Сибирский государственный университет

телекоммуникаций и информатики

Методические указания

к лабораторным работам

по курсу «Программирование на языках высокого уровня»

Язык программирования С

Новосибирск

1 часть.

**Лабораторная работа №1**

# Работа в интегрированной среде С на примере программ линейной структуры

1. ЦЕЛЬ РАБОТЫ: приобретение практических навыков работы в интегрированной среде C, изучение структуры программы на языке С.

2. ОСНОВНЫЕ СВЕДЕНИЯ

Чаще всего линейные алгоритмы используются для программирования вычислений по формулам. В этом случае удобно использовать набор стандартных функций Си, содержащихся в заголовочном файле math.h :

Sin (x) - sin x; cos (x) – cos x ; tan(x) - tg х ; log (x) - ln x; log10(x) – lg x; exp(x) - показательная функция ex ; sqrt (x)- корень квадратный от х; роw(x, y) – x в степени y; abs(x)- модуль x.; acos(x) – arccos x; asin(x) – arcsin x; atan(x)- arctg x; sinh(x)- sh x; cosh (x) – ch x; tanh(x)-th x.

Для тригонометрических функций аргумент х измеряется в радианах и имеет тип double, как и значения функций.

Пример программы линейной структуры

Вычислить площадь и стороны прямоугольного треугольника, если известны гипотенуза c и угол x. Для вычислений воспользуемся формулами :

a=c⋅sin x; b=c⋅cos x; S=ab;

Программа вычислений имеет вид:

//lab1\_prim.c вычисление высот и площади треугольника

#include <stdio.h> //заголовочный файл для организации ввода-вывода

#include <math.h> //заголовочный файл математических функций

main() //главная функция

{ float a,b,c,x, S; //описание переменных вещественного типа

printf (“\n Введите гипотенузу и угол треугольника ”);

scanf (“%f%f “,&c,&x); // ввод значений вещественного типа

a=c\*sin(x);

b=c\*cos(x);

S=a\*b;

printf (“\n Результаты: a=%7.2f b=%7.2f S=%7.2f “,a, b, S);

return 0;

}

Комментарий в программе не влияет на компиляцию программы, а служит для разъяснения действий основных блоков текста и программы в целом. Однострочный комментарий действует от двух символов // до конца строки. Многострочный комментарий заключается в пары символов /\* и \*/.

Вторая строка программы #include <stdio.h> является директивой компилятора для включения заголовочного файла stdio.h , в котором содержатся средства ввода-вывода (связи с внешними устройствами), отсутствующие в самом языке Си. Третья строка - #include<math.h> является директивой компилятора для включения заголовочного файла math.h, обеспечивающего выполнение математических функций.

Программа на языке Си состоит из ряда функций, из которых функция main (главная) является обязательной и служит точкой входа в программу. В круглые скобки заключаются параметры функции, причем наличие круглых скобок обязательно, если даже список параметров пуст. В фигурные скобки заключаются составной оператор (несколько операторов). Для придания тексту программы наглядности открывающая и соответствующая ей закрывающая фигурные скобки печатаются на одном уровне, а заключенный между ними текст, сдвигается на 1-2 символа вправо, вложенный блок также сдвигается вправо и т.д. Образуется иерархия вложенных блоков, придающая программе на Си характерный вид.

В программе описаны переменные a, b, c, х, S вещественного типа (float). В отличие от других языков в Си учитывается регистр при определении имени переменной, т.е. s и S – разные переменные. Функции printf и scanf содержатся в заголовочном файле stdio.h и служат для вывода на экран и ввода с клавиатуры соответственно. Управляющие символы \n в функции printf служат для перевода на новую строку. Символ & в функции scanf указывает на адрес вводимой переменной. Ввод и вывод переменных вещественного типа производится в формате f. Признаком форматного вывода в функции printf является %. При выводе между знаком процента и форматной переменной f можно включить общую ширину поля вывода и число позиций после десятичной точки. Оператор return 0; служит для выхода из функции main в Интегрированную Среду, 0 является признаком успешного окончания программы. Завершает текст программы закрывающая фигурная скобка, означающая конец функции main.

3. ВЫПОЛНЕНИЕ РАБОТЫ

3.1. Наберите и выполните приведенную выше программу вычисления высот треугольника по заданным сторонам.

3.2. Составьте и выполните программу линейной структуры согласно вариантам заданий.

