуществляет выход из оператора выбора, если совпадений нет, выполняется оператор, стоящий после слова ***default***.

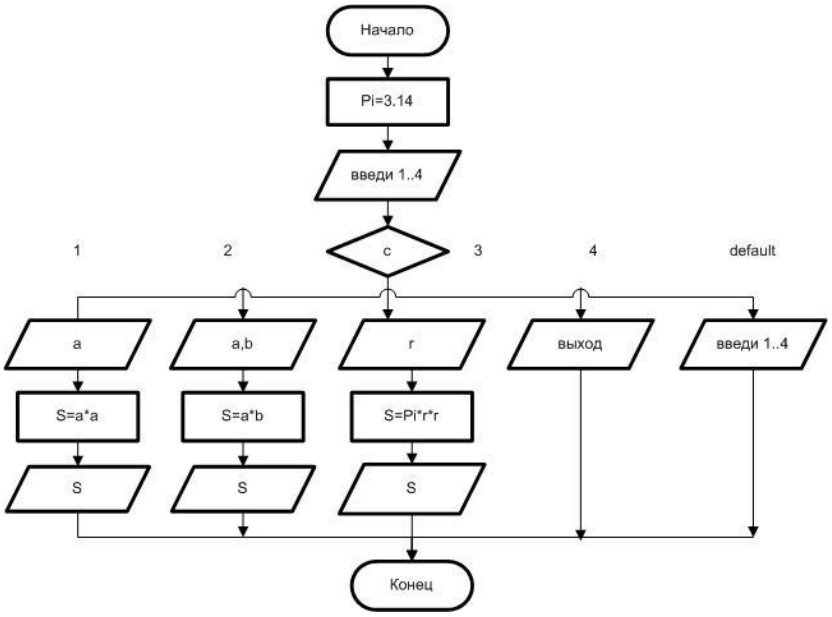
switch (селектор)
{case k_1: оператор 1; break;
case k_2: оператор 2; break;
…….
case k_n: оператор n;break;
default:операторы n+1;
}



Селектор – выражение или переменная порядкового типа данных (целого или символьного).

Пример. *Вычислите площадь заданной фигуры.*

```
#include <iostream.h>
 void main()
   {int c;
    float a,b,r,c;
    const float Pi=3.14;
    cout<< ″1.Площадь квадрата \n″;
    cout<< ″2.Площадь прямоугольника \n″;
    cout<< ″3.Площадь круга \n″;
    cout<< ″4.Выход \n″;
    cout<< ″Выберите пункты от 1 до 4 \n ″;
    cin>>c;
    switch (c)
     {case 1:{cout<< ″Введите сторону квадрата \n″;
              cin>>a;
              S=a*a;
              cout<< ″Площадь квадрата S=″<<S<<′\n′;
              } break;
      case 2:{cout<< ″Введите 1-ю сторону \n″;
              cin>>a;
               cout<< ″Введите 2-ю сторону \n″;
              cin>>b;
              S=a*b;
              cout<< ″Площадь прямоугольника
      S=″<<S<<′\n′;} break;
      case 3:{cout<< ″Введите радиус \n″;
              cin>>r;
              S=Pi*r*r;
              cout<< ″Площадь круга S=″<<S<<′\n′;
              } break;
      case 4:{cout<< ″Выход \n″;} break;
      default:cout<< ″ Выбирайте пункты 1-4 \n″;
              }
      }
```

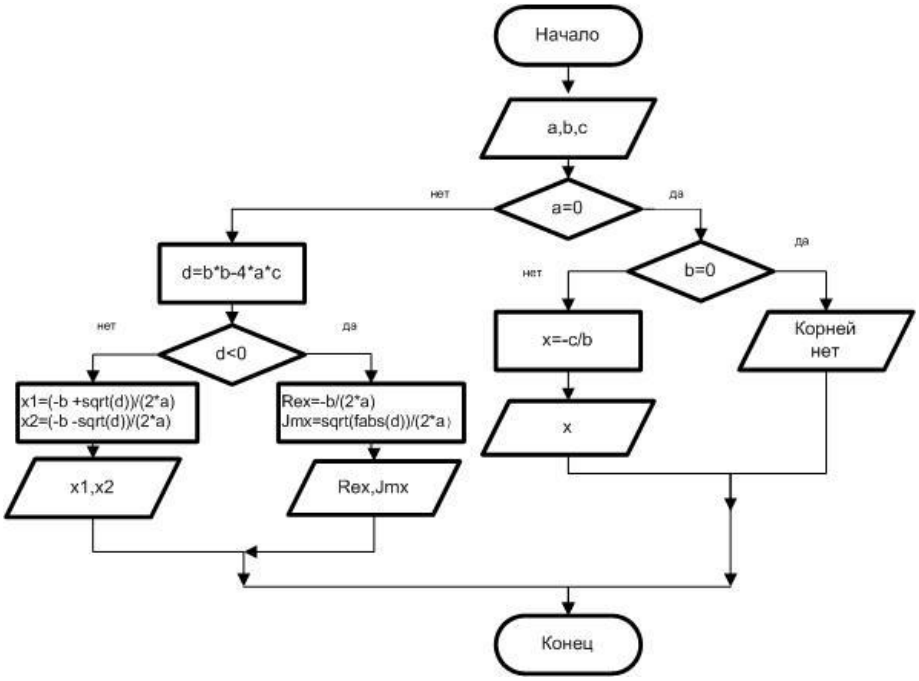


Пример. *Вычислить корни квадратного уравнения* $ax^2 + bx + c = 0$.

В зависимости от a, b, c возможны ситуации: уравнение имеет бесчисленное множество корней; один корень, если оно линейно; корни комплексные; корни действительные.

```
#include <stdio.h>
#include <math.h>
 void main()
 {float x, d, x1, x2, Rex, Jmx;
  int a, b, c;
  scanf ("%d %d %d", &a, &b, &c);
  if (a = = 0)
   {if (b = = 0 )
    printf (" \n корней нет");
    else
     {x = - c/(float) b;
      printf ("\n Один корень x = %f",x);
      }
   else
    {d = b*b - 4*a*c;
     if (d < 0)
        {Rex = - b/(float)(2*a);
       Jmx=sqrt(fabs(d))/(2*a);
       printf("\n комплексные Rex =%f, Jmx=%f", Rex, Jmx);
        }
       else
        {x1 = (-b +sqrt(d))/(2*a);
         x2 = (-b - sqrt(d))/(2*a);
         printf("\n действительные корни x1 = %f, x2 = %f", x1, x2);
```

```
    }
}
```



Контрольные вопросы

1. Какой процесс называется разветвляющимся?
2. Приведите пример разветвляющегося процесса, имеющего три ветви.
3. Какова форма условного оператора if?
4. Что такое истина выражения в языке C++?
5. Что такое отношение? Каковы операции отношения?
6. Приведите примеры сложного условного оператора.
7. Что такое составной оператор? Где он применяется?
8. С какой целью в выражениях используются логические операции?
9. Чем отличается условная операция от оператора if?
10. Какую структуру имеет оператор switch?
11. Каков принцип работы оператора switch?

Варианты заданий

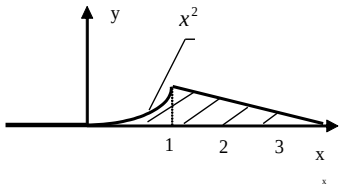
1. Даны значения A, B, C. Составить программу вычисления значения наименьшего отклонения каждого из трех чисел от их среднего арифметического значения и печати его значения и имени соответствующей переменной.

2. Даны два произвольных числа A, B. Если оба значения принадлежат отрезку [c,d], то вычислить R = sin(a)*sin(b), в противном случае печатать сообщение "Нет решения".

3. В зависимости от вещественных параметров A, B, C квадратного уравнения составить программу, печатающую одно из сообщений: "корней нет", "линейное

уравнение", "корни мнимые, корни действительные различные", "корни действительные равные".

4. Определить принадлежность точки B(x,y) к заштрихованной области



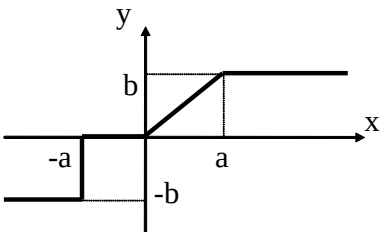
5. Даны три числа A, B, C. Найти остаток k от деления на 3 величины M и вычислить значения функции y.

$$M=\frac{a+b^2}{c};\quad \begin{cases} e^{M+C} \text{ ïðè } k=0, \\ \ln(a/b) \text{ ïðè } k=1, \\ \sqrt{(a+b)^2+c} \text{ ïðè } k=2. \end{cases}$$

6. Составить программу вычисления значения функции

$$f(x)=\begin{cases} \sin(x) \text{ ïðè } -2 \le x \le -0.1, \\ x^2+2x \text{ ïðè } 0 < x < a, \\ 1, \quad \text{â îñòàëüíûõ ñëó÷àÿx} \end{cases}$$

7. Составить программу вычисления значения функции, заданной графиком:



8. Даны три числа A, B, C. Составить программу вычисления экспоненты числа, значение которого ближе всего к значению функции.

$$y=\frac{\sin(b)+\cos(a)}{\ln(c+2)}.$$

9. В зависимости от величины остатка от деления на семь значения функции y = ln(x^2 + ab) напечатать сообщение об одном из дней недели.

