ЛАБОРАТОРНАЯ РАБОТА № 22

ФУНКЦИИ ПОЛЬЗОВАТЕЛЯ

Цель работы: получение навыков в написании программ с использованием функций; изучение механизма передачи параметров.

Краткие теоретические сведения

Функция – это вспомогательная программа (подпрограмма), предназначенная для получения некоторого объекта-результата (например, числа). Функция имеет следующий вид:

```
Тип_результата   Имя_функции ([Список_параметров])
        {
          Тело_функции
        }
```

Тип_результата – это тип возвращаемого из функции результата. Типом возвращаемого значения может быть любой тип данных, кроме массива или функции, но может быть указатель на массив или указатель на функцию. Если тип возвращаемого значения **void**, то это означает, что функция 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:

Тип_результата Имя_функции ([Список_параметров]);

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.

Пример. Написать программу, которая вводит целое число и определяет сумму его цифр. Использовать функцию, вычисляющую сумму цифр числа.

Для того чтобы найти последнюю цифру числа, надо взять остаток от его деления на 10. Затем делим число на 10, отбрасывая его последнюю цифру, и т.д. Сложив все эти остатки-цифры, мы получим сумму цифр числа.

```
#include <stdio.h>
#include <conio.h>
int SumDigits (int N)      // заголовок функции
{                          // начало функции
  int  d, sum = 0;         // тело функции
  while (N != 0 )
```

```
      {
         d = N % 10;
         sum = sum + d;
         N = N / 10;
       }
  return sum ;    // возврат суммы
}
void main()
{
  int N, s ;
  puts( “Введите целое число “);
  scanf (“%d”, &N);
  s = SumDigits (N)                    // Вызов функции
  printf (“ Сумма цифр числа %d = %d\n”, N, s );
  getch();
}
```

Логические функции

Очень часто надо составить функцию, которая просто решает какой-то вопрос и отвечает на вопрос «Да» или «Нет». Такие функции называются *логическими.* В языке Си ноль означает ложное условие, а единица – истинное.

Логическая функция – это функция, возвращающая **1** (если ответ «Да») или **0** (если ответ «Нет»). Логические функции используются, главным образом, в двух случаях:

• если надо проанализировать ситуацию и ответить на вопрос, от которого зависят дальнейшие действия программы;

• если надо выполнить какие-то сложные операции и определить, была ли при этом какая-то ошибка.

Пример. Ввести число ***N*** и определить, простое оно или нет. Использовать функцию, которая отвечает на этот вопрос.

Теперь расположим тело функции ниже основной программы. Чтобы транслятор знал об этой функции во время обработки основной программы, надо объявить её заранее.

```
#include <stdio.h>
#include <conio.h>
int Prime (int N);       // объявление функции
```

```
void main()
{
  int N;
  puts( “Введите целое число “);
  scanf (“%d”, &N);
  if ( Prime(N)  )                // Вызов функции
      printf (“ Число %d – простое\n”, N );
  else  printf (“ Число %d – составное\n”, N );
  getch();
}
int Prime (int N)     // описание функции
{
  for ( int i=2; i*i <=N; i++ )
     if ( N % i = = 0) return 0 ;   // нашли делитель – составное
  return 1 ;    // не нашли ни одного делителя – простое
}
```

Функции, возвращающие два значения

По определению функция может вернуть только одно значение-результат. Если надо вернуть два и больше результатов, приходится использовать специальный прием — **передачу параметров по ссылке**.

Пример. Написать функцию, которая определяет максимальное и минимальное из двух целых чисел.

В следующей программе используется достаточно хитрый прием: мы сделаем так, чтобы функция изменяла значение переменной, которая принадлежит основной программе. Один результат (минимальное из двух чисел) функция вернет как обычно, а второй – за счет изменения переменной, которая передана из основной программы.

```
#include <stdio.h>
#include <conio.h>
int MinMax (int a, int b, int &Max)
{
  if  (a>b)  { Max = a; return b; }
  else  { Max = b; return a; }
}
```

```
void main()
{
  int N, M, min, max ;
  puts( “Введите 2 целых числа “);
  scanf (“%d%d”, &N, &M );
  min = MinMax (N, M, max);              // Вызов функции
  printf (“min= %d, max = %d\n”, min, max );
  getch();
}
```

Глобальные и локальные переменные

Глобальные переменные — это переменные, объявленные вне основной программы и подпрограмм.