## Варианты заданий

Вычислить значение функции переменных при заданных значениях параметров:

1. x=4y2/(4z-2t3) при t=1 ; z=3; y=sin t.

2 x=4y3-z/t при t=2; z=3; y=cos(t+z).

3. x=6t2-(z+1)/y2 при y=2; z=4; t=sin(2+z).

4. x=(8z2+1)/(y+t2) при z=1; t=2; y=t+z.

5 x=8z / (et+2)-y2 при t=3; z=ctg t +2; y=4.

6. x=8z/(et+2)-y2 при t=1; z=t+2; y=4.

7. x=2y+3 sh t- z при y=2; t=5 / (1+y2); z=4

8. x=3 y2/ (4 tg z-2t2) при t=0.5; z=6; y=t+2 ctg z.

9. x=4y2 /( 4y ez - 2t3) при t=1 ; z=3; y=sin t.

10. x=4 ln y3-z / t при t=2; z=3; y=cos(t+z).

11. x=6 t2- (ctg z+1)/ y2 при y=2; z=4; t=sin(2+z).

12. x=(8z2+1)/( y et +t2) при z=1; t=2; y=tg t+z.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Что такое алгоритм линейной структуры, программа линейной структуры?

4.2. Для чего используются заголовочные файлы?

4.3. Структура программы на языке Си.

**Лабораторная работа №2**

**Программы разветвлённой структуры**

1. ЦЕЛЬ РАБОТЫ: приобретение практических навыков в составлении алгоритмов и программ разветвленной структуры.

2. ОСНОВНЫЕ СВЕДЕНИЯ

Алгоритм разветвленной структуры - это алгоритм, в котором вычислительный процесс осуществляется по одной из ветвей. Если количество ветвей две – то используется условный блок, если больше – то множественный выбор. В программах используют соответственно условный оператор IF или оператор SWITCH для реализации разветвленного алгоритма.

2.1. Условный оператор IF

Условный оператор в языке Си имеет формат:

if (условие) оператор1; else оператор2; (полная форма) или

if (условие) оператор1; (сокращенная форма).

Если оператор1 и оператор2 состоят из одного оператора, то в фигурные скобки они не заключаются. Если оператор1 и/или оператор2 являются составным оператором (несколько операторов), то он заключается в фигурные скобки. Условие всегда заключается в круглые скобки.

В качестве примера приведем программу вычисления наибольшего из значений функции y1=x²+1, y2=7-x² , y3=x+1 для любого х.

//lab2\_prim1.c

#include <stdio.h>

main()

{ float x,y1,y2,y3,max;

printf("Введите x\n"); scanf(“%f “,&x);

y1=x\*x+1; y2=7-x\*x; y3=x+1;

if (y1>y2) max=y1; else max=y2;

if (y3>max) max=y3;

printf (“ x=%8.2f y1=%8.2f y2=%8.2f y3=%8.2f max=%8.2f\n”,x,y1,y2,y3,max); }

2.2 Множественный выбор

В программах с множественным выбором используется переключатель switch , который сравнивает значение выражения, указанного за ним, и выполняет один из операторов, метка которого совпадает с этим значением. Общий вид:

Switch (выражение)

{ case метка\_1: список\_операторов\_1;

……………………………………….

case метка\_n: список\_операторов\_n;

default : операторы; }

Значения выражения и меток должны быть целочисленными константами.

Например, определим количество дней по введенному номеру месяца.

//lab2\_prim2.c множественный выбор

#include<stdio.h>

#include<conio.h> // Заголовочный файл для работы с терминалом

int m;

main()

{

printf("\n Введите номер месяца :"); scanf("%d",&m);

printf("\n В %4d месяце дней: ",m);

switch(m){

case 1: case 3: case 5: case 7: case 8: case 10: case 12:printf("тридцать один\n");break;

case 2: printf ("двадцать восемь \n");break;

case 4: case 6: case 9: case 11: printf("тридцать\n");break;

default: printf("\n Номер месяца неверен \n");}

getch ();

}

В примере программы, если номер месяца превышает 12, выводится сообщение о неверном вводе месяца, для чего используется default. Оператор break служит для прерывания цикла проверки и перехода в конец переключателя. В случае отсутствия break, происходит переход на следующую ветвь. Функция getch, подключающаяся заголовочным файлом conio.h, ожидает нажатия любой клавиши. Это позволяет просмотреть результаты, не используя ALT+F5 для просмотра экрана пользователя.