10. Даны отрезки A, B, C, D. Составить программу определения возможности построения из них параллелограмма.

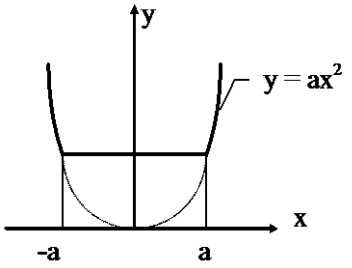
11. Даны три числа A, B, C. Определить, сколько среди них отрицательных чисел.

12. Расположить три заданных числа a, b, c в порядке убывания их значений.

13. Даны три числа A, B, C, которые являются сторонами треугольника. Определить вид треугольника: прямоугольный, остроугольный, тупоугольный.

14. Даны прямоугольное отверстие размером x на y и кирпич с гранями A, B, C. Составить программу для определения возможности прохождения кирпича через заданное отверстие.

15. Составить программу вычисления значения функции, заданной графиком:



16. Даны значения A, B, C. Если ни одно из значений не равно нулю, то вычислить $D = 1/a + 1/b + 1/c$, и если D > 0 и A > 0, вычислить $y = \sqrt{D}$, в противном случае y = 2. Если хотя бы одно из значений A, B, C равно нулю, то печатать сообщение ‘ Нет решения ’.

17. Даны значения A, B, C. Если A > B, B > C, то присвоить x = 0.2; $y = x^2 + 0.6x + \sin(x/2);$ если b < c и a < c, то x = 2; $y = x^3 + x^2 + x + 1;$ в остальных случаях x = 0, y = 0.

18. Составить программу нахождения меньшего из трех данных и печати его имени и значения.

19. Составить программу нахождения большего из трех данных и печати его имени и значения.

20. Даны значения x1, x2, x3. Составить программу присвоения переменным a1, a2, a3 значений по следующему правилу: если x1 > x2 > x3, то a1 = x1, a2 = x2, a3 = x3; если x1 ≤ x2 ≤ x3, то a1 = x3, a2 = x1, a3 = =x2, в остальных случаях печатать значения x1,x2,x3.

Лабораторная работа № 3

ПРОГРАММИРОВАНИЕ АЛГОРИТМОВ И ПРОГРАММ ЦИКЛИЧЕСКОЙ СТРУКТУРЫ ПРИ ПОМОЩИ ЦИКЛА FOR

Цель работы: изучение правил программирования циклических вычислительных процессов и вычисления конечных сумм и произведений.

Методические указания

Параметр цикла - это переменная, которая изменяется при выполнении цикла и определяет количество повторений цикла.

for (Выражение 1; Выражение 2; Выражение 3)
оператор; (или {блок операторов};)

Обычно ***Выражение 1*** используется для инициализации параметра только один раз перед выполнением цикла. ***Выражение 2*** используется для проверки

условия продолжения выполнения цикла. ***Выражение 3*** - для модификации параметра цикла после каждого выполнения тела цикла (итерации). Оператор или блок операторов выполняются до тех пор, пока истинно (не равно нулю) ***Выражение 2***.

Выражение 1, выражение 2, выражение 3 могут быть как порядкового, так и вещественного типа.

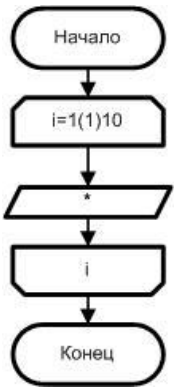
Тело цикла - это оператор или группа операторов (блок), выполняемых многократно. После каждого выполнения тела цикла осуществляется проверка истинности условия продолжения цикла. Операторы, реализующие циклы на языке С++, продолжают выполнение цикла до тех пор, пока условие продолжения цикла истинно (отлично от нуля).

При неправильно указанном ***Выражении 2*** может возникнуть ситуация, называемая бесконечным циклом или зацикливанием. В этом случае условие продолжения цикла постоянно остается истинным, тело цикла повторяется непрерывно и завершение цикла не происходит.

Если модификация параметра цикла проводится корректно, то при каждой итерации его значение приближается к выполнению условия выхода из цикла, затем цикл завершается.

Пример. *Вывести 10 раз на экран символ ‘*’*

```
# include < stdio.h >
  void main()
  { int i;
    for ( i = 1; i <= 10; i++)
    printf (″ %c ″,‘*’);
   }
```



Так же можно использовать операцию присваивания начальных значений или изменения нескольких переменных, используемых циклом:

```
for (p = 1, i = 0; i <= 10; p+ = 2, i++)
    { //блок операторов
    };
```

В этом случае следует иметь в виду, что запятая определяет порядок вычисления выражений, которые выполняются слева направо. Результат всего выражения будет равен результату последней операции. Для приведенного примера: значение ***i = 0*** в первом выражении и значение ***i++*** в третьем.

В теле цикла могут быть использованы управляющие операторы ***if***, которые в зависимости от выполнения каких-то условий требуют прервать выполнение цикла или перейти к следующей итерации, пропустив часть операторов тела

цикла. Для прерывания цикла используется оператор ***break*** , при его выполнении цикл заканчивается, хотя ***Выражение 2*** истинно. Для перехода к следующей итерации используется оператор ***continue***.

Пример. *Вывести на экран все четные числа из диапазона от 1 до 50. Задачу выполнить с использованием оператора continue.*

```
# include < stdio.h >
void main ()
  { int i;
    for (i = 1; i <= 50; i++)
      { if (i%2 ! = 0) continue;
        printf (″ %d \n ″, i);
      }
    }
```

В теле цикла, если остаток от деления числа *i* на 2 равен 0, печатается число *i,* в противном случае выполняется оператор continue и осуществляется переход к началу цикла, а функция *printf (″%d \n ″, i)* не выполняется.

Вычисление конечных сумм и произведений

Примерами использования цикла ***for*** могут служить алгоритмы вычисления конечной суммы или произведения. Вычисление конечной суммы сводится к вычислению суммы известного числа слагаемых. Конечная сумма представляет собой выражение типа

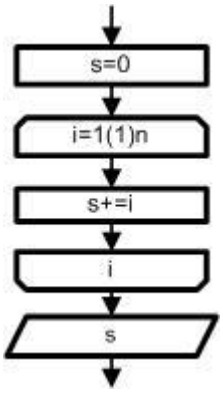
$$S = \sum_{i=1}^{n} a_i = a_1 + a_2 + a_3 + ... + a_n ,$$

где i - номер слагаемого;

a_i - слагаемое.

Подсчет значения суммы, состоящей в последовательном ее накоплении в переменной ***S***, можно представить в виде следующего фрагмента программы:

```
S = 0;
for ( i = 1; i <= n; i++).
S+= i;// вычисляется сумма всех
целых чисел от 1 до n.
```



Конечное произведение представляет собой выражение вида

$$P = \prod_{i=1}^{n} a_i = a_1 * a_2 * a_3 * ... * a_n ,$$

где i - номер сомножителя;

a_i - сомножитель.

Вычисляется конечное произведение аналогично подсчету суммы с тем отличием, что начальному значению ***P*** до цикла присваивается значение, равное 1, а в теле цикла знак сложения заменяется знаком умножения.

В том случае, когда общий член суммы содержит факториал или степенную функцию, он подсчитывается по рекуррентной формуле, умножением текущего слагаемого a_{n-1} на коэффициент k вычисляемый, как $\frac{a_n}{a_{n-1}}$.

Рассмотрим пример вычисления функции $(-1)^n \frac{x^n}{n!}$;

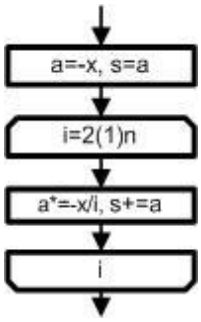
$$S = \sum_{i=1}^{n} (-1)^n \frac{x^n}{n!}.$$

Предварительно нужно найти начальный член ряда и рекуррентное соотношение:

$$a_1 = -x; \qquad a_n = (-1)^n \frac{x^n}{n!}; \qquad a_{n-1} = (-1)^{n-1} \frac{x^{n-1}}{(n-1)!};$$

$$k = \frac{a_n}{a_{n-1}} = \frac{(-1)^n \cdot x^n}{n!} \cdot \frac{(n-1)!}{(-1)^{n-1} \cdot x^{n-1}} = -\frac{x}{n}; \qquad a_n = k \cdot a_{n-1}.$$

Фрагмент программы данной задачи:




```
a=-x;
S = a;
for ( i = 2; i <= n; i++).
 {a*=-x/i
  S+ =a;
 }
```

Контрольные вопросы

1. Что такое циклический вычислительный процесс?
2. Какие элементы включают в себя циклы?
3. Какова общая форма оператора for?
4. Для чего используется оператор break? Приведите пример.
5. Как используется оператор continue?
6. Как можно упростить пример программы печати четных чисел?
7. Нарисуйте алгоритм программы для подсчета конечного произведения.