Глобальные переменные доступны из любой функции. Поэтому их надо объявлять вне всех подпрограмм. Остальные переменные, объявленные в функциях, называются *локальными* (местными), поскольку они известны только той подпрограмме, где они объявлены.

• Если в подпрограмме объявлена локальная переменная с таким же именем, как и глобальная переменная, то используется **локальная** переменная.

• Если имена глобальной и локальной переменных совпадают, то для обращения к глобальной переменной в подпрограмме перед ее именем ставится два двоеточия:

```
:: var = ::var * 2  +var;
```

Везде, где можно, надо передавать данные в функции через их параметры. Если же надо, чтобы подпрограмма меняла значения переменных, надо передавать параметр по ссылке.

Порядок выполнения работы

1. Изучить теоретические сведения.
2. Ответить на контрольные вопросы.
3. Выполнить задание.

Контрольные вопросы

1. Что такое функция?
2. Что включает в себя заголовок функции?

3. Какая существует связь между формальными и фактическими параметрами?

4. Как вы понимаете глобальные и локальные данные?

Задания для выполнения

1. На отрезке [2,n] найти все натуральные числа, сумма цифр которых при умножении числа на **a** не изменяется. Сумму цифр числа вычислять в функции.

2. Ввести натуральное число **n**. Найти и вывести все числа в интервале от 1 до **n-1**, у которых сумма всех цифр совпадает с суммой цифр числа **n**. Если таких чисел нет, то вывести сообщение. Сумму цифр числа вычислять в функции.

3. Ввести координаты **n** точек. Вычислить количество точек, попадающих в кольцо с внутренним радиусом **R1** и внешним **R2** (R1<R2) и координаты точек, не принадлежащих этому кольцу. Проверку на принадлежность точки кольцу определять в функции.

4. Вычислить сумму факториалов всех нечетных чисел от **m** до **n.** Вычисление факториала оформить в функции.

5. Ввести **n** целых чисел**.** Вычислить сумму тех из них, которые содержат только четные цифры. Определить также, сколько нечетных цифр в найденной сумме. Поиск четных и нечетных цифр осуществлять в одной функции.

6. Ввести **n** целых чисел. Для каждого из них найти и вывести наибольшее число **m** (m>1), на которое сумма цифр числа делится без остатка. Если такого числа нет, то выводить слово “нет”. Сумму цифр числа вычислять в функции.

7. Ввести координаты **n** точек. Вывести количество точек, не попадающих в круг радиуса **r** с центром в точке $O(x_0, y_0)$, и координаты точек, лежащих в этом круге. Расстояние от точки до центра круга вычислять в функции.

8. Ввести **n** целых чисел. Вывести наибольшую и наименьшую цифры в записи каждого из этих чисел, используя одну подпрограмму для их поиска .

9. Ввести **n** натуральных чисел. Вывести значения тех из них, которые делятся на каждую из своих цифр, используя подпрограмму. Предусмотреть случай, что таких чисел нет.

10. Ввести **n** троек вещественных чисел **a, b, c**. Вычислить среднее арифметическое значение максимальных элементов каждой тройки. Поиск максимального значения осуществлять в функции.

11. Натуральное число, в записи которого **n** цифр, называется числом Армстронга, если сумма его цифр, возведенная в степень **n**, равна самому числу. Вывести все числа Армстронга от 1 до K или сообщение об их отсутствии. Сумму цифр числа вычислять в функции.

12. Ввести натуральное число **n.** Вычислить **y=1!+2!+3!+...+n! (n>1).** Вычисление факториала оформить в функции.

13. Ввести **n** троек целых чисел. Вывести номер первого четного числа в каждой тройке, осуществляя его поиск в функции. Если четного числа в тройке нет, то считать его номер равным нулю.

14. Ввести координаты **n** точек. Подсчитать число точек, находящихся внутри круга радиусом **R** с центром в начале координат. Расстояние точки от начала координат вычислять в функции.

15. Вычислить суммы факториалов всех нечетных чисел от 1 до 9. Вычисление факториала осуществлять в функции.