3. ВЫПОЛНЕНИЕ РАБОТЫ

Составить программы разветвляющейся структуры согласно вариантам задания 1 (используя IF) и задания 2 (используя SWITCH) .

Варианты задания 1

1. Даны четыре числа. На сколько их сумма меньше их произведения?
2. Даны четыре числа. Вычислить сумму положительных среди них чисел.
3. Даны четыре числа. Вычислить произведение отрицательных среди них чисел.
4. Даны четыре числа. Все отрицательные среди них числа заменить на 0.
5. Даны четыре числа. Определить сколько среди них отрицательных и сколько положительных.
6. Даны четыре числа. Определить порядковый номер наименьшего среди них.
7. Даны два числа. Большее из этих двух чисел заменить их полусуммой, а меньшее удвоенным произведением.
8. Даны три числа. Меньшее среди них заменить на 0 .

9) Даны четыре числа. Найти разность между наибольшим и наименьшим среди них.

10) Даны три числа K, M и N. Поменять их значения местами таким образом, чтобы K < M < N.

1. Даны четыре разных числа. Найти среди них два наибольших.
2. Даны три числа. Поменять местами большее и меньшее из этих чисел.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Сравните виды условных операторов Си.

4.2. Всегда ли ставятся скобки и какие в условном операторе?

4.3. Зачем ставятся в переключателе операторы break и default?

4.4. Как будет работать программа из примера без break?

**Лабораторная работа №3 (РГЗ)**

**Программы циклической структуры**

1. ЦЕЛЬ РАБОТЫ: приобретение практических навыков в составлении алгоритмов и программ циклической структуры.

2. ОСНОВНЫЕ СВЕДЕНИЯ

2.1. Операторы циклов

В языке Си существует 3 вида циклов: 1) цикл с параметром или цикл типа for; 2) цикл с предусловием или цикл типа while, 3) цикл с постусловием или цикл типа do ... while . Во всех этих циклах условие продолжения цикла заключается в круглые скобки. В циклах типов for и while повторяющаяся часть состоит из одного оператора, если требуется выполнить в цикле несколько операторов, они заключаются в фигурные скобки, образуя составной оператор. В цикле с постусловием тело цикла помещается между словами do и while. В отличие от цикла с предусловием, цикл с постусловием выполнится хотя бы один раз. Циклы с пред- и постусловием продолжаются, если условие продолжения истинно.

2.2. Цикл с параметром

Для цикла типа for заголовок состоит из трех разделов: инициализации (присваивания начальных значений), проверки условия повторения, модификации (изменения параметров). Разделителем между разделами заголовка служит запятая.

Пример вычисления суммы квадратов натурального ряда чисел от 1 до n.

S= = 12 + 2 2 + . . . + n 2.

//lab3\_1.c сумма квадратов натурального ряда

#include <stdio.h>

main()

{ int S,n,i;

printf(“Введите n ”); scanf(“%d”,&n);

for (S=0, i=1; i<=n; i++)

S+=i\*i; // соответствует S=S+i\*i

printf (“n=%d S=%d”,n,S);

}

В разделе инициализации присваиваются начальные значения переменным S и i. При i, не превышающем n, цикл повторяется, иначе выполняется выход из цикла. Оператор инкремента i++ (или декремента i--), эквивалентен i=i +1 (i=i-1). Для досрочного выхода из цикла и перехода на следующий оператор после цикла используется оператор break. Для пропуска всех операторов, оставшихся до конца тела цикла и перехода к следующему повторению цикла используется оператор continue.

2.3 Циклы с неизвестным количеством повторений

Изменим условие в предыдущем примере: Определить число n, при котором сумма чисел S предыдущего ряда не превысит величину K, введенную с клавиатуры. Такую программу можно реализовать с помощью циклов предусловия или постусловия следующим образом:

//lab3\_2 циклы while и do…while

#include<stdio.h>

#include<conio.h>

main()

{int K,S, i;

clrscr(); // очистка экрана перед выводом

printf (“Введите К ”); scanf (“%d”, &K);

// цикл с предусловием //цикл с постусловием

S=0; S=0; i=0;

i= 0; do

while ( S<K) { i++ ; S+=i\*i }; { i++; S+=i\*i; }

while ( S<K) ;

printf (“n=%d S=%d\n”, i, S);

getch();

}

Отличие этих программ: если ввести К<1, то в программе с предусловием получим S=0, i=0 ; а в программе с постусловием S=1, i=1.

