Передача аргументов в функцию

Как уже отмечалось выше, при вызове функции в неё передаются не сами переменные, а только их значения. Т.е. функция, что бы она там ни делала, не может никак повлиять на значения переменных, которые в неё передаются. Такой способ передачи значений в функцию называется передачей аргументов по значению.

Рассмотрим пример. Допустим, мы хотим написать программу, которая меняет местами значения переменных. Кажется, всё просто.

Листинг 1.

```
#include <stdio.h>

void swap(int a, int b){
  int temp;

  temp = a;
  a = b;
  b = temp;
}

int main(void) {
  int x = 4, y = 9;

  printf("x=%d y=%d\n",x,y);
  swap(x,y);
  printf("x=%d y=%d\n",x,y);

  return 0;
}
```

Вроде бы всё должно работать, но нет. Запустите программу и убедитесь в этом. И дело, как видите, не в функции, функция работает правильно, и значения переменных **a** и **b** действительно меняются, но к нашим исходным переменным это не имеет никакого отношения, т.к. они в функцию не передавались, а передавались только их значения.

Это никуда не годится. Чтобы с этим справиться, необходимо использовать другой способ передачи аргументов в функцию – передачу аргументов по ссылке. Для этого нам придётся воспользоваться указателями. Надеюсь вы хорошо с ними разобрались на прошлом шаге.

Напомню, что указатель – это такая переменная, в которой хранится адрес в памяти. В зависимости от типа указателя по этому адресу хранится значение соответствующего типа. Перепишем нашу программу используя передачу аргумента по ссылке.

Листинг 2.

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

```
// функция swap принимает два указателя на целые числа
// другими словами, два адреса в памяти компьютера,
// в которых записаны целые числа
void swap(int *a, int *b){
// a -- адрес первого числа
// b -- адрес второго числа

  int temp;
// *a -- разыменование указателя
// *a позволяет обращаться к значению, хранящемуся по адресу a
// выводим значения, хранящиеся по адресам a и b
  printf("a=%d b=%d\n",*a,*b);

// переменной temp присваиваем значение, хранящееся по адресу a
  temp = *a;
// в ячейку по адресу a записываем значение, хранящееся по адресу b
  *a = *b;
// в ячейку по адресу b записываем значение из переменной temp
  *b = temp;
  printf("a=%d b=%d\n",*a,*b);
}

int main(void) {
  int x = 4, y = 9;

  printf("x=%d y=%d\n",x,y);
// передаём адреса переменных x и y в фунцию swap
  swap(&x,&y);

  printf("x=%d y=%d\n",x,y);

  return 0;
}
```

Программа снабжена подробными комментариями, но я ещё раз поясню, что происходит. В начале программы объявляются переменные **x** и **y**. Им присваиваются значения **4** и **9** соответственно. Адреса этих переменных передаются в функцию **swap**. Обращаю особое внимание. Передаются адреса переменных, а не их значения. Внутри функции, используя дополнительную переменную, значения по переданным адресам меняются местами. Функция **swap** заканчивает свою работу.

Передача массивов в функцию

Массивы, как и простые переменные, можно передавать в функцию, но есть ряд особенностей, с которыми нам следует познакомиться. Продемонстрирую их на примере. В следующей программе я написал две функции, которые выводят на экран переданный им массив.

Листинг 3.

```
#include <stdio.h>

void print_arr(int arr[], int n){

  for (int k=0; k<n; k++){
    printf("arr[%d] = %d\t", k, arr[k]);
  }

  printf("\n");
}

void print_arr2(int arr[][5], int k, int n){

  for (int i=0; i<k; i++,printf("\n"))
    for(int j=0; j<n; j++)
      printf("arr[%d][%d] = %d\t", i, j, arr[i][j]);

  printf("\n");
}


int main(void){

  int q[5] = {5,4,2,3,4};
  int q2[2][5] = { {1,2,3,4,5}, {0,9,8,7,6} };

  print_arr(q,5);

  print_arr2(q2,2,5);

  return 0;
}
```

Первый момент. Если в функцию предполагается передавать массив, то необходимо об этом предупредить компилятор. Для этого в заголовке функции при её описании необходимо рядом с именем переменной-массива указать пару квадратных скобок **[]**. Посмотрите на объявление функции **print_arr**: из него сразу понятно, что переменная **arr** будет массивом, т.к. рядом с её именем указаны **[]**.

Листинг 4.

```
void print_arr(int arr[], int n){
  //..
}
```

Второй момент. Когда мы передаём массив в функцию, то функция ничего не знает о размере этого массива. Поэтому необходимо дополнительно передавать размер массива в функцию отдельным параметром.

Третий нюанс. Если в функцию передаётся двумерный массив, то кроме двух пар квадратных скобочек `[][]`, во второй из них необходимо явно указать её размерность. В примере выше это `5`.

Четвертый нюанс. Массивы всегда передаются в функцию по ссылке. Это означает, что любые изменения массива внутри функции отразятся на исходном массиве.

Есть и другие способы передать массив в функцию, они связаны с передачей указателей на начало массива, но о них мы не будем упоминать в рамках данного курса.

Практика

1. Решите предложенные задачи с автоматической проверкой решения.

Интернет версия: http://youngcoder.ru/lessons/10/argumenty_funkcii.php