Варианты заданий

1. Вычислить сумму $S = \sum_{i=2}^{20} n/(n^2-2)$.
2. Вычислить сумму $S = \sum_{i=2}^{100} i/(10-2i)$.
3. Составить программу вычисления таблицы значений функции для 2< x <=5 с шагом 1, $f(x) = (\cos(x^2-4)\sqrt{1+x})/(x^2-2)$ Точки разрыва исключить.
4. Вычислить произведение для x = 2. $P = \prod_{k=0}^{15}(1+(x/(k+1)))$.
5. Вычислить сумму $S = \sum_{n=1}^{10}(n+1)/10^n$.
6. Вычислить произведение $P = \prod_{i=1}^{10}(i(2\cdot i+10))/(i+10)$.
7. Вычислить сумму $S = \sum_{i=2}^{100} i/(i-1)^2$.
8. Составить программу табулирования функции $Y = (x^2+\cos(x))/x$ с шагом 0.5 на интервале [0.5;3].
9. Составить программу табулирования функции $Y = \sin^2(x)/(x^2-4)$ с шагом 0.25 на интервале [-1.9;1.9].
10. Составить программу табулирования функции $Y = x^2(\tilde{n}os(x)+\sin(x))$ с шагом 0.5 в диапазоне $0 \le x \le 5$.
11. Вычислить произведение при x = 2. $P = \prod_{n=1}^{8}(x/n)^n$.
12. Вычислить произведение при a = 2, точку a = x исключить: $Y = \prod_{k=1}^{5}(a^k - x^k)$.
13. Вычислить $Z = k! + \sum_{i=1}^{5} k\cdot i$.
14. Вычислить сумму при x = 5. $P = \sum_{n=1}^{10}(x/n)^n$.
15. Вычислить сумму при x = 2. $S = \sum_{n=0}^{8}(1+x/(n+2))$.
16. Вычислить сумму при x = 0.5. $S = \sum_{k=0}^{12} 2kx/(2+k^2)$.
17. Вычислить $P = \sqrt{\sum_{n=0}^{50} 1/(2n+1)^2}$.
18. Составить программу табулирования функции $Y = 2x^3+3x^2+4x+10$ с шагом 0.5 в диапазоне $1 \le x \le 10$.
19. Вычислить сумму при x = 0.5 $S = \sum_{n=0}^{40} x^n \sin(\pi n/4)$.

20.Составить программу вычисления интеграла $I = \int_1^4 x^3/(x^4+1)dx$ по формуле прямоугольников $I = h\sum_{i=1}^{n-1} f(x_i v)$, где $h=(b-a)/n,\ x_i = x_0 + ih,\ f(x_i)$- подынтегральная функция, a, b - нижний и верхний пределы интегрирования, n= 50.

Лабораторная работа № 4

ПРОГРАММИРОВАНИЕ ИТЕРАЦИОННЫХ ЦИКЛИЧЕСКИХ ПРОЦЕССОВ

Цель работы: изучение правил программирования программ циклической структуры с предусловием и методов нахождения корней нелинейных и трансцендентных уравнений.

Методические указания

При решении многих задач количество повторений цикла заранее не определено. В таких случаях используют циклы с неизвестным числом повторений: предусловием и постусловием. В этих видах циклов проверка условия продолжения цикла осуществляется соответственно до и после выполнения тела цикла.

Цикл с предусловием

Синтаксис цикла с предусловием:

while (Выражение)
оператор [блок операторов];

Проверка условия продолжения цикла осуществляется до выполнения тела цикла. Пока результат ***Выражения*** не равен нулю, выполняется оператор (блок операторов). Перед входом в цикл ***while*** инициализируется переменная цикла. Если ***Выражение*** не равно нулю, то выполняется тело цикла, в противном случае оператор прекращает свою работу.

Для того чтобы ***Выражение*** обратилось в ноль и цикл завершился, в теле цикла должна изменяться переменная (переменные) цикла.

Для цикла ***while***, как и для других видов циклов, характерны типичные ошибки, приводящие к зацикливанию. Пример такой ошибки:

```
i = 0;
while ( i < 10);
i + = 2;
```

В этом случае следует помнить, что точка с запятой представляет собой ‘пустой’ оператор, который будет выполняться бесконечно, а оператор модификации переменной цикла i + = 2; не выполняется.

Другой распространенной ошибкой является использование в условном выражении вместо операции отношения = = операции присваивания =. Например:

```
while ( i = 2 )
< блок операторов >;
```

В этом случае условное выражение всегда дает результат, отличный от нуля, что также приводит к зацикливанию. Закончить выполнение цикла ***while*** можно при обращении в нуль ***Выражения*** либо в теле цикла использовать операторы ***break*** и ***return***.

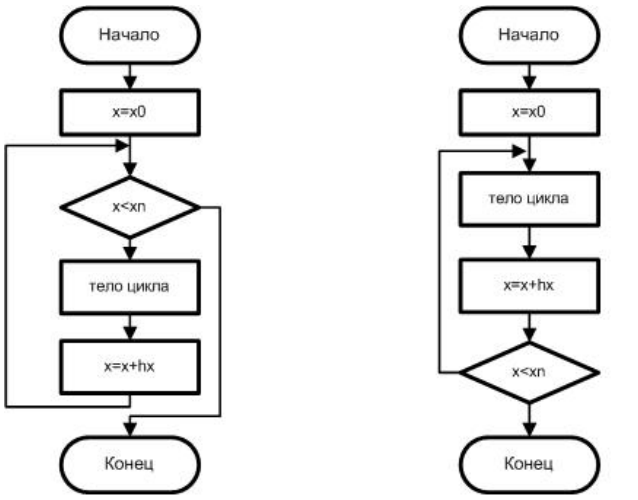
Пример. *Выполнить ввод данных только из указанного диапазона от 1 до 99*

```
int i = 0
  while ( i < 0 || i >= 100)
  { printf(″Введите число в диапазоне от 1 до 99″);
    scanf (″ %d ″,&i);
  }
```

Цикл с постусловием

```
do
    {
  тело цикла
    }
while (Выражение);
```

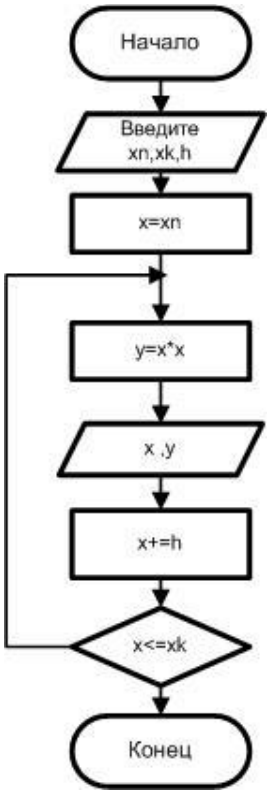
Пока выражение ***while*** отлично от 0, тело цикла повторяется. В противном случае происходит выход из цикла.



Пример. *Протабулировать функцию* $y=x^2$

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

```
{ float x,y,p,xn,xk,h;
  cout <<"Введите x начальное \n" ;
  cin>>xn;
  cout <<"Введите x конечное \n" ;
  cin>>xk;
  cout <<"Введите шаг\n" ;
  cin>>h;
  x=xn;
  do
  { y=x*x; // тело цикла
       printf("|%10.2f| %10.2f|\n",x,y);
       x+=h;// изменение переменной цикла
  }
  while(x<=xk); }
```



Нахождение корней уравнений

Нахождение корней уравнений *F(x) = 0* рассматриваемыми методами осуществляется в два этапа: отделение и уточнение корней.

На первом этапе производится нахождение диапазонов, в каждом из которых находится только один корень. Реализация этого этапа может быть приведена графически или табулированием функции. При табулировании в качестве границ диапазона следует выбирать значения аргумента, соответствующие соседним значениям функции, имеющим разные знаки. Второй этап - уточнение

корней, т.е. нахождение их значений с необходимой погрешностью, может производиться разными методами. В данной работе рассматриваются метод половинного деления и метод Ньютона.

Метод половинного деления

Основа метода состоит в последовательном делении отрезка *(a, b),* в котором находится только один корень, пополам: *x = (a + b)/2.* Затем находят значение функции *F(x)* и сравнивают его знак со знаком функции в левой границе интервала *F(a)*. Если знаки совпадают *(F(a)·F(x) > 0)*, корня в диапазоне от a до x нет, тогда левая граница сдвигается до значения *a = x*. Если знаки не совпадают *(F(a)·F(x) < 0)*, корень находится в диапазоне *(a, x),* а диапазон *(x, b)* не имеет корня. В этом случае сдвигается правая граница диапазона *b = x.* Процесс сужения диапазона продолжается до достижения погрешности

|a-b| <= eps. За корень принимают середину суженного отрезка *(a+b)/2*. Метод прост в реализации и имеет гарантированную сходимость. Блок - схема метода представлена на рис. 1.