3. ВЫПОЛНЕНИЕ РАБОТЫ

Составить 3 варианта программ циклической структуры типа for и while, сравнить полученные результаты. x, n ввести с клавиатуры.

Варианты заданий

1.  . 2.  .

3.  . 4.  .

5.  . 6.  .

7.  8. 

9.  . 10.  .

11.  12. 

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Какой алгоритм является алгоритмом циклической структуры?

4.2. Типы циклов в языке Си.

4.3. Какой цикл выполнится хотя бы один раз? А какой - ни разу?

2 часть

**Лабораторная работа №4**

**Обработка одномерных массивов**

1. ЦЕЛЬ РАБОТЫ: приобретение практических навыков в составлении программ с массивами.

2. ОСНОВНЫЕ СВЕДЕНИЯ

Массивы - структурированный тип данных с элементами одного типа, имеющий одно имя и определенное количество элементов. Количество элементов определяет размер массива. Порядковый номер элемента массива называется его индексом. Число индексов называется размерностью массива, например, массив с двумя индексами называется двумерным массивом. Строка символов является массивом символов, вектор – массив чисел, матрица – массив векторов. Обработка массивов выполняется следующим образом: описание, ввод или инициализация элементов массива, преобразование и вывод.

При описании массивов после имени в квадратных скобках указывается число его элементов. В языке Си индекс элемента массива начинается с нуля, поэтому индекс последнего элемента массива на 1 меньше числа элементов в данном массиве.

Пример обработки одномерного массива

Дан массив из 50 целых чисел. Найти наибольший элемент в массиве и его порядковый номер.

# include <stdio.h>

#include <conio.h>

# define n 50 // определение константы n=50

int i, max, nom, a[n]; //описание массива целых чисел из n элементов

main()

{ for (i=0; i<n; i++)

{ printf( “\n Введите элемент массива ”);

scanf ("%d", &a[i]) };

for (i=1,max=a[0],nom=0; i<n; i++)

if (max<a[i])

{nom=i; max=a[i];}

printf("\n Вывод элементов исходного массива : \n");

for (i=0; i<n; i++) printf ( "%6d", a[i] );

printf ("\n Максимальное число в массиве %4d, его индекс %4d " , max, nom+1);

getch();

}

3. ВЫПОЛНЕНИЕ РАБОТЫ

Даны вещественные числа a,b. Значения функции (согласно вариантам) записать в массив. Вычислить значение интеграла, используя:

1) Формула трапеций

I1=h\*(a[0]/2+a[1]+a[2]+…+a[N-1]+a[N]/2)

2) Формула Симпсона

I2=h/3\*( a[0]+a[N] + 4\*(a[1]+a[3]+…+a[N-1]) +

2\*(a[2]+a[4]+…+a[N-2]))

h=(b-a)/n, n=100.

1. f(x)=x2/(10+x3); a=-2; b=5;

2. f(x)=(2.5x2-0.1)/(tg x+sin x); a=4; b=6;

3. f(x)=(x+1)2√lg x; a=2; b=10;

4. f(x)=x2 ln x /(1+x)2; a=1; b=20;

5. f(x)=1/((0.5+0.1x3)√x); a=0.1; b=2.1;

6. f(x)=x 2√(2+3x)3; a=0.5; b=2.5;

7. f(x)=1/√(0.02+0.01x); a=1; b=30;

8. f(x)=(1+2x+x2)/(5+2x2); a=-2; b=2;

9. f(x)=(2x+lg x)/(1+lg x); a=1; b=10;

10. f(x)=√(2+x)3/x2; a=0.2; b=10;

11. f(x)=(1+x2)/(x3+√(1+x)); a=0.5; b=5;

12. f(x)=(1-x) lg x/√(1-lg x); a=2; b=7;

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Что такое массив? индекс элемента массива?

4.2. Как ввести и вывести элементы вектора в строку и в столбец?

4.3. Способы описания и инициализации массивов.

**Лабораторная работа №5**

**Матрицы (двумерные массивы)**

1. Цель работы: программирование алгоритмов обработки двумерных массивов

1. Основные сведения

Матрица – это прямоугольная таблица элементов (например, чисел или символов). Матрица представляется в виде двумерного массива, то есть массива, все элементы которого имеют два индекса. Матрица, как и таблица, состоит из строк и столбцов. Два индекса элемента - это и есть номера строки и столбца, на пересечении которых этот элемент находится. В языке Си каждый индекс записывается отдельно в квадратных скобках. Каждую строку и каждый столбец матрицы можно рассматривать как обычный одномерный массив. Поэтому можно сказать, что матрица – это массив из массивов.

