# ЛАБОРАТОРНАЯ РАБОТА №4
# ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ

***Цель работы**:* познакомиться с механизмом составления и организации взаимодействия пользовательских функций, рекурсивных функции, указателя на функцию и генерацию случайных чисел.

## Краткие теоретические сведения

## УКАЗАТЕЛИ

**Указатель** – это особая переменная, в которой хранится адрес области памяти. *Указатель* не является самостоятельным типом, он всегда связан с каким-то другим типом. *Указатели* делятся на две категории: *указатели на переменные* и *указатели на функции.*

**Определение указателя**:

**Тип_Данных *имя_указателя;**

**Тип_Данных** – это тип, на который будет указывать указатель и тип может быть любым, кроме ссылки;

***** – определяет тип указатель;

**имя_указателя** – это имя переменной, которая хранит адрес переменной, на которую будет указывать указатель;

***имя_указателя** – это содержание адреса, на который указывает указатель.

С указателями связаны две унарные операции: & и *. Операция & означает «взять адрес», которая допустима только над переменными. Операция * – «значение, расположенное по указанному адресу».

*Пример:*

```
double x=67; //x – это переменная вещественного типа
double *p;//p указатель на вещественный тип
    p=&x;// присвоение адреса переменной x указателю p
    printf("\n%.2lf\n",*p);//выводит 67.00, т.е. *p – это другое имя переменной x
    *p=*p+3;
```

```
printf("%.2lf\n",x);//выводит 70.00
```

Любая переменная имеет область действия и время жизни.

**Область действия** – это та часть в программе, в которой переменная доступна.

**Время жизни** – в течение какого времени переменная хранит свое значение.

Переменные в языке C/C++ могут быть *локальными* или *глобальными*.

Переменная, определенная *внутри блока* или *функции,* является **локальной**. Для **локальной** переменной *область действия* и *время жизни* в той функции или в том блоке, где данная переменная определена.

Переменная, определенная вне любого блока и вне любой функции, является **глобальной**. Для **глобальной** переменной *область действия* и *время жизни* в течение всей программы в том проекте, в котором она определена, кроме того блока или функции, где данная переменная переопределена.

## ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

Чтобы уменьшить сложность программы, ее разбивают на части. В языке C/C++ задача может быть разделена на более простые подзадачи для выполнения с помощью функций.

**Функция** – это именованная последовательность описаний и операторов, выполняющая законченное действие.

**Функция имеет следующий вид:**

**ТипВозвращаемогоЗначения ИмяФункции**([**СписокПараметров**])

{

**тело функции**

}

**Тип возвращаемого значения** – это тип результата, возвращаемого из функции. Типом возвращаемого значения может быть любой тип данных, кроме массива или функции, но может быть указатель на массив или указатель на функцию. Функция не может вернуть адрес локальной переменной.  Если тип возвращаемого значения void, то это означает, что

функция не возвращает никакого значения, или напрямую не возвращает никакого значения.

**Имя функции** – это любой правильно написанный идентификатор.

**Список параметров** – это список разделенных запятыми объявлений тех параметров, которые получает функция при вызове. Для каждого параметра, передаваемого в функцию, указывается его тип и имя. Если функция *не получает никаких значений*, список параметров задается как void, т.е. список параметров пустой.

**Тело функции** – это блок или составной оператор. В теле функции может быть оператор return, который возвращает полученное значение функции в точку вызова. Он имеет следующую форму: return выражение;

Любая функция должна быть **объявлена** (**прототип функции**), **вызвана** и **определена** (**описание функции**).

**Объявление (прототип) функции** задает имя функции, тип возвращаемого значения и список передаваемых параметров. Прототип функции указывает компилятору тип данных, возвращаемый функцией, количество и тип параметров, которые ожидает функция, а также порядок их следования. Также прототип функции сообщает компилятору о том, что далее в тексте программы будет приведено ее полное определение.

Прототип функции имеет следующий вид:

**ТипВозвращаемогоЗначения ИмяФункции**([**СписокПараметров**]);

*Пример прототипа функции fun, которая получает три целых числа и одно вещественное, а возвращает вещественное значение:*

**double** fun(**int**, **int**, **int**, **double**);

При **вызове** (активизации) **функции** указываются: *имя функции* и *фактические параметры* (т.е. значения которые передаются при вызове функции).

Существуют два способа передачи параметров в функцию: **по адресу (**с помощью указателя**)** и **по значению**.