Метод Ньютона

Метод Ньютона (касательных) основан на нахождении очередного приближения x_{n+1} как точки пересечения касательной в точке $F(x_n)$ и оси абсцисс. Итерационный процесс схождения к корню реализуется формулой

$$x_{n+1} = x_n - F(x_n) / F'(x_n),$$

где $F'(x_n)$ - первая производная функции *F(x)*, блок-схема алгоритма реализующего метод касательных показана на рис. 2.

В качестве x0 выбирают тот конец отрезка *a, b*, на котором знаки функции *F(x)* и ее второй производной *F''(x)* совпадают *(F(x)·F''(x) > 0).* В этом случае обеспечивается наиболее быстрая (квадратичная) сходимость. Блок схема метода представлена на рис. 2.

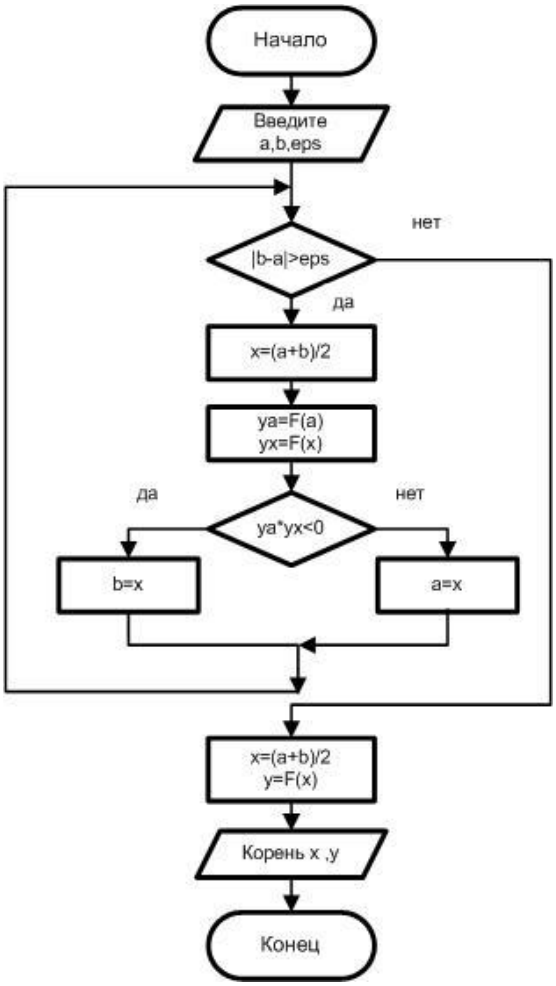

Начало
Введите
a,b,eps
нет
|b-a|>eps
да
x=(a+b)/2
ya=F(a)
yx=F(x)
да
нет
ya*yx<0
b=x
a=x
x=(a+b)/2
y=F(x)
Корень x ,y
Конец

Рис. 1

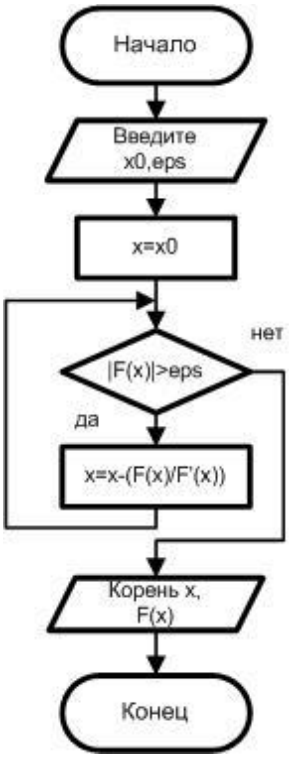

Начало
Введите
x0,eps
x=x0
нет
|F(x)|>eps
да
x=x-(F(x)/F'(x))
Корень x,
F(x)
Конец

Рис. 2

Контрольные вопросы

1. Нарисуйте блок-схему циклического вычислительного процесса с предусловием.
2. Какое минимальное число раз может выполняться цикл с предусловием и постусловием?
3. Объясните принцип метода половинного деления для уточнения корней.
4. Как подсчитать количество итераций при уточнении корней?
5. Какое минимальное значение может иметь погрешность уточнения корня для типа *float*?
6. Покажите графически, в чем состоит принцип уточнения корней методом Ньютона.
7. Что неправильно в приведенном цикле?

   while (i <=S // S > i)

   операторы;
8. Приведите синтаксическую структуру оператора *do-while*.
9. Чем отличается оператор *do-while* от ранее рассмотренных циклов?

Варианты заданий

1-6. Вычислить значение интеграла *I* с погрешностью *eps* < 10^{-3} по формуле трапеций:

$$I = h\left[(f(a) + f(b))/2 + \sum_{i=1}^{n-1} f(x_i) \right],$$

-где *f(x)* - подынтегральная функция, *a* и *b* - соответственно нижний и верхний пределы интегрирования, *h = (b - a)/n*, $x_i = a + ih$.

1. $I = \int_{10}^{15} \frac{\sin^2 x}{\cos^2 x + 1} dx.$
2. $I = \int_{3}^{15} \frac{\sin x}{\sin x + \cos x} dx.$
3. $I = \int_{0}^{1} \frac{dx}{x^3 \ln(x)}.$
4. $I = \int_{0}^{1} \frac{x^2 dx}{(x+1)^4}.$
5. $I = \int_{0}^{1} \frac{ctg(x)dx}{1 + ctg^2(x)}.$
6. $I = \int_{2}^{3} \frac{dx}{1 + e^{\frac{x}{2}}}.$

7-13. Уточнить корни уравнения в заданном диапазоне с погрешностью *eps* < 10^{-3} методом Ньютона:

$$x_{n=1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

7. $3\sin(x) + 0.35x - 3.8 = 0$ $2 < x < 3.$
8. $\arccos(x) - \sqrt{1 - 0.3x^3} = 0$ $0 < x < 1.$
9. $\sqrt{1 - 0.4x^2} - \arcsin(x) = 0$ $0 < x < 1.$
10. $x - 2 + \sin(1/x) = 0$ $1.2 < x < 2.$
11. $1 - x + \sin(x) - \ln(1 + x) = 0$ $0 < x < 15.$
12. $x^2 - \ln(1+x) - 3 = 0$ $2 < x < 3.$
13. $\ln(x) - x + 1.8 = 0$ $2 < x < 3.$

14-20. Найти корни уравнения с погрешностью $eps < 10^{-4}$ методом половинного деления:

14. $x + \sqrt{x} + \sqrt[3]{x} - 2.5 = 0.$
15. $3^x/1.6 - 2.3x - 3 = 0.$
16. $tg(x) - tg^3(x)/3 + tg^5(x)/5 = 0.$
17. $\cos(2/x) - 2\sin(1/x) + 1/x = 0.$
18. $\cos(x) - e^{-x^2/2} + x - 1 = 0.$
19. $\sqrt{1-x} - tg(x) = 0.$
20. $\sin(x^2) + \cos(x^2) - 10x = 0.$

Лабораторная работа № 5

ПРОГРАММИРОВАНИЕ СЛОЖНЫХ ЦИКЛИЧЕСКИХ ПРОЦЕССОВ

Цель работы: изучение приемов программирования сложных циклических процессов.

Методические указания

Любой циклический вычислительный процесс состоит из следующих частей:

- блока подготовки, в котором инициализируются начальные значения переменных, прежде всего параметра цикла;
- тела цикла, состоящего из операторов, которые выполняются до тех пор, пока выполняется условие продолжения цикла;
- блока или части, содержащей условие продолжения циклического процесса.

Если телом цикла является циклическая структура, то такие циклы называются вложенными или сложными циклическими процессами.

Циклическая структура, содержащая в себе другую циклическую структуру, называется внешним циклом. А цикл, находящийся внутри другого, называется внутренним.

Внутренний и внешний циклы могут быть любыми из трех рассмотренных выше типов: с параметром, с предусловием или с постусловием.

Правила организации как внешнего, так и внутреннего цикла такие же, как и для простых циклов. Однако нужно выполнять следующие условия:

- все операторы внутреннего цикла должны полностью лежать в теле внешнего цикла;
- при передаче управления от внешнего цикла к внутреннему необходимо восстановить начальное значение параметра внутреннего цикла.

Рассмотрим фрагмент программы вычисления значения функции двух переменных *z = f(x, y)*, если аргумент x меняется от *x0* до *xN* с шагом h_x, а аргумент *y* меняется от *y0* до *yM* с шагом h_y. Задачу двойного вложенного цикла можно представить несколькими способами.

