Функции в С

Тема 1.2

Функции — это отдельные независимые блоки кода, представляющие собой именованные последовательности описаний и операторов, выполняющих ряд предопределенных действий.

Классификация функций

- Системные функции хранятся в стандартных библиотеках. Примером системных функций являются функции `printf()` и `scanf()`.
- Собственные функции - это функции, написанные пользователем для решения конкретной подзадачи.

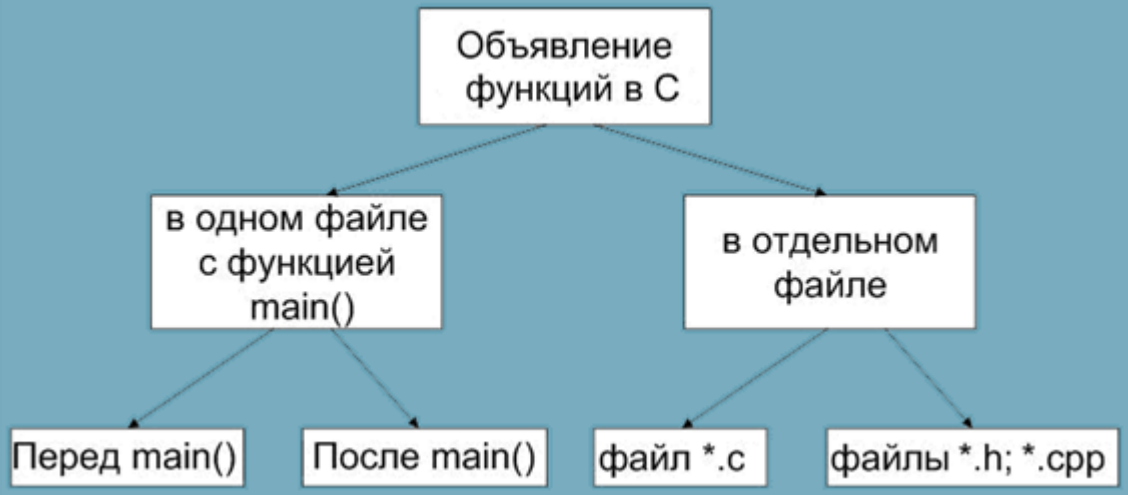
Функция должна быть объявлена и определена.

Объявление функции (прототип, заголовок) задает имя, тип значения и список формальных параметров.

Определение содержит кроме объявления переменные и операторы, называемые телом функции, и определяющие действие функции.

Объявление функции

Способы объявления(определения) функций



Объявление функции

Объявление функции: прототип функции

```
returnDataType functionName( dataType argName1, dataType argName2, ..., dataType argNameN);
```

где,

- `returnDataType` — возвращаемый тип данных
- `functionName` — имя функции
- `dataType` — тип данных
- `argName1` — имена параметров функции (количество параметров неограниченно)

Пример объявления функции:

```
/ / Объявление прототипа функции с двумя целыми параметрами
/ / функция принимает два аргумента и возвращает их сумму
int sum(int num1, int num2);
```

или

```
int sum(int , int ); // тот же прототип функции
```

Определение функции

Определение функции задает тип возвращаемого значения, имя функции, типы и число формальных параметров, а также объявления переменных и операторы, называемые телом функции, и определяющие действие функции

Синтаксис определения функции:

```
returnDataType functionName( dataType argName1, dataType argName2, ..., dataType argNameN)
{
    // тело функции
}
```

Пример определения функции:

```
int sum(int num1, int num2)
{
    return (num1 + num2);
}
```

Определение функции

Для того, чтобы компилятор мог осуществить проверку соответствия типов передаваемых фактических параметров типам формальных параметров до вызова функции нужно поместить объявление (прототип) функции. Если функция ранее была объявлена, она должна быть определена с тем же возвращаемым значением и типами данных. Имена параметров функции могут не быть одинаковыми.

Пример :

```
1  // объявление функции суммирования
2  int sum(int, int);
3
4  // определение функции суммирования
5  int sum(int num1, int num2)
6  {
7      return (num1 + num2);
8  }
```

Вызов функции

Вызов функции выполняется следующим образом:

```
1   funcName( arg1, arg2, ...  );
```

где,

- `funcName` — **имя функции**
- `arg1` — **аргументы функции (фактические параметры)**

Пример объявления и вызова функции

```
1   // объявление функции нахождения n!
2   void faktorial(int numb)// заголовок функции
3   {
4       int rezult = 1;
5       for (int i = 1; i <= numb; i++)
6           rezult *= i;
7       cout << numb << "! = " << rezult << endl;
8   }
9
10  int main(int argc, char* argv[])
11  {
12      int digit;
13      cout << "Enter number: ";
14      cin >> digit;
15      faktorial(digit);//вызов функции
16      system("pause");
17      return 0;
18  }
```

Библиотечные функций и их подключение

Прототипы библиотечных функций находятся в специальных заголовочных файлах, поставляемых вместе с библиотеками в составе систем программирования, и включаются в программу с помощью директивы `#include`. Рассмотрим пример, в котором воспользуемся функцией, которая возводит некоторое число в степень. Для этого нужно подключить заголовочный файл `<cmath>` и запустить функцию `pow()` в теле программы.

Пример:

```
1  //подключаем заголовочный файл <cmath>
2  #include <cmath>
3
4  int main(int argc, char* argv[])
5  {
6      float power = pow(3.14,2); //запуск функции
7      return 0;
8  }
```

Рекурсия функций

Рекурсия – определение части функции через саму себя, то есть это функция, которая вызывает саму себя, непосредственно (в своём теле) или косвенно (через другую функцию).

```
3  #include "stdafx.h"
4  #include <iostream>
5  using namespace std;
6
7  unsigned long int factorial(unsigned long int);// прототип рекурсивной функции
8  int i = 1; // инициализация глобальной переменной для подсчёта кол-ва рекурсивных вызовов
9  unsigned long int result; // глобальная переменная для хранения возвращаемого результата
10
11 int main(int argc, char* argv[])
12 {
13        int n; // локальная переменная для передачи введенного числа с клавиатуры
14        cout << "Enter n!: ";
15        cin >> n;
16        cout << n << "!" << "=" << factorial(n) << endl; // вызов рекурсивной функции
17        system("pause");
18        return 0;
19 }
20
21 unsigned long int factorial(unsigned long int f) // рекурсивная функция для нахождения n!
22 {
23        if (f == 1 || f == 0) // базовое или частное решение
24              return 1; // все мы знаем, что 1!=1 и 0!=1
25        cout << "Step\t" << i << endl;
26        i++; // операция инкремента шага рекурсивных вызовов
27        cout << "Result= " << result << endl;
28        result = f * factorial(f - 1); // функция вызывает саму себя, причём её аргумент уже на 1 меньш
29        return result;
30 }
```