Первый индекс элемента матрицы – это строка, второй – столбец. Поэтому когда говорят о «матрице 4 на 5», это означает, что матрица имеет 4 строки и 5 столбцов. Матрицы, у которых число строк равно числу столбцов, называют квадратными. В квадратных матрицах можно выделить главную диагональ – это все элементы, у которых номер строки равен номеру столбца, то есть A[0][0], A[1][1], ..., A[N-1][N-1] для матрицы размером N на N.

Объявление матриц

Матрицы объявляются так же, как и простые массивы, но у них не один индекс, а два. При объявлении в отдельных квадратных скобках указывается количество строк и количество столбцов. Например, оператор

int B[10][10];

выделит место в памяти под матрицу целых чисел, имеющую 10 строк и 10 столбцов. Если матрица глобальная (объявляется выше всех процедур и функций), то она в самом начале заполняется нулями. Локальные матрицы (объявленные внутри процедуры или функции) первоначально содержат «мусор» – неизвестные значения.

Начальные значения элементов

При объявлении можно сразу задать все или часть ее элементов, например:

float X[2][3] = {{1., 2., 3.},{4., 5., 6.}};

Как видно из примера, элементы каждой строки заключаются в отдельные фигурные скобки. Если задать не все элементы, то остальные заполнятся нулями:

float X[2][3] = {{1., 3.},{6.}};

Здесь элементы X[1][2], X[2][1] и X[2][2] будут нулевыми.

Расположение матриц в памяти

Во всех современных языках программирования элементы матрицы располагаются по строкам, то есть сначала изменяется последний индекс. Объявленная выше матрица X расположена так:

X[0][0] X[0][1] X[0][2] X[1][0] X[1][1] X[1][2]

Стандартный ввод и вывод

Для работы с матрицами требуется вложенный цикл, то есть цикл в цикле.

#include <stdio.h>

const int M = 5; // число строк

const int N = 4; // число столбцов

main()

{

int i, j, A[M][N];

for ( i = 0; i < M; i ++ ) // цикл по строкам

for ( j = 0; j < N; j ++ ) // цикл по столбцам строки (элементы строки)

{

printf ("A[%d][%d]=", i, j); // подсказка для ввода

scanf ("%d", & A[i][j]); // ввод A[i][j]

}

// работа с матрицей

}

Заполнение случайными числами

Выполняется также в двойном цикле аналогично одномерным массивам. В примере показано заполнение целой матрицы случайными числами в интервале [a,b].

for ( i = 0; i < M; i ++ )

for ( j = 0; j < N; j ++ )

A[i][j] = random(b-a+1) + a;

Вывод на экран

При выводе матрицы ее элементы желательно расположить в привычном виде – по строкам, т.е. вывели одну строку матрицы, перешли на новую строку экрана, и т.д. Надо учитывать, что для красивого вывода на каждый элемент матрицы надо отвести равное количество символов (иначе столбцы будут неровные). Делается это с помощью форматирования – цифра после знака процента задает количество символов, отводимое на данное число.

printf("Матрица A\n");

for ( i = 0; i < M; i ++ ) // цикл по строкам

{

for ( j = 0; j < N; j ++ ) // вывод одной строки (в цикле)

printf ( "%4d", A[i][j] ); // 4 символа на число

printf("\n"); // переход на другую строку

}

Алгоритмы для работы с матрицами

1) Поиск минимального элемента

В отличие от одномерных массивов, для перебора всех элементов матрицы надо использовать двойной цикл. Ниже показано, как найти минимальный элемент в массиве и его индексы. Сначала считаем, что минимальным является элемент A[0][0], а затем проходим все элементы, проверяя, нет ли где еще меньшего. Можно запоминать только индексы, а значение минимального элемента получать прямо из массива.

float A[M][N], i, j, row, col;

...

row = col = 0; // сначала считаем, что A[0][0] - минимальный

for ( i = 0; i < M; i ++ ) // просмотр всех строк

for ( j = 0; j < N; j ++ ) // просмотр всех столбцов

if ( A[i][j] < A[row][col] )

{

row = i; // запомнили новые индексы

col = j;

}

printf ("Минимальный элемент A[%d][%d]=%d",row, col, A[row][col]);

2) Работа с отдельными элементами

Рассмотрим квадратную матрицу N на N. Выведем на экран обе ее диагонали (главную диагональ и перпендикулярную ей). С главной диагональю все просто – в цикле выводим все элементы, у которых номера строки и столбца равны, то есть A[i][i] для всех i от 0 до N-1.