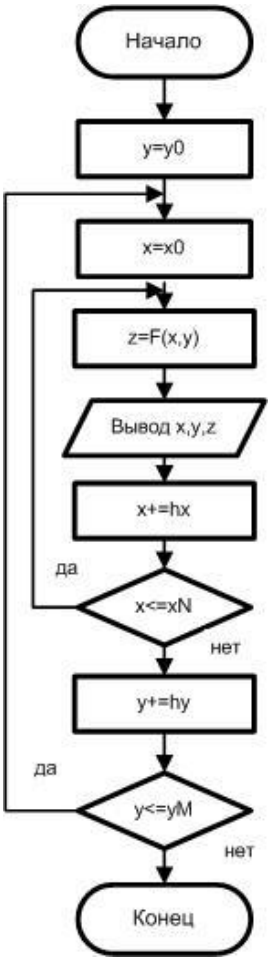
Программа с использованием оператора ***for*:**

```
for (y=y0; y <= yM; y+ = hy)
 { for (x=x0; x <= xN; x + = hx)
     { z = F(x, y);
       printf("x = %f y = %f z = %f \n", x, y, z);
     }
 }
```

Программа с использованием оператора ***do-while*** .

```
y = y0;
do
 { x = x0;
   do
      { z = F(x, y);
        printf("x = %f y = %f z = %f \n", x, y, z);
        x+= hx;
      }
   while (x <= xN);
   y+= hx;
 }
while (y <= yM);
```

Блок-схема вычислительного процесса с использованием цикла *do while* представлена рисунке.



Вычислить значение интеграла $\int_0^a e^{-2x}dx$ *методом трапеций для различных значений верхнего предела a, если он меняется от* $a_0=1$ *до* $a_n=10$ *с шагом* $h_a=0.1$. *Число точек разбиения интервала задано и равно* $n = 46$. *Для каждого вычисленного значения оценить абсолютную погрешность как разность между приближенным значением интеграла и точным, которая подсчитывается по первообразной интеграла.*

Первообразная данного интеграла имеет вид $\int_0^x e^{-2x}dx = \frac{1}{2}(1-e^{-2x})$. Решение.

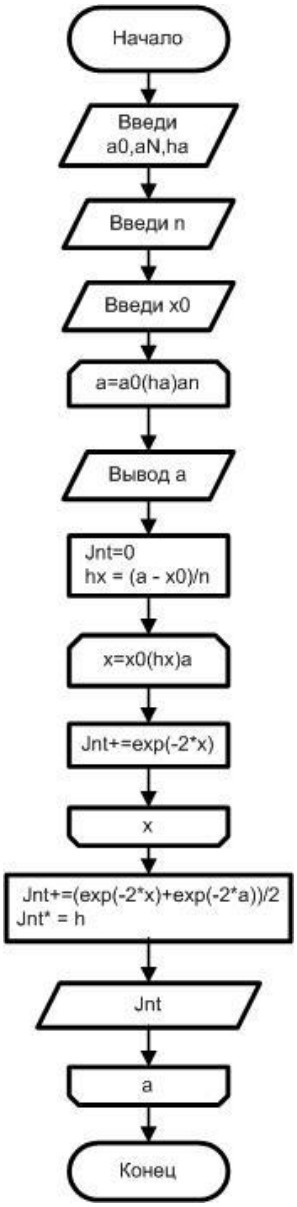
Формула трапеций для вычисления интеграла $J = \int_a^b y(x)dx = \int_a^b f(x)dx$ имеет вид $J = \frac{b-a}{n}\left(\frac{y_0+y_n}{2} + \sum_{k=1}^{n-1} y_k\right)$ или, в другом виде, $J = \frac{b-a}{2n}\left(\sum_{k=0}^{n-1} y_i + y_{i+1}\right)$, где $y_k = y_i$ - значение подынтегральной функции в точке $x_k = x_{k-1} + h$, где шаг $h = \frac{b-a}{n}$. В программе можно использовать как первую, так и вторую функции.

Программа представляет сложный циклический процесс, в котором внешний цикл обеспечивает изменение верхнего предела интегрирования *a* с шагом h_a в

данном диапазоне, а внутренний цикл по переменной *x* обеспечивает вычисление интеграла при фиксированном значении *a* методом трапеций.

При выполнении варианта задания, в котором нужно найти корни уравнения *f(x) = 0* при различных параметрах, прежде всего, нужно представить уравнение в виде *x = f(x), где |f'(x)| < 1*. Это обеспечивает сходимость итерационного процесса.

```
   /* пример вычисления интеграла */
# include <stdio.h>
void main ( )
{float a, a0,aN, ha;
 float x, x0, hx, Jnt;
 int n;
 printf(″a0, aN, ha \n”);
 scanf(″%f%f%f″,&a0,&aN,&ha);
 printf(″введем n “);
 scanf (″%d″, &n);
 printf (″введем x0“);
 scanf (″%f ″, &x0);
 for (a = a0; a <= aN; a + = ha)
 {/*текущий предел интегрирования */
  printf (″a = %f \n″, a);
  /* вычисляем интеграл при a */
  hx = (a - x0)/n;
  for (x=x0, Jnt = 0; x <= a; x+ = hx;)
   {Jnt += exp(-2*x);
    }
  Jnt += (exp(-2*x-) + exp(-2*a))/2;
  Jnt *= h;
  /* печатаем a и значение Jnt */
  printf (″a = %f Jnt = %f″, a , Jnt);
  }
}.
```


Начало
Введи
a0,aN,ha
Введи n
Введи x0
a=a0(ha)an
Вывод a
Jnt=0
hx = (a - x0)/n
x=x0(hx)a
Jnt+=exp(-2*x)
x
Jnt+=(exp(-2*x)+exp(-2*a))/2
Jnt* = h
Jnt
a
Конец

Контрольные вопросы

1. Дайте определение сложного циклического процесса.
2. Нарисуйте блок-схему вложенных циклов для циклических процессов со счетчиком, с постусловием и предусловием.
3. Какие условия нужно выполнять при написании программ, содержащих вложенные циклы?
4. Что изменится, если поменять местами внутренний и внешний циклы?

Варианты заданий

В вариантах заданий 1-6 найти корень уравнения методом половинного деления.

N	Уравнение	Диапазон изменения параметра A	h_a	Начальное приближение
1	$3\sin\sqrt{x} + Ax - 3.8 = 0$	*0.2<A<0.4*	*0.01*	*2.2985*
2	$x - \frac{1}{3+\sin Ax} = 0$	*2≤A≤2.5*	*0.1*	*0.2561*
3	$x - 2 + \sin\frac{A}{x} = 0$	*0.5≤A≤1*	*0.1*	*1.3077*
4	$x + \cos(x^A + 2) = 0$	*0.25≤A≤0.55*	*0.05*	*0.9800*
5	$x^2 - \ln(A + x) - 3 = 0$	*1≤A≤2*	*0.1*	*2.0297*
6	*$\sin(x^2) + \cos(x^2) - A\,x = 0$.*	*5≤A≤10*	*1*	*0.01*

В заданиях с номером 7-15 вычислить значения интеграла $\int_a^b f(x)$ методом трапеций, если один из пределов интегрирования a или b меняется в заданном диапазоне с некоторым шагом. Для каждого значения параметра вычислить абсолютную погрешность как разность между приближенным и точным значениями интеграла, вычисленным по первообразной этого интеграла. Исходные данные для выполнения задания приведены в таблице, где *f(x)* – функция, *a* и *b* - пределы интегрирования, *n* – количество точек разбиения, *f′(x)* – первообразная.

N	f(x)	Пределы	n	f′(x)

7	$e^x \cos^2 x$	$a = 0$ $1/8\pi \le b \le \pi$ $h_b=1/8\pi$	60	$e^x\left(1 + \frac{2\sin 2x + \cos 2x}{5}\right.$
8	$(x\,lnx)^2$	$b = 2.71$ $0.5 \le a \le 1.5$ $h_a=0.1$	52	$\frac{x^3(9ln^2x - 6lnx + 2}{27} + \frac{2}{27}$
9	$\frac{1}{(x+1)\sqrt{x^2+}}$	$a = 0$ $1/4 \le b \le 3/4$ $h_b=1/8$	40	$\frac{1}{\sqrt{2}}\left(ln\frac{1+\sqrt{2}}{2}\right.$ $- ln\frac{1 - x + \sqrt{2(x}}{2(x+1}$
10	$\frac{x}{x^4 + 3x^2 + 2}$	$a = 0$ $1.5 \le b \le 2.5$ $h_b=0.1$	36	$\frac{1}{2} ln\frac{x^2+1}{x^2+2} - \frac{1}{2} ln\frac{2}{3}$
11	$\frac{e^x(1+sin\,x)}{1+cosx}$	$a = 0$ $1 \le b \le 1.5$ $h_b=0.1$	150	$e^x tg\frac{x}{2}$
12	$\frac{1}{(3sinx + 2co}$	$a = 0$ $0.5 \le b \le 1$ $h_b=0.1$	78	$\frac{3}{26} + \frac{3cosx - 2sin}{13(2cosx + 3si}$
13	$\frac{1}{\sqrt{9+x^2}}$	$a = 0$ $0.5 \le b \le 2$ $h_b=0.1$	150	$\ln(x + \sqrt{x^2+9})$
14	$\frac{e^{3x}+1}{e^x+1}$	$a = 0$ $1.5 \le b \le 2$ $h_b=0.1$	50	$\frac{e^{2x}}{2} - e^x + x$
15	$\frac{\sqrt{x^2-0.16}}{x}$	$a = 0$ $1.5 \le b \le 2.5$ $h_b=0.1$	150	$\sqrt{x^2-0.16} - 0.4arccos\frac{0.4}{x}$

16. Составить программу вычисления значений функции *f(a, b)*, если *a* меняется от a_0 до a_n с шагом h_a $(a = a_0(h_a)a_n)$, а $(b = b_0(h_b)b_m)$. Все данные вводятся с клавиатуры.

