Функции (введение)

Подпрограммы в Си

В языке Си подпрограммы представлены только функциями.

```
// заголовок функции
тип-результата имя-функции(список формальных параметров
                                                с их типами)
// тело функции
{
    определения
    операторы
}
```

```
float avg(float a, float b)
{
    float c;
    c = a + b;
    return c / 2.0;
}
```

Тип возвращаемого значения

- Функция может вернуть значение любого типа кроме массива.
- Если функция ничего не возвращает, то в качестве типа возвращаемого значения следует указать void.
- Если тип возвращаемого значения не указан, то согласно стандарту C89, компилятор предполагает, что возвращается значение целого типа.
- Согласно стандарту C99 тип возвращаемого значения опускать нельзя (warning).

Параметры функции

- Любая функция может принимать параметры.
- Если список параметров содержит только ключевое слово void, у функции нет параметров.
- Параметры функции перечисляются через запятую.
- Определение параметра начинается с указания его типа, за которым следует имя параметра. При этом для каждого имени тип указывается отдельно!

Примеры заголовков функций

```
// возвращает максимальное из
// двух чисел
int max(int a, int b)
{
...
```

```
// только c89!
max(int a, int b)
{
...
```

```
// ошибка компиляции
int max(int a, b)
{
...
```

```
// выводит значение
// положительного числа
void print_pos_num(int num)
{
...
```

```
// выдает звуковой сигнал
void beep(void)
{
...
```

```
// возвращает число секунд,
// оставшееся в текущих сутках
int get_rest_second(void)
{
...
```

Тело функции

- У каждой функции есть исполнимая часть, которая называется *телом функции* и заключена в фигурные скобки (которые также являются частью тела функции).
- Тело функции может содержать как объявления переменных, так и операторы.
- Переменные, описанные в теле функции, «принадлежат» только этой функции и не могут быть ни прочитаны, ни изменены другой функцией.
- Тело функции не может содержать в себе определения других функций.
- Если функция ничего не возвращает, ее тело может быть пустым.

Тело функции

```
#include <stdio.h>
void f1(void)
{
    int a = 5;
    printf("%d\n", a);
}
```

```
void f2(void)
{
}
```

```
void f3(void)
{
   // ошибка компиляции
   void g3(void)
   {
   }
}
```

```
#include <stdio.h>

void f4(void)
{
   int a = 5;
   printf("%d\n", a);
}

void g4(void)
{
   // ошибка компиляции
   printf("%d\n", a);
}
```

Оператор return

return *выражение*;

- Завершает выполнение функции и возвращает управление вызывающей стороне.
- Используется для возврата значения (если функция возвращает результат).

- Функция может содержать произвольное число операторов return.
- Оператор return может использоваться в функциях типа void. При этом никакое выражение не указывается.

Оператор return

```
int max(int a, int b)
{
    if (a < b)
        return b;

    return a;
}
```

```
void beep(void)
{
    printf("\a");

    // обычно не пишут
    return;
}
```

```
void print_pos_num(int a)
{
    if (a < 0)
        return;

    printf("%d\n", a);
}
```

Вызов функции

Для вызова функции необходимо указать ее имя, за которым в круглых скобках через запятую перечислить аргументы.

```
float a = avg(2.0, 5.0);
```

Если функция возвращает значение, ее можно использовать в выражениях.

```
float a, b;

printf("Enter a and b:");
scanf("%f%f", &a, &b);

if (avg(a, b) < 0.0)
    printf("Averadge is negative!\n");
```

Вызов функции

Указывать скобки при вызове функции необходимо, даже если у этой функции нет параметров.

```
beep();

// Функция НЕ будет вызвана
beep;   // warning: statement with no effect
```

Вызов функции

Значение, возвращаемое функцией, может быть проигнорировано.

```
#include <stdio.h>

int main(void)
{
    int n_chars;

    n_chars = printf("Hello, world!\n");
    // после вызова printf n_chars равно 14
    printf("n_chars = %d\n", n_chars);

    (void) printf("Hello, world!\n");
    // явно указано, что возвращаемое значение не используется

    return 0;
}
```

Вызов функции

Операция	Название	Нотация	Класс	Приоритет	Ассоциат.
`( . . . )`	Вызов функции	`X(Y)`	Постфиксная	16	Слева направо

Объявление функции

```
#include <stdio.h>

int main(void)
{
    // error: implicit declaration of function 'avg'
    float a = avg(2.0, 3.0);

    printf("%f\n", a);

    return 0;
}

float avg(float a, float b)
{
    return (a + b) / 2.0;
}
```

Объявление функции

Объявление функции предоставляет компилятору всю информацию, необходимую для вызова функции: количество и типы параметров, их последовательность, тип возвращаемого значения.

Объявление функции состоит из заголовка функции

```
тип-результата имя-функции(список формальных параметров
                                                с их типами);
```

Объявление функции должно соответствовать ее определению.

Объявление функции может не содержать имен параметров. Однако их обычно оставляют для большей наглядности.

Объявление функции

```
#include <stdio.h>

float avg(float a, float b);    // float avg(float, float);

int main(void)
{
    float a = avg(2.0, 3.0);

    printf("%f\n", a);

    return 0;
}

float avg(float a, float b)
{
    return (a + b) / 2.0;
}
```

Функции без параметров

```
#include <stdio.h>

void f()
{
    printf("f\n");
}

void g(void)
{
    printf("g\n");
}
```

```
int main(void)
{
    // ошибка компиляции?
    f();

    // ошибка компиляции?
    g();

    // ошибка компиляции?
    f(1, 2, 3);

    // ошибка компиляции?
    g(1, 2, 3);

    return 0;
}
```

Функции без параметров

Объявление

```
void f(void);
```

означает, что у функции нет ни одного параметра.

Объявление

```
void f();
```

означает, что у функции могут быть, а могут и не быть параметры. Если параметры есть, мы не знаем ни их количество, ни их тип.

Аргументы функции

В Си все аргументы функции передаются «по значению».

Авторы языка: «Благодаря этому свойству обычно удается написать более компактную программу, содержащую меньшее число посторонних переменных, поскольку параметры можно рассматривать как должным образом инициализированные локальные переменные.»

Аргументы функции

```
#include <stdio.h>

int power(int base, int n)
{
    int res = 1;

    while (n > 0)
    {
        res = res * base;
        n = n - 1;
    }
    // n = 0

    return res;
}
```

```
int main(void)
{
    int a, n = 5;

    printf("n = %d\n", n);
    // n = 5

    a = power(2, n);

    printf("%d^%d = %d\n",
                    2, n, a);
    // 2^5 = 32

    return 0;
}
```

Аргументы функции

Из-за такого способа передачи параметров возникают трудности при реализации функций, которые должны возвратить несколько параметров одновременно.

```
#include <stdio.h>

void decompose(
        float f,
          int int_part,
             float frac_part)
{
    int_part  = f;
    frac_part = f - int_part;
}
```

```
int main(void)
{
    int i;
    float f;

    decompose(3.14159, i, f);

    printf("%d %f\n", i, f);

    return 0;
}
```