**Лабораторная работа № 6**

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

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include <time.h>

#include <stdlib.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()

{setlocale(0,"");

int m;

printf("Введите размерность массива ");

scanf("%d",&m);

float z[m];int j;

srand(time(NULL));

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

printf(" %6.2f",z[j]=rand()%10);

quart(m,z);

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

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

getch();

}

**Пример 4:** Подключение файлов с функциями и создание меню.

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

ВСЕ 4 ФАЙЛА ДОЛЖНЫ НАХОДИТЬСЯ В ОДНОЙ ПАПКЕ!!!

Их имена : fun\_menu.c:

lab\_sum.c

lab\_mat.c

lab\_str.c

**1 файл**

#include <stdio.h>

#include <conio.h>

#include <locale.h>

#include "lab\_sum.c"

#include "lab\_mat.c"

#include "lab\_str.c"

main()

{ int nom;

setlocale(0,"");

while(1)

{

printf("\n 1. Cумма ряда\n");

printf("\n 2. Матрица \n");

printf("\n 3. Cтроки\n");

printf("\n 4. Bыход\n");

scanf( "%d",&nom);

switch(nom)

{ case 1: lab\_sum();break;

case 2: lab\_mat();break;

case 3: lab\_str();break;

case 4: return 0;

default: printf("неверный режим");}

}

getch();

}

**2 файл**

//lab\_sum.c сумма квадратов натурального ряда

//#include <stdio.h>

lab\_sum()

{ 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);

}

**3 файл**

//# include <stdio.h>

//#include <conio.h>

# define N 5 // определение константы n=50

int lab\_mat()

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

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

{ printf( "\n Введите элемент массива ");

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

for (i=0,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();

}

**4 файл**

//#include <stdio.h>

//#include <conio.h>

#include <string.h>

lab\_str()

{ char gwords [10] [40], temp [40];

int i=0;

printf ("Введите 10 слов, начинающихся с d: \n");

while ( i < 10 )

{ gets (temp);

if (temp [0] != 'd') // проверка первого символа

printf (" %s не начинается с d! \n", temp);

else { strcpy (gwords [i], temp); i++;}

}

puts ("Следующие слова соответствуют условию:");

for (i=0; i< 10; i++) puts (gwords [i] );

getch();

}

**3. ВЫПОЛНЕНИЕ РАБОТЫ**

3.1. Проанализировать приведенные выше программы.

3.2. Используя функции, написать программы 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 определителя квадратной матрицы A размера 3x3:

detA=a(1,1)a(2,2)a(3,3)+a(3,1)a(1,2)a(2,3)+a(2,1)a(3,2)a(1,3)-a(3,1)a(2,2)a(1,3)-a(1,1)a(3,2)a(2,3)-a(2,1)a(1,2)a(3,3).

10. Написать функцию вычисления суммы ряда y=sinx-(sin2x)/2+…

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

11. 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.

12. Написать функцию обработки матриц A и B одинакового размера m x n. Получить матрицу C =max(a(i,j),b(i,j)), и матрицу D=min(a(i,j),b(i,j)). Матрицы C и D вывести в главной программе.

**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.