**Определение функции** содержит, кроме объявления, тело функции, которое представляет собой последовательность описаний и операторов.

## РЕКУРСИВНЫЕ ФУНКЦИИ

**Рекурсивная функция** – это функция, которая вызывает сама себя – это **прямая рекурсия,** либо вызывает себя **косвенно** с помощью другой функции.

Рекурсивная функция умеет решать только простейшую часть задачи – так называемую **базовую задачу** (их может быть несколько). Если функция вызывается для решения **базовой задачи**, то она просто возвращает результат. Если функция вызывается для решения более сложной задачи, то задача делится на две части: на **базовую задачу** и **рекурсивный вызовов** (шаг рекурсии). **Шаг рекурсии** должен быть похож на исходную задачу, но по сравнению с ней должен быть проще и включать в себя ключевое слово **return**.

## ГЕНЕРАЦИЯ СЛУЧАЙНЫХ ЧИСЕЛ

Для работы с **генерацией случайных чисел** необходимо подключить следующие библиотеки: **stdlib.h** и **time.h**

В библиотеке **stdlib.h** находятся прототипы функции:

**int** rand(**void**);

Функция возвращает числа в диапазоне от 0 до 32 767.

**void** srand( **unsingned int**);

Функция получает в качестве аргумента число в пределах от 0 до 65535, если **int** занимает 2 байта и от 0 до 4 294 967 295, если 4 байта.

Функция **srand** устанавливает начальное псевдослучайное число. Функцию **srand** рекомендуется использовать следующим образом:

**srand( time(NULL));**

где в качестве параметра используется вызов функции **time**, которая находится в библиотеке **time.h** и возвращает текущую дату в виде целого числа.

Получение случайного числа из интервала [a , b]

**int** n;

n=a + rand() % (b – a + 1);

## ПРИМЕРЫ РЕШЕНИЙ

1. **Ввести 5 чисел x1,x2,x3,x4,x5. Определить максимальное из x1,x2,x3; из x4,x2,x3; из x4,x5,x3. Написать функцию, которая получает 3 числа и возвращает максимальное из этих чисел.**

```
#include<stdio.h>
//прототип функции
int max_3(int, int, int);
void main(){
int x1,x2,x3,x4,x5,max1,max2,max3;
printf("5 chisel:");
scanf("%d%d%d%d%d", &x1,&x2,&x3,&x4,&x5);

//вызов функции
max1=max_3(x1,x2,x3);
printf("Iz chisel:%d%d%d max=%d\n",x1,x2,x3,max1);

//вызов функции
max2=max_3(x4,x2,x3);
printf("Iz chisel:%d%d%d max=%d\n",x4,x2,x3,max2);
//вызов функции
max3=max_3(x3,x4,x5);
printf("Iz chisel:%d%d%d max=%d\n",x4,x5,x3,max3);
}
//описание функции
int max_3(int a,int b,int c){
     int max=a;
     if(max<b) max=b;
     if(max<c) max=c;
     return max;
}
```

2. **Заданы координаты сторон треугольника, найти его площадь.**

```
#include <stdio.h>
#include <math.h>
//прототип функции line, которая получает
//координаты точек и возвращает длину отрезка
double line(double ,double ,double ,double );
//прототип функции square, которая получает стороны
//треугольника и возвращает площадь треугольника
double square(double , double , double );

void main()
{
// координаты
double x1=2.0, y1=3.0, x2=4.0, y2=6.0, x3=7.0, y3=9.0;

double line1, line2, line3;

line1 = line(x1,y1,x2,y2); // вызов функции line
line2 = line(x1,y1,x3,y3); // вызов функции line
line3 = line(x2,y2,x3,y3); // вызов функции line

printf("S= %.2lf\n",square(line1,line2,line3));
}

//определение или описание функции line
double line(double x1,double y1,double x2,double y2)
{    return sqrt(pow(x1-x2,2)+pow(y1-y2,2));  }

//определение или описание функции square
double square(double a, double b, double c){
```

```
    double s, p=(a+b+c)/2;
  // формула Герона
    return s=sqrt(p*(p-a)*(p-b)*(p-c));
}
```

3. **Функция, которая получает два целых числа и возвращает сумму, произведение, разность и деление этих чисел.**

```
#include <stdio.h>
```