Функция

$f(a,b) = \frac{a+b}{a^2+b^2} \sum_{n=1}^{10} \frac{(xa+b)^n}{n!}$, где $x = \begin{cases} 2, & \text{åñëè } a < b, \\ 1, & \text{åñëè } a \ge b. \end{cases}$

17. Составить программу вычисления значений функции *f(a, b)*, если *a* меняется от a_0 до a_n с шагом h_a $(a = a_0(h_a)a_n)$, а $(b = b_0(h_b)b_m)$. Все данные вводятся с клавиатуры. Вычислить таблицу значений функции

$$f(a,b) = \frac{a+b^2}{ab} \sum_{n=1}^{20} \frac{(x\cos ax + b)^n}{n!} \text{ при } x = \begin{cases} \frac{\pi}{2}, & \text{åñëè } a < b, \\ \frac{3}{4}\pi, & \text{åñëè } a \ge b. \end{cases}$$

18.Вычислить таблицу значений функции $f(a,b) = \sum_{n=1}^{20} a\cos bxn + \frac{b}{n}\sin anx$, где h_a $(a = a_0(h_a)a_n)$ и $(b = b_0(h_b)b_n)$ и $x = \begin{cases} \frac{\pi}{6}, & \text{åñëè } a < b, \\ \frac{\pi}{2}, & \text{åñëè } a \ge b. \end{cases}$

19.Вычислить таблицу значений функции $f(x,y,a) = \frac{a}{a^2+b^2} \sum_{n=1}^{20} a\cos(bxn + 0.1y^2)$, где y = 0.1 (0.1) 1.5,

b = 5 (0.2) 6, b – вводится, $x = \begin{cases} \cos(b^2+1), & \text{åñëè } y \le 0.7, \\ \sin(b^2+1), & \text{åñëè } y \ge 0.7. \end{cases}$

20.Вычислить таблицу значений функции

$f(t,w) = \sum_{n=1}^{20} (t^2+1)\cos wn + \frac{t}{t^2+1}\sin w(n+1)$, где $w = \quad w_0(h_w)w_n$ и $t = \quad t_0(h_t)t_m$. Все параметры выбираются произвольно и вводятся с клавиатуры.

Лабораторная работа № 6

МАССИВЫ

Цель работы: изучение особенностей работы с массивами.

Методические указания

Массив – это совокупность данных одного и того же типа, расположенных в памяти ЭВМ последовательно и обозначаемых одним именем.

Массив характеризуется следующими понятиями: элемент массива, имя массива, размер массива, тип элемента, размерность массива.

Элемент массива – значение элемента, хранящееся в определенной ячейке памяти, расположенной в пределах массива.

Адрес массива – адрес начального элемента массива.

Имя массива – идентификатор, используемый для обращения к элементам массива, который компилятор связывает с адресом начального элемента массива.

Размер массива – количество элементов в массиве, размер в байтах равен размеру элемента массива, умноженного на количество элементов.

Размерность массива – это способ его представления. Если он представляется в виде одной строки, то это одномерный, если таблицей или матрицей - двумерный массив, а пачкой таблиц - трехмерный и т.д.

Для определения размера элемента массива используется операция **sizeof (тип).**

Размер элемента – количество байт, занимаемых одним элементом массива, определяется типом элементов при описании.

Адрес	n	n+k	n+2k		n+k(q-1)
Значение	a[0]	a[1]	a[2]	…	a[q-1]
Индекс	0	1	2		q-1

На рисунке рассмотрен массив *a* из *q* элементов с индексами от *0* до *q-1*, *к*аждый элемент, которого в памяти занимает *k* байт. Адрес *i* элемента *n+k*i.*

Для обращения к элементу массива используют индекс [ ], например *a*[5] - обращение к 6-му элементу массива *a*.

Для объявления массива необходимо указать:

тип элементов имя массива [количество элементов]

Например, int a[10].

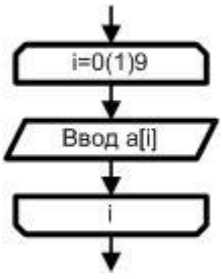
Можно сразу инициализировать элементы массива. Например,

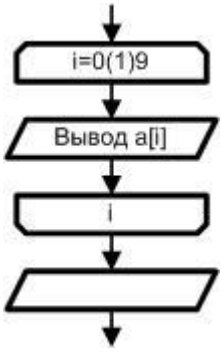
int a[10]={1,2,3,4,5,6,7,8,9,0} // массив из 10 целых чисел.

Ввод и вывод элементов массива можно выполнять с клавиатуры:

```
// ввод элементов массива
for (i=0; i <= 9; i++)
 scanf (″%d″, &a[i]);
// &a[i] – адрес i-элемента
```

```
// вывод элементов массива
for (i=0; i <= 9; i++)
 printf (″%d″, a[i]);
 printf(″\n″)
```



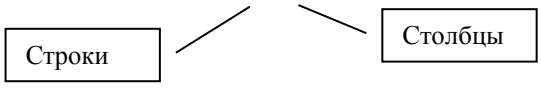


При объявлении двумерного массива необходимо указать:

тип элементов имя массива [количество строк][количество столбцов]

Например,

int a[2][3];

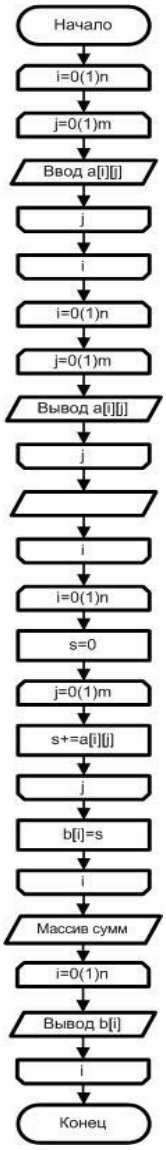


Адрес	n	n+4	n+8	n+12	n+16	n+20
Значение	a[0][0]	a[0][1]	a[0][2]	a[1][0]	a[1][1]	a[1][2]
Индекс	00	01	02	10	11	12

Инициализировать двумерный массив $\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\end{bmatrix}$ можно:

int b[2][3]={1,2,3,4,5,6};
Или int b[2][3]={{1,2,3},{4,5,6}}.

Пример. *Программы для работы с одномерный массив из сумм размером 5 строк и 5 столбцов.*



массивами. Составить элементов строк матрицы a

```
# include <stdio.h>
# define n 5
# define m 5
void main ( )
{int a[n][m];
 int b[n];
 int s,i,j;
// ввод исходной матрицы
 for (i = 0; i < n; i ++)
  for (j = 0; j < m; j ++)
   scanf (″%d ″, &a[i][j]);
// вывод матрицы
for (i = 0; i < n; i ++)
 {
  for (j = 0; j < m; j ++)
   printf (″%5d ″, a[i][j]);
 printf (″\n”);
 }
// формирование одномерного массива
for (i = 0; i < n; i ++)
 {
  for (s=0,j = 0; j < m; j ++)
   s+=a[i][j];
  b[i]=s;
 }
// вывод массива сумм строк
printf (″массив сумм″);
for (i = 0; i < n; i ++)
  printf (″%5d ″, b[i]);
}.
```

Контрольные вопросы

1. Дайте определение массива.
2. Каковы основные характеристики массива?
3. Чем характеризуется элемент массива?
4. Приведите пример описания одномерного и двумерного массивов.
5. Какой будет размер массива *double a[10]* в байтах?
6. Какой будет размер массива *float a[10][10]* в байтах?
7. Напишите фрагмент для вывода матрицы *a[5][5]* на экран в виде таблицы.

Варианты заданий

1. В произвольно заданных матрицах *a[5][4],b[5][4]* определить максимальные элементы и поменять их значения местами. В матрице *a* все отрицательные элементы заменить максимальным значением.

2. Сформировать одномерный массив, состоящий из максимальных значений положительных элементов соответствующих строк произвольно заданной матрицы *b[5][6]*.

3. Определить максимальный элемент в квадратной матрице размером *a[5][5]*, лежащий выше главной диагонали, и минимальный элемент среди элементов, лежащих ниже главной диагонали. Поменять найденные значения местами.

4. Определить максимальный элемент в произвольно заданной матрице *a[n][n]* и обнулить все элементы строки и столбца, на пересечении которых расположено найденное значение.

5. Сформировать одномерный массив из максимальных элементов столбцов произвольно заданной матрицы размером *a[m][n]*. В полученном массиве найти минимальный элемент.

6. Сформировать произвольный двумерный массив целочисленных значений размером *a[6][6]*. Определить число повторений каждого из значений первой строки.

7. В произвольно заданном двумерном массиве *a[4][5]* определить три элемента с наибольшими значениями.

8. В произвольно заданном двумерном массиве *a[n][n]* поменять местами строки, содержащие минимальный и максимальный элемент. Если минимальный и максимальный элемент принадлежат одной строке, то поменять местами соответствующие столбцы.

