

ФУНКЦИИ

C \ C++

Подпрограмма – группа операторов реализующая законченный алгоритм и оформленная как самостоятельная синтаксическая единица вызываемая по имени.

Модульное программирование – разбиение больших сложных алгоритмов на отдельные подзадачи и реализация подзадач (в том числе, иерархических) в виде подпрограмм.

Подпрограммы могут быть реализованы в виде **Процедур** и в виде **Функций**.

Процедура и **Функция** – независимые части программы, имеющие имя и реализующие определенный алгоритм.

Отличие Процедуры от Функции состоит в том, что Процедура может вычислять и возвращать в основную программу векторный результат (несколько переменных), а Функция только скалярный, при этом вычисленное значение присваивается имени Функции.

В языке C Процедур нет – есть только Функции.

ФУНКЦИИ

C / C++

Функции

Формат описания Функции:

```
[класс] <возвращаемый_тип> <имя_функции> ([<тип1>  
<имя_формального_параметра1>, ..., <типN>  
<имя_формального_параметраN>]) [throw (исключения)]  
{  
<тело_функции >  
  return <возвращаемое_значение>;  
}
```

*где - класс – extern или static – явно задает область видимости функции:
глобальная (умолчание) или в пределах модуля;
- исключения – обрабатываемые функцией исключения.*

Пример вызова Функции:

C - `y = cube (a);`

ФУНКЦИИ

C / C++

Замечания

- Количество и тип фактических параметров, передаваемых в Функцию при вызове должны точно соответствовать объявленному количеству и типам формальных параметров,
- Имя Функции обычно используется в качестве операнда в выражениях,
- При возврате из Функции в вызывающую программу управление передается оператору следующему за оператором вызова процедуры,
- Все переменные объявленные внутри Функции являются локальными.
- Для каждого параметра, передаваемого в функцию указывается его тип и имя (в описании Функции имена можно опускать,
- Тип возвращаемого Функцией значения может быть любым, кроме массива и функции (но может быть указателем на массив или функцию,
- Если Функция не должна возвращать значения указывается тип `void`, но тогда она не может входить в выражения.

ФУНКЦИИ

C / C++

Примеры функции

```
#include <STDIO.H>
int max (int a, int b)
{ /* Функция возвращает максимальное из двух чисел */
  if (a > b)
    return(a);
  else
    return (b);
}
main ()
{
  int a, b;
  printf ("Введите два целые числа -> ");
  scanf ("%d %d", &a, &b);
  printf ("Максимальное значение -> %d\n", max(a, b));
  return 0;
}
```

ФУНКЦИИ

C \ C++

Практическое занятие

Написать программу, использующую три последовательно выполняемые функции:

- F1 вычисляет произведение 3-х чисел,
- F2 – вычисляет корень квадратный из F1,
- F3 – выводит на печать результат F2.

```

#include <conio.h>
#include <stdio.h>
#include <math.h>
float F1 (float a, float b, float c)           // int F1(int a, int b, int c)
// считает произведение трех вещественных чисел
{ float k;   k=a*b*c; return (k);   }       // { return (a*b*c); }
float F2 (float d)
// вычисляет корень квадратный числа
{ float j;   j = pow(d,0.5); return (j); }   // { return (pow(d,0.5)); }
void F3 (float e)
// выводит на экран вещественное число
{ printf ("\nчисло -> %6.3f\n", e); }
main ()
{
    float a,b,c; clrscr ();
    printf ("Введите через пробел 3-и вещественных числа и нажмите
            Enter\n");
    scanf ("%d %d %d", &a, &b, &c);
    F3(F2(F1(a,b,c)));
    getch (); return 0;
}

```

Библиотека C – это подключаемая к головной программе библиотека ресурсов в виде одного или нескольких **Заголовочных файлов**. Это файлы с расширением **.h**, которые включаются в программу с помощью **директивы препроцессора #include**. Заголовочные файлы представляют собой файлы в формате ASCII.

В заголовочном файле могут содержаться:

- ❖ Определения типов - `struct point { int x, y; }`
- ❖ Описания функций - `extern int strlen(const char*);`
- ❖ Определения inline-функций - `inline char get() { return *p++; }`
- ❖ Описания данных - `extern int a;`
- ❖ Определения констант - `const float pi = 3.141593`
- ❖ Перечисления - `enum bool { false, true };`
- ❖ Другие директивы include - `#include`
- ❖ Определения макросов - `#define Case break; case`
- ❖ Комментарии - `/* проверка на конец файла */`
- ❖ и др. элементы программ на C.

Директива `#include` включает в программу содержимое указанного файла. Имя файла может быть указано двумя способами:

`#include <some_file.h>`

`#include "my_file.h"`

Если имя файла заключено в **угловые скобки (<>)**, то это означает, что подключается стандартный заголовочный файл, и компилятор ищет этот файл в заданных в настройках местах.

Двойные кавычки (") означают, что заголовочный файл – пользовательский, и компилятор ищет его в том каталоге, где находится исходный текст программы.

Заголовочный файл также может содержать **вложенные директивы #include**.

Файл LibEx.h

```
/* Простой пример пользовательского
заголовочного файла C из 3-х функций
*/
```

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

```
int F1 (int a, int b, int c)
```

```
// считает произведение трех целых чисел
```

```
{
    return (a*b*c);
}
```

```
float F2 (int d)
```

```
// вычисляет корень квадратный числа
```

```
{
    return (pow(d,0.5));
}
```

```
void F3 (float e)
```

```
// выводит на экран вещественное число
```

```
{
    printf ("\nчисло -> %6.3f\n", e);
}
```

```
/* Программа последовательно вычисляет
произведение 3-х целых чисел, затем корень
квадратный из этого произведения и затем
выводит на печать результат */
```

```
#include <conio.h>
```

```
#include "LibEx.h"
```

```
main ()
```

```
{
    int a,b,c; clrscr ();
    printf ("Введите через пробел 3-и
вещественных числа и нажмите Enter\n");
    scanf ("%d %d %d", &a, &b, &c);

    F3(F2(F1(a,b,c)));
    getch ();
    return 0;
}
```