acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . CoffeeFunction :

Функция - что такое и зачем нужно?

Зачем нужны функции?

Здесь нужно написать пример, когда код повторяется с разными вариантами.

Функция

Часть программы можно описать отдельно, дать этой части собственное имя и исполнять как отдельную инструкцию.

Таким образом мы получим **функцию** .

Каждая функция имеет:

- собственное имя,
- набор переменных, которые функции необходимо знать для вычислений,
- значение, которое вычисляет функция в процессе своей работы,
- набор инструкций, которые выполняет функция

Чтобы определить функцию, надо написать:

тип_возвращаемого_значения **имя_функции** (список параметров через запятую) {
    декларации локальных переменных
    инструкции
}

Самая простая функция (параметров нет, возвращаемого значения нет)

Дадим коду, который печатает Hello! имя hi. Напишем функцию hi() и будем дальше исполнять ее как отдельную инструкцию.

```
#include <stdio.h>

// объявляем функцию
// имя функции hi, переменных нет, значений не вычисляет (пишем void)
void hi()
{                       // в { } пишем инструкции функции
    printf("Hello!");   // код функции пишем с отступами
}                       // тут функция закончилась

int main()       // выполнение программы начинается с main
{
                 // ВЫЗОВ функции, с этого места начинаются выполняться инструкции  функции
    hi();
                 // дальше опять будут выполняться инструкции main
    hi();
    hi();

    return 0;
}
```

Этот код напечатает:

```
Hello!
Hello!
Hello!
```

Функция с параметром

Изменим функцию hi. Пусть она печатает Hello, *номер группы*. Для этого передадим номер группы функции в виде параметра.

```
#include <stdio.h>

void hi(int group) {                        // 1 раз определили функцию, передали параметр group, описали тип параметра
                                            // переменная int group уже описана как аргумент и является
                                            // внутренней (локальной) переменной.
                                            // имя group - это локальное имя, видимое только внутри функции
                                            // все переменные с таким же именем, описанные в другом месте
                                            // функция hi не видит
                                            // Во время передачи параметра будет передаваться только значение выражение при вызове

    printf("Hello, group %d!\n", group);    // использовали параметр (это переменная)
}

int main() {
    hi(716);   // Hello, 716!
    hi(778);   // Hello, 778!

    int a = 417;
    hi(a);     // Hello, 417! (При вызове функции СНАЧАЛА вычисляем значение параметра)
    hi(a - 2); // Hello, 415! (При вызове вычислили a-2, потом переменной group функции hi присвоили вычисленное значение 415

    return 0;
}
```


Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

Несколько параметров, возвращает значение

Надо писать тип каждого параметра, даже если типы одинаковые.

Напишем функции, которые вычисляют периметр и площадь прямоугольника. Эти функции принимают параметры (длины сторон) и возвращают посчитанное значение.

```
#include <stdio.h>
                                  // создали первую функцию perimetr, в нее передают два числа a и b
float perimetr(float a, float b)
{
   float res;
   res = (a+b)*2;
   return res;                    // возвращает число
}                                 // первая функция закончилась

float area(float a, float b) {    // создали другую функцию area, в нее передают два числа a и b
   return a*b;                    // можно сразу написать return выражение
                                  // без дополнительных переменных
}
int main() {
   float p, s;

   p = perimetr(3, 5);
   s = area(3, 5);
   printf("Периметр = %f\n", p); // напечатали p (Периметр = 16)
   printf("Площадь = %f\n", s);  // напечатали s (Площадь = 15)

   // везде, где можно написать число, можно написать вызов функции:
   printf("Периметр = %f\n", perimetr(3.3, 5));// Периметр = 16.6
   printf("Площадь = %f\n", area(3.3, 5));     // Площадь = 16.5

   return 0;
}
```

Функция может вернуть не больше 1 значения.

Функция с более формальной точки зрения (declaration, definition, call)

Функция - именованный кусок кода с параметрами и возвращаемым значением.

Нельзя определить две функции с одинаковыми именами.

Определение функции (definition)

Описываем имя функции, какие у нее аргументы (параметры), какой тип данных она возвращает и какие инструкции выполняет.

тип_возвращаемого_значения **имя_функции** (список параметров через запятую) {
    декларации локальных переменных
    инструкции
}

```
/* Начало определения функции с именем power, которая возводит основание a в целую положительную степень n */
long int power (int a, unsigned int n)
{
    long int res;
    unsigned int i;
    for (i=0, res=1; i<n; i++) {
        res = res * a;
    }
    return res;   // Возвращается из функции число, равное значению переменной res
}
/* конец определения функции power */
```

Реализация функции.

Прототип функции (declaration)

Прототип функции - это определение функции без описания ее тела. Вместо них ставится точка с запятой.

тип_возвращаемого_значения имя_функции (список параметров через запятую) **;**

```
long int power (int a, unsigned int n); // прототип функции power, возводящей a в n
```

Обещание компилятору, что когда-то мы определим функцию с таким именем, типом возвращаемого значения и этими параметрами.

Если обещание не выполнили, и нигде реализации нет, то linker выдаст ошибку.

Название параметров можно опустить и оставить только перечисление типов через запятую.

Некоторые компиляторы требуют void в параметрах, если их нет.

Вызов функции (call) - ее использование

```
int main()
{
    long y;
    y = power (2, 3);   // вызов (использование) функции power
```

```
    return 0;
}
```

Запоминаем

Eng	Rus	C	Аналогия
declaration	прототип	`int abs(int);`	мечтаем о самокате

definition	определение
	реализация
		`int abs(int x) {`
		`  if (x<0)`
		`  return -x;`
		`  return x;`
		`}`
			делаем самокат

call	вызов использование	`y = abs(x+3);`	катаемся на самокате

Не путайте функции с вводом данных с клавиатуры /выводом на экран.

Философские размышления после дополнительных вопросов.

- **scanf, getchar** и тп – считывают данные из входного потока программы (их вы обычно вводите с клавиатуры)
- **printf, putchar** и тп – печатают данные на выходной поток программы (обычно на экран)
- **аргументы функции** – их передают в функцию при вызове этой функции, не имеет ничего общего с процессом считывания введенного с клавиатуры. В задаче вообще может не быть ни одного введенного символа, но есть вызовы функций с аргументами.

sin(0) в данном случае 0 аргумент.

- **значение, возвращаемое функцией.** Обычно его куда-то сохраняют. Например, y=sin(0); возвращаемое функцией sin значение, сохранили в переменной y.

Значение из функции возвращают с помощью оператора **return.** Например, return 1; передаст управление из той функции, где мы сейчас находимся обратно в точку вызова и значение функции будет 1.

-- TatyanaDerbysheva - 20 Oct 2013