9. В произвольно заданной матрице размером *a[4][6]* определить строку с максимальной суммой элементов и столбец с минимальной суммой.

10. Из произвольно заданной матрицы *a[5][5]* сформировать одномерный массив из положительных элементов исходной матрицы в порядке следования строк.

11. В двух произвольно заданных двумерных массивах *a[n][n],b[n][n]* поменять местами строки, содержащие максимальные элементы. Вывести на экран исходные и измененные матрицы.

12. Сформировать два произвольных двумерных массива размером *a[5][4]*, *b[5][4]*. Поменять местами столбцы исходных матриц, содержащие минимальные элементы.

13. Определить минимальный элемент в произвольно заданной матрице размером *a[4][6]*. Заменить на это минимальное значение все элементы строки и столбца, которым принадлежит найденное значение.

14. Найти минимальный элемент на главной диагонали и максимальный элемент на побочной диагонали в квадратной матрице *a[5][5]*. Найденные значения поменять местами. Если они одинаковые, то это значение присвоить всем элементам главной и побочной диагоналей.

15. Определить и поменять местами максимальное и минимальное значения среди элементов, расположенных выше главной и ниже вспомогательной диагоналей в произвольно заданной квадратной матрице размером *a[6][6]*.

16. В произвольно заданной матрице *a[5][4]*определить минимальный элемент и обнулить значение элементов, расположенных ниже и правее найденного элемента.

17. В трех произвольно заданных положительных матрицах размером *a[3][3]*, *b[3][3]*, *c[3][3]* определить максимальные элементы. Считая найденные значения длинами отрезков, определить возможность построения из них треугольника.

18. Сформировать одномерный массив, элементами которого являются средние значения строк произвольно заданной матрицы размером *a[8][3]*. Упорядочить значения одномерного массива по возрастанию.

19. Сформировать одномерный массив из положительных элементов произвольно заданной матрицы *a[n][n]* и упорядочить отобранные значения по убыванию.

20. Произвольно заданы две матрицы размером *a[5][3]*,*b[5][3]*. Сформировать новую матрицу, каждый элемент которой является наибольшим значением из соответствующих элементов исходных матриц. В полученной матрице определить сумму положительных элементов.

Лабораторная работа № 7

ЗНАКОМСТВО С ПОЛЬЗОВАТЕЛЬСКИМИ ФУНКЦИЯМИ

Цель работы: изучение особенностей построения и использования простейших функций.

Методические указания

Программа на языке С++ представляет собой совокупность функций. Обязательно присутствует хотя бы одна функция *main()* или *WinMain()*, которая является точкой входа в программу. С нее обычно начинается выполнение программы. Пользователь может сам определять функции для улучшения структуры программы в случае использования алгоритмов с часто

повторяющимися фрагментами, а также применять более тысячи библиотечных функций.

Функция – это совокупность объявлений и операторов, предназначенных для решения конкретной задачи. Результатом работы функции может быть одно скалярное значение.

Допускается использование функции, не имеющей аргументов и не возвращающей значения. Например, изменение значений некоторых переменных или вывод на экран некоторых сообщений.

При определении функции необходимо указать ее заголовок и блок выполняемых функцией операторов.

Функции имеют следующее определение:

тип имя функции (список формальных параметров)
{ тело функции
return <выражение> };

Тип – тип значения, возвращаемого функцией, он может быть любым. Если тип возвращаемого значения не задан, то он по умолчанию определяется как *int*. Если значение не возвращается, то тип *void*.

Имя функции – идентификатор.

Список формальных параметров – это последовательность объявлений (переменных), разделенных запятыми с указанием их типов. Определенное количество формальных параметров задается на этапе написания функции соответственно решаемой задаче.

Тело функции – это описания локальных переменных и код реализации функции, оператор или набор операторов, определяющих вычислительный процесс.

Локальные переменные – это переменные, используемые только в теле функции. Они не доступны другим функциям программы, в том числе и *main*.

Результат работы функции - это значение, получаемое после подсчета выражения, стоящего после слова ***return***. Тип выражения должен совпадать с типом функции, заявленным при описании. Если ***return*** отсутствует, то возвращаемое значение неопределенно и тогда тип возвращаемой функции необходимо указать ***void***.

Например, определить значение функции $y=x^2$.

Результат вычислений можно реализовать тремя видами функций:

```
//функция вычисляет значение x^2
float square1(float x)
{ float tmp=x*x;
//tmp локальная переменная, используется только в теле функции
  return tmp;
}

//функция вычисляет значение x^2 без дополнительных объявлений
float square2(float x)
```

```
{
  return x*x;
}

//функция вычисляет значение x^2  и выводит результат на печать
void print_square3(float x)
{
  printf(“y= %f”,x*x);
}
```

В случае когда функция не использует параметров, то вместо них можно указать *void*.

Пример описания функции без аргументов:

```
void print_Hello()
{
  printf(“Hello”);
}
// либо
void print_Hello(void)
{
  printf(“Hello”);
}
```

При вызове функции необходимо указать ее имя и список фактических параметров:

имя функции (список фактических параметров);

```
printf(“Функция=%f”, square1(1.5));//на экране  функция=2.25
print_square3(1.5); // на экране y=2.25
```

Фактические параметры – это переменные, константы, выражения, указанные при вызове функции в программе и соответствующие по количеству, типу и последовательности (порядку следования) формальным, т.е. они заменяют в основной программе формальные.

Функция может вызываться в операторе присваивания, в качестве параметров других функций и самостоятельно.

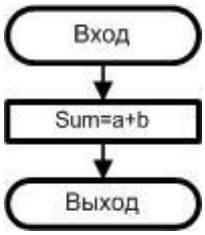
Функции в С++ равноправны, любая функция может быть вызвана из любой другой. Допускается рекурсивный вызов функции, когда функция вызывает сама себя.

Функция *main()* представляет собой точку входа в программу, она запускается первой.

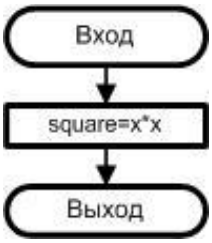
Прототип функции или описание - это заголовок функции без тела, заканчивающийся ***;***. Его применяют в том случае, когда вызов функции предшествует ее определению.

Пример. *Составить функции возведения в квадрат числа и нахождения суммы двух слагаемых. Используя их вычислить выражения:* $a+b$, x^2, $(a+b)^2$.

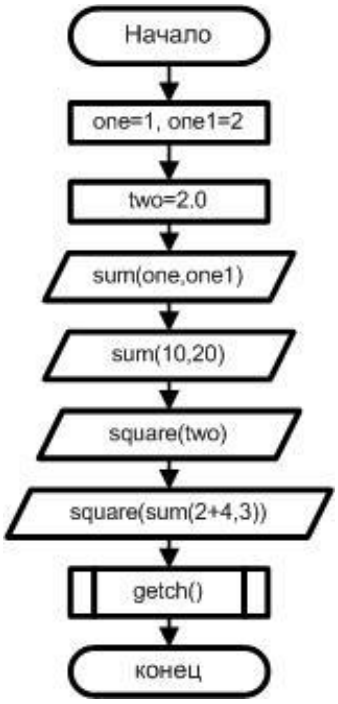
```
# include <stdio.h>
# include <conio.h>
//описание функций
//функция суммирования
//формальные параметры a и b
```




```
float sum(float a, float b)
{
  return a+b;
}
//функция возведения в квадрат
float square(float x)
{
  return x*x;
}
```




```
void main ( ) //основная программа
{ //определение фактических параметров
 float one=1,one1=2;
 float two=2.0;
 //вызываем функцию
 //и передаем фактические параметры
 printf("%f\n",sum(one,one1));
 // передаем числовые константы
 printf("%f\n",sum(10,20));
 //вызов функции x2
 printf("%f",square(two));
 //в качестве аргумента
 //может быть другая функция
 printf("%f",square(sum(2+4,3)));
// используем стандартную функцию,
// которая считывает символ с консоли,
// но не выводит его на экран
 getch();//остановка программы, для продол-
        // жения нажмите любую клавишу.
```



}

Передача параметров функции может выполняться по значению. В этом случае фактическим параметром может быть константа, переменная, выражение. Число соответствующее этому параметру, копируется в локальную память функции, это означает невозможность изменения этих значений внутри функции. Рассматриваемые параметры являются входными, а это значит, что изменить их в процессе вычисления невозможно (см. вышерассмотренный пример).

```
float sum(float a, float b)// описание
{
  return a+b;
}
printf(“%f\n”,sum(10,20));//вызов функции в операторе
```

Если внутри функции нужно изменить больше чем одно значение, то в качестве параметра передаются адреса (указатели) переменных или ссылки на них:

```
void fun_2(float *a, float *b)// описание
{
  *a+=2;
  *b+=2;
  return ;
}
float y=5,z=7;
fun_2(&y,&z);
printf(“y=% z=%f\n”,y,z);//вывод на экран y=7 z=9
```