Вторую диагональ формируют такие элементы:

A[0][N-1], A[1][N-2], A[2][N-3], ..., A[N-1][0]

Обратим внимание, что каждый следующий элемент имеет номер строки на 1 больше, а номер столбца – на 1 меньше. Таким образом, сумма номеров строки и столбца постоянна и равна N-1. Тогда, зная номер строки i можно сразу сказать, что на второй диагонали стоит ее элемент A[i][N-1-i].

3) Перестановка строк и столбцов

Пусть надо переставить две строки с индексами i1 и i2. Это значит, что для каждого столбца j надо поменять местами элементы A[i1][j] и A[i2][j] через временную переменную temp.

for ( j = 0; j < N; j ++ )

{

temp = A[i1][j];

A[i1][j] = A[i2][j];

A[i2][j] = temp;

}

Пример, объединяющий некоторые типовые алгоритмы работы с матрицей.

Даны две матрицы А и В одинакового размера m x n, заполненные случайными числами в диапазоне от 0 до 19. Получить матрицу C=max(A[i.j],B[i,j]) и матрицу D=min(A[i,j],B[i,j]).

#include <stdio.h>

#include <conio.h>

#include <stdlib.h> //для генерации случайных чисел

const int m=4, n=4;

int random(int r)

{ return rand()%r;

}

main()

{

int i,j,a[m][n],b[m][n],c[m][n],d[m][n];

system("CLS");

srand(11); //инициализация датчика случайных чисел

for(i=0;i<m;i++)

for(j=0;j<n;j++)

a[i][j]=random(20);

for(i=0;i<m;i++)

for(j=0;j<n;j++)

b[i][j]=random(20);

// вывод матриц а и b

printf("\n матрица a\n");

for (i=0;i<m;i++)

{ printf("\n");

for (j=0;j<n;j++)

printf("%4d",a[i][j]);

}

printf("\n\n матрица b\n");

for (i=0;i<m;i++)

{ printf("\n");

for (j=0;j<n;j++)

printf("%4d",b[i][j]);

}

// формирование матриц c и d

for (i=0;i<m;i++)

for (j=0;j<n;j++)

if (a[i][j]>b[i][j])

{ c[i][j]=a[i][j];

d[i][j]=b[i][j];

}

else

{ c[i][j]=b[i][j];

d[i][j]=a[i][j];

}

// вывод матриц c и d

printf("\n\n матрица c\n");

for (i=0;i<m;i++)

{ printf("\n");

for (j=0;j<n;j++)

printf("%4d",c[i][j]);

}

printf("\n\n матрица d\n");

for (i=0;i<m;i++)

{ printf("\n");

for (j=0;j<n;j++)

printf("%4d",d[i][j]);

}

puts("\n");

system("PAUSE");

}

1. Выполнение работы

### Варианты задания (номер варианта определяется по последней цифре номера студенческого билета):

**0.** Сформировать двумерный массив A размером N x 2 (N - количество строк, равное предпоследней цифре номера студенческого билета+2; 2 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Найти наибольший элемент каждой строки матрицы А. Из этих максимальных элементов составить одномерный массив F. Вывести элементы массива F.

**1.** Сформировать двумерный массив B размером N x 3 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 3 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Составить одномерный массив D из минимальных элементов столбцов матрицы. Вывести элементы массива D.

**2.** Сформировать двумерный массив B размером N x 4 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 4 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Вычислить и вывести строку матрицы B, содержащую максимальное количество отрицательных.

**3.** Сформировать двумерный массив C размером N x 5 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 5 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Вычислить сумму положительных элементов в каждой строке матрицы C . Из полученных сумм составить одномерный массив D. Вывести элементы массива D.

**4.** Сформировать двумерный массив D размером N x 6 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 6 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива на экран и в файл (см. п.3.4. и п.3.6.). В каждом столбце матрицы D найти максимальный элемент. Среди найденных чисел найти минимальное и вывести на экран и в файл.