//прототип функции, которая получает два целых числа по значению, и три указателя на //целый тип и один указатель на вещественный тип

```
void fff(int, int, int*, int*, int*, double*);
void main(){
    int x = 5 , y = 2, sum, pr, raz;
    double del;
    fff( x, y, &sum, &pr, &raz, &del);
    printf("%d + %d=%d\n", x, y, sum);
    printf("%d -%d=%d\n", x, y, raz);
    printf("%d * %d=%d\n", x, y, pr);
    printf("%d / %d=%.2lf\n", x, y, del);
}
void fff(int a, int b, int*p1, int*p2, int*p3,double*p4){
    *p1=a+b;
    *p2=a*b;
    *p3=a-b;
    *p4=(double)a/b;
}
```

4. **Написать рекурсивную функцию подсчета факториала числа.**

```
#include<stdio.h>
int fact (int );
void main(){
    int n;
```

```
    printf("n= ");
    scanf("%d", &n);
    printf("%d!=%d\n", n, fact(n));
}
int fact (int n){
    return (n>1) ? n * fact(n-1) : 1;
}
```

## ЗАДАЧИ ДЛЯ ВЫПОЛНЕНИЯ:

1. Ввести 5 чисел x1, x2, x3, x4, x5 и определить минимальное и максимальное из x1, x2, x3; из x4, x2, x3; из x4, x5, x3. Использовать 2 функции, 1-я получает 3 числа и возвращает минимальное из них, 2-я получает 3 числа и возвращает максимальное из полученных чисел.

2. Написать функцию, которая считает факториал числа. Ввести 3 числа, посчитать сумму факториалов этих чисел.

**3.** Решение квадратного уравнения, используя 3 функции: 1-я выводит решаемое уравнение, 2-я считает дискриминант, 3-я выводит корни уравнения.

4. Функция, которая получает степень n и основание x и возвращает $x^n$.

**5.** Функция получает 2 целых числа и возвращает деление этих чисел.

**6.** Функция, которая получает число и определяет, является ли данное число простым или нет. Вывести все простые числа от 1 до 1000.

7. Функция, которая получает число и определяет, является ли данное число совершенным или нет. Вывести все совершенные числа от 1 до 1000. Напечатайте все сомножители каждого совершенного числа, чтобы убедиться, что число действительно совершенно.

**8.** Написать функцию, которая получает целое число и возвращает число с обратным порядком чисел. Например, получает число 3456, а возвращает 6543.

9. В функции **main()** определить 2 переменные. Написать функцию ввода, в которой вводятся данные в эти переменные.

**10.** Функция, которая получает только 3 значения из **main()** и возвращает в первую переменную выражение a+b+c –(a*a); во вторую a*a +b*b-c*c, а в третью a*b*c, т.е если начальные значения a=3, b=2, c=1, то теперь в **a** хранится -3, в **b** хранится 12, а в **c** – 6.

**11.** Функция получает два числа и меняет местами значения полученных переменных.

**12.** Написать рекурсивную функцию нахождения числа Фибоначчи.

**13.** Написать рекурсивную функцию подсчета сумму цифр числа. Прототип функции: **int** sum_cifr(**int** );

**14.** Наибольший общий делитель (НОД) двух целых чисел **x** и **y** – это наибольшее целое, на которое без остатка делится каждое из этих чисел. Написать рекурсивную функцию **nod**, который возвращает наибольший общий делитель полученных чисел. НОД для **x** и **y** рекурсивно определяется следующим образом: если **y** равен 0, то функция возвращает **x**, в противном случае **nod(x,y)** равняется **nod(y, x % y).**

**15.** Программа выбирает число в интервале [7, 57], необходимо угадать выбранное число. Причем программа помогает угадать число следующим образом:

**Введите число из интервала [7,57]=**

если ввели число больше, чем она выбрала, то выводит

**Введите число из интервала [7, введенное число]=**

если ввели число меньше, то

**Введите число из интервала [введенное число, 57]=**

**16**. Программа, которая помогает выучить таблицу умножения. Программа печатает вопрос типа: Сколько будет 6 * 7?

Затем вводится ответ. Если ответ правильный, то программа печатает: “МОЛОДЕЦ!!!!”, если же неверный, то печатает: “Попробуй еще раз” и дает возможность ввода нового числа. Если пользователь три раза отвечает неверно, то выводит таблицу умножения и дается возможность ввода нового числа до тех пор, пока не получит правильный ответ.