Такое использование переменных называется работой с указателем. Указатели - это переменные, позволяющие хранить адрес. Компьютер выделяет память в объеме, соответствующему адресу. Объявление указателя выполняется следующим образом:

тип * имя_указателя;

Способ передачи параметров по ссылке, при этом в списке формальных параметров они указываются со знаком **&**:

```
void fun_3(float &a, float &b)// описание
{
  a+=2;
  b+=2;
  return ;
}
float y=5,z=7;
fun_3(y,z);
printf("y=% z=%f\n",y,z);//вывод на экран y=7 z=9
```

В языке С++ в качестве средства повышения быстродействия применяются встроенные функции. Для их объявления используется спецификатор ***inline***, при этом в теле функции не должно быть операторов цикла, условных операторов и оператора ***goto***, а их общее число не более 5 операторов.

```
inline float sum(float a, float b)// описание
{
  return a+b;
}
```

Пример. *Программы для работы с массивами. Составить одномерный массив из сумм элементов строк матрицы a размером 5 строк и 5 столбцов.*

При составлении программы количество строк и столбцов исходной матрицы определим константами *n*, *m*. Составим функции для ввода и вывода исходной матрицы, а также функцию, которая будет по строкам формировать массив из сумм элементов строк исходной матрицы. Результат работы данной программы показан на рисунке 3.

```
#include<stdio.h>
const int n=5;
const int m=5;
//функция для ввода матрицы
void input(int a[n][m],int n,int m)
{
 for (int i=0;i<n;i++)
  for (int j=0;j<m;j++)
   scanf("%d",&a[i][j]);
}
//функция для вывода матрицы на экран
```

```
void out(int a[n][m],int n,int m)
{
 for (int i=0;i<n;i++)
 {
  for (int j=0;j<m;j++)
   printf("%5d",a[i][j]);
  printf("\n");
 }
}
//функция формирования одномерного массива
void schet(int a[n][m],int n,int m,int b[n])
{
 for (int i=0;i<n;i++)
 {
  for (int s=0,j=0;j<m;j++)
    s+=a[i][j];
  b[i]=s;
 }
}


void main()//основная программа
{
  int a[n][m],b[n];
  int i,j,s;
  input(a,n,m);
  printf("исходная матрица \n");
  out(a,n,m);
  schet(a,n,m,b);
  printf("массив сумм строк \n ");
  for (i=0;i<n;i++)
   printf("%5d",b[i]);
  printf("\n");
}
```

Рис. 3

Данную задачу можно выполнить другим способом: размерность матрицы ввести с клавиатуры, в программе представить каждую строку матрицы как отдельный массив, а функция *summ* будет определять сумму элементов

одномерного массива. Результат работы данного варианта программы представлен на рисунке 4.

```
#include<stdio.h>
#include<conio.h>
const int n=5;
const int m=5;
void input(int a[][m],int n,int m)
{
 for (int i=0;i<n;i++)
  for (int j=0;j<m;j++)
   scanf("%d",&a[i][j]);
}
void out(int a[][m],int n,int m)
{
 for (int i=0;i<n;i++)
 {
  for (int j=0;j<m;j++)
   printf("%5d",a[i][j]);
  printf("\n");
 }
}
int summ(int c[],int m)
{
 int i; int s=0;
 for (i=0;i<m;i++)
   s+=c[i];
 return s;
}
void main()
{
 int aa[n][m],b[m];
 int i,j,s,n1,m1;
 printf("введи n1,m1 меньше 5\n");
 scanf("%d%d",&n1,&m1);
 input(aa,n1,m1);
 printf("исходная матрица \n");
 out(aa,n1,m1);
 printf("размер матрицы n1=%3d,m1=%3d \n",n1,m1);
 printf("массив сумм элементов строк \n ");
 for (i=0;i<n1;i++)
 {//представляем каждую строку матрицы как отдельный
   //массив и находим сумму его элементов
  for (j=0;j<m1;j++)
   b[j]=aa[i][j];
```

```
 printf("%5d",summ(b,m1));
 }
 printf("\n");
}
```

Рис.4

Контрольные вопросы

1. Что такое функция и как определить результат ее работы?
2. Как описать функцию?
3. Как и где происходит вызов функции?
4. Что такое формальные и фактические параметры?
5. Какое соответствие должно быть между формальными и фактическими параметрами?
6. Какие существуют особенности передачи параметров в функцию?
7. Что такое встроенная функция?

Варианты заданий

1. Составить функцию для определения суммы элементов одномерного массива. Используя ее, найти сумму элементов произвольно заданной матрицы.
2. С помощью двух различных функций найти минимальный и максимальный элементы массива.
3. Составить программу для вычисления максимального из двух целочисленных значений. Используя ее, найти максимальное число из шести.
4. Создать три функции, которые вычисляют сумму, разность и произведение двух чисел. Используя их, вычислить выражение $z = \dfrac{x - a^2}{2x^3 + \cos(x) \cdot \left(3x + \dfrac{\pi}{9}\right)}$.
5. Составить программу нахождения корня уравнения методом половинного деления. Найти корни уравнения для функций *f1(x)=x²-cos(x)=0* и *f2(x)=sin(x²).*
6. Составить функцию для упорядочивания по убыванию элементов одномерного массива. Используя ее отсортировать элементы столбцов произвольно заданной матрицы.
7. Составить функцию для нахождения среднего арифметического элементов одномерного массива. Используя ее, найти среднее арифметическое элементов произвольно заданной матрицы.

8. Составить функцию для вычисления $a^x = e^{x\ln a}$, где $a>0$. Используя ее, вычислить выражения

$$y=\frac{2^{3x+2}+3^{4x}}{\left(\frac{1}{2}\right)^x+\left(\frac{1}{3}\right)^{2x}},\quad z=\sin\frac{x}{2}+(x+2)^{x+1}.$$

9. Составить функцию для перевода целого числа из десятичной системы счисления в двоичную. Используя ее, представить 106, 999 и 235 в двоичном коде.

10. Составить функцию для преобразования числа из двоичного кода числа в десятичный. Используя ее, сравнить десятичные эквиваленты двоичных чисел 0.01110111, 0.00111011, 0.011111 и найти из них максимальный.

11. Составить функцию определения суммы цифр целого числа. Используя ее, найти число из шести, предложенных с минимальной суммой цифр.

12. Составить функцию для вычисления $tgx=\frac{\sin(x)}{\cos(x)}$, $ctgx=\frac{\cos(x)}{\sin(x)}$, где $\left(x\neq\frac{\pi}{2};\quad x\neq 0;\quad x=\pi\right)$. Используя ее, вычислить выражения

$$y=\frac{2tg\frac{x}{2}+3ctg\frac{x}{2}}{4+tg\frac{x}{4}},\quad z=\frac{x^2+3{,}4\cdot 10^{-3}}{x^3+tgx}.$$

13. Составить функцию для вычисления $f(n,k)=\sin\frac{\pi\cdot n}{k}$. Используя ее, вычислить выражения

$$y=\frac{\sin\frac{4\pi}{3}+7\sin\frac{5\pi}{2}}{5\sin\frac{8\pi}{5}+\sin\frac{7\pi}{3}},\quad z=\frac{y^2+\sin\frac{\pi}{2}}{1+y}.$$

14. Составить функцию для вычисления площади треугольника $S=\sqrt{p(p-a)(p-b)(p-c)}$, где $P=\frac{a+b+c}{2}$. Используя ее, найти треугольник с максимальной площадью: Δabc стороны 3, 7 и 8 см, Δfde, стороны 9, 9 и 6 см Δgkl, стороны 13, 6 и 8 см.

15. Составить функцию для вычисления $f(a,b,x)=a\sin x+b\cos x$. Используя ее, вычислить выражения

$$y=4\sin x+\frac{5\sin x+3\cos x}{8\sin x+4\cos x},\quad z=tgx+\frac{1{,}3\sin x+1{,}8\cos x}{2{,}5\sin x-4{,}5\cos x}.$$

16. Составить функцию для вычисления $f(n,m,x)=\sqrt[n]{x^m}$. Используя ее, вычислить выражения

$$y=2+\sqrt[3]{\left(\frac{1+x+x^2}{1+3x}\right)^2},\quad z=\frac{1}{2}\sqrt[5]{\left(\frac{x^8+x^7+3}{x+1}\right)^3}.$$

17. Составить функцию для вычисления $f(a,b,x)=\sin(ax+b)$. Используя ее, вычислить выражения

$$y=\frac{\sin(2x+1)+\sin(3x+0{,}5)}{1+\sin\frac{\pi}{8}},\quad z=\frac{1+\sin(3x)}{2+\sin\left(3x+\frac{\pi}{9}\right)}.$$

18. Составить функцию, определяющую количество положительных элементов в одномерном массиве. Используя ее, составить массив из количества положительных элементов столбцов произвольно заданной матрицы.

19. Составить функцию для вычисления корней уравнения $ax^2+bx+c=0$. Использовать ее для различных целых аргументов *a, b, c*.

20. Составить функцию для вычисления суммы элементов одномерного массива. Используя ее, найти сумму элементов выше и ниже главной диагонали квадратной матрицы.