**5.** Сформировать двумерный массив E размером N x 7 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 7 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Вычислить сумму отрицательных элементов столбца, в котором находится максимальный элемент матрицы E. Вывести полученную сумму и номер столбца.

**6.** Сформировать двумерный массив F размером N x 8 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 8 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. В строке матрицы F, содержащей максимальный элемент, заменить все отрицательные числа на нули. Вывести номер этой строки и максимальный элемент.

**7.** Сформировать двумерный массив G размером N x 9 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 9 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Записать в одномерный массив F строку матрицы G, содержащую минимальный элемент. Вывести элементы массива F.

**8.** Сформировать двумерный массив L размером N x 10 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 10 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Вывести номер строки и саму строку матрицы L, в которой сумма положительных элементов максимальна.

**9.** Сформировать двумерный массив Z размером N x 11 (N – количество строк, равное предпоследней цифре номера студенческого билета +2; 11 – количество столбцов) с помощью генератора случайных чисел и вывести элементы массива. Вычислить сумму положительных элементов строки, в которой находится минимальный элемент матрицы Z. Вывести полученную сумму и номер строки.

1. Контрольные вопросы
   1. Как объявить матрицу? Как инициализировать?
   2. Ввод – вывод матрицы по строкам.
   3. Как поменять местами строки в матрице?
   4. Как найти минимальный элемент матрицы и его индексы?
   5. Как вычислить сумму элементов строки матрицы?
   6. Как получить транспонированную матрицу?

Лабораторная работа № 6 (КР)

Функции

1. ЦЕЛЬ РАБОТЫ: Изучение методов использования функций языка Си.

2. ОСНОВНЫЕ СВЕДЕНИЯ

Часто в программе требуется повторить определенную последовательность операторов в разных частях программы. Для того, чтобы описывать эту последовательность один раз, а применять многократно, в языках программирования применяются подпрограммы. Подпрограмма - автономная часть программы, выполняющая определенный алгоритм и допускающая обращение к ней из различных частей общей программы.

В языке Си существует один вид подпрограмм - функции. Каждая программа в своем составе должна иметь главную функцию main(), служащую точкой входа в программу. Кроме функции main(), в программу может входить произвольное число функций, выполнение которых инициализируется либо прямо, либо вызовами из функции main(). Каждая функция по отношению к другой является внешней. Для того, чтобы функция была доступной, необходимо, чтобы до ее вызова о ней было известно компилятору. Форма записи функции:

<тип > <имя функции>(<формальные параметры>){<тело функции >}

Если тип возвращаемого функцией значения не указан, то подразумевается int. Если с именем функции не связан результат, то нужно указать тип функции void. Параметры, записываемые в обращении к функции, называются фактическими; параметры, указанные в описании функции - формальными. Фактические пара­метры должны соответствовать формальным по количеству, порядку следования и типу. Объекты, объявленные вне функции, действуют в любой функции и называются глобальны­ми. Объекты, объявленные в функции, действуют только в ней и называются локальными. В теле функции обычно присутствует оператор return <выражение>, определяющий возвращаемое функцией значение.

Все параметры функции, кроме массивов, передаются по значению, т.е. внутри функции создаются локальные копии параметров. Если необходимо передать саму переменную, а не её копию, то в функцию передаётся адрес этой переменной. Таким образом, через параметры можно передавать результат выполнения функции. То есть, параметры, с помощью которых результаты должны передаваться из функции в точку вызова, описываются как указатели. Вызов функции может быть оформлен в виде оператора, если с именем функции не связано возвращаемое значение, или в виде выражения, если возвращаемое значение связано с именем функции.

Прототип функции может указываться до вызова функции вместо описания функции для того, чтобы компилятор мог выполнить проверку соответствия типов аргументов и параметров. Прототип функции по форме такой же, как и заголовок функции. В конце него ставится «;».

Функции можно подключать с помощью директивы #include <имя файла>. Такие файлы с функциями удобно использовать в диалоговых программах с пользовательским меню, позволяющих выбрать один из режимов.

Пример 1: Функция с параметрами-значениями. Результат связан с именем функции. В программе объявляется прототип функции, а сама функция описывается ниже.

//lab8\_1

#include <stdio.h>

#include <conio.h>

int max(int,int); //Прототип функции

void main()

{ int x,y,z;

printf(" input x,y ");

scanf("%d%d",&x,&y);

z=max(x,y); //Вызов функции с фактическими параметрами

printf("x=%d y=%d max=%d",x,y,z);

getch();

}

int max(int a ,int b) //Заголовок функции с формальными параметрами

{ int c;

if (a>b) c=a;

else c=b;

return c;

}

Пример 2: Функция с параметрами-указателями. Здесь передаются адреса фактических параметров, по которым и получаем результат. Функция меняет местами переменные x,y.

//lab8\_2

#include <stdio.h>

#include <conio.h>

main()

{ float x,y;

void swap(float \*, float \*); // Прототип функции с параметрами - указателями

printf("\n введите x,y ");

scanf("%f%f",&x,&y);

swap(&x,&y); // Передаём адреса переменных

printf("\n x=%4.2f y=%4.2f ",x,y);

getch();

}

void swap(float \* a, float \* b)

{float c;

c=\*a; // \*a - содержимое по адресу a

\*a=\*b;

\*b=c;

}

Пример 3: Подключение файлов с функциями и создание меню.

! Внимание! Следите за тем, чтобы константы, объявленные директивой #define, не переобъявлялись в функциях.

//lab8\_3

#include <stdio.h>

#include <conio.h>

#include "lab3.c"

#include "lab5.c"

#include "lab6.c"

main()

{ int nom;

while(1)

{ clrscr();

printf("\n 1. Сумма ряда \n");

printf(" 2. Матрица \n");

printf(" 3. Строки \n");

printf(" 4. Выход \n");

scanf("%d",&nom);

switch(nom)

{

case 1:lab3();break;

case 2:lab5();break;

case 3:lab6();break;

case 4:return 0;

default:printf("Неверный режим");

}

}

getch();

}

Пример 4: Передача в функцию массива с использованием указателя. Результат – элементы массива возводятся в квадрат. Функция описывается до вызова, поэтому прототип не объявляется.

//lab8\_4

#include <stdio.h>

#include <conio.h>

void quart(int n, float \* x) // Можно void quart(int n, float x[])

{ int i;

for (i=0;i<n;i++)

x[i]=x[i]\*x[i];

}

main()

{ float z[]={1,2,3,4};int j;

clrscr();

for (j=0;j<4;j++)

printf(" %6.2f",z[j]);

quart(4,z);

for (j=0;j<4;j++)

printf("\n %6.2f",z[j]);

getch();

}

3. ВЫПОЛНЕНИЕ РАБОТЫ

Используя функции, написать программу по своему варианту.

Варианты заданий

1. Написать функцию, выводящую в порядке возрастания элементы одномерного массива. В главной программе вызвать функцию для двух разных массивов.

2. Написать функцию вычисления произведения прямоугольной матрицы A размера k x m на прямоугольную матрицу B размера m x n. В главной программе обратиться к этой функции.

3. Написать функцию вычисления суммы ряда s=s(1)+…+s(n),

где s(n)=(-1)n x(2n-1)/(2n+1) с точностью до eps=0.001. В качестве параметров выбрать x и eps.

4. Написать функцию, которая вычисляет для заданной квадратной матрицы A её симметричную часть S(ij)=(A(ij)+A(ji))/2 и кососимметричную часть K(ij)=(A(ij)-A(ji))/2.

5. Написать функцию “шапочка” f(x), зависящую от параметров a и b: если |x| >a то f(x)=0 иначе f(x)=b\*exp(-a2/(a2-|x|2)). В качестве параметров передать a,b,x.

6. Написать функцию поиска максимального и минимального элементов одномерного массива. В основной программе вызвать эту функцию для двух разных массивов.

7. Написать функцию, которая сортирует одномерный массив в порядке убывания методом пузырька. В основной программе вызвать эту функцию для двух разных массивов.

8. Написать функцию, которая по двум заданным одномерным массивам (A размера m и B размера n) вычисляет максимальное значение двумерного массива c(ij)=a(i)\*b(j).

9. Написать функцию вычисления суммы ряда y=x-x/2+…

+(-1)n+1x/n с точностью до eps=0.001. В качестве параметров передать x и eps.

10. Написать функцию вычисления ряда y=x+x3/3!+…+x2n+1/(2n+1)! с точностью до eps=0.0001. В качестве параметров передать x и eps.

4. КОНТРОЛЬНЫЕ ВОПРОСЫ

4.1. Описание функции. Для чего объявляется прототип?

4.2. Что такое формальные и фактические параметры? Локальные и глобальные?

4.3. Как можно передавать массив в функцию?

4.4. Способы вызова функций.