Actual Parameters, Formal Parameters, Call by Value & Call by Reference

Dr. Rahul Das Gupta

What is **Actual Parameters**?

Actual parameters are specified in the **function call**.

What is **Formal Parameters**?

Formal parameters are specified in the **function declaration** and **function definition**.

Example:

```
#include<stdio.h>
int sum_of_digits(int );

void main()
{
  int n;
  printf("\n Enter the integer:");
  scanf("%d", &n);
  printf("\n Sum of the digits of %d is %d.", n, sum_of_digits(n));
  /* Here, n is the actual parameter in the function call sum_of_digits.*/
}

int sum_of_digits(int x)    /* x is the formal parameter.*/
{
   int S=0;
   while (x>0)
   {
     S=S+x%10;
     x=x/10;
   }
   return S;
}
```

Basics of Pointer

Case 1: Decleration Statement

Here, * is associated with primitive data type int.

```
int  *p;
int  *p,*q;
```

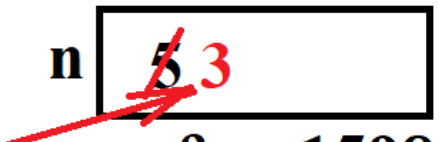
int * : Address of an integer

Case 2: Non-declerative Statement

Here, * is associated with variable.
Here, * is Content of Operaror

```
int  n=5;
int *p;

p=&n;
*p=3;
```



p= &n=1598

printf("\n n = %d", n); /*Output: n=3*/

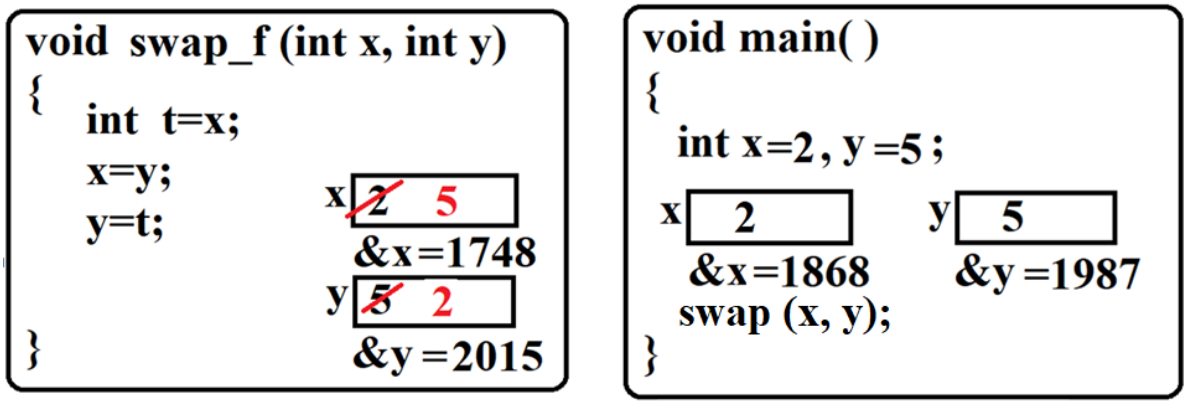
Call by Value & Call by Reference

Let us first try to analyse output of the following program:

```
#include<stdio.h>

void swap_f (int, int);   /*Fake Swap: Not effective */
void swap_a(int *, int *); /*Actual Swap: Effective */

void main()
{
   int x=2, y=5;
   printf("\n x=%d and y=%d", x, y);
   swap_f (x, y); /* Actual Parameters : x=2 and y = 5 */
   printf("\n x=%d and y=%d", x, y);
   /*
     &: Address of operator
     &x: address of x and &y: address of y
*/
   swap_a (&x, &y);
   printf("\n x=%d and y=%d", x, y);
}
```



But no change at the locations of x and y inside the main ( ) function. Hence, this swapping will not be effective

```
void swap_f (int x, int y)
/*
```

==========

Important point to be noted:

Address of these x and y are not same as of those x and y declared in the main.

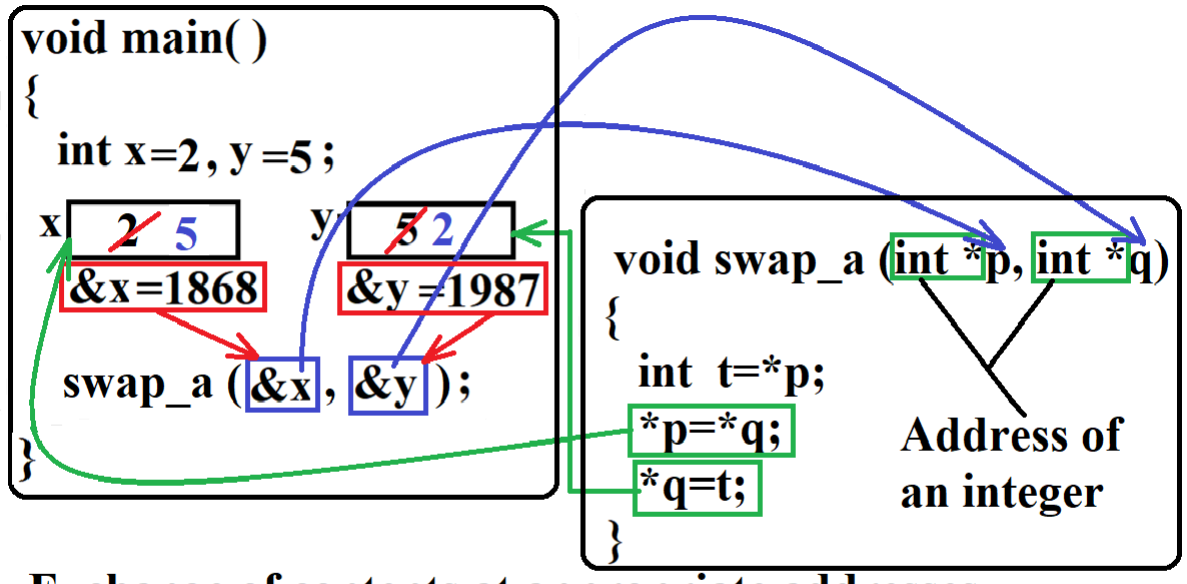
Therefore, these x and y are entirely different from those x and y declared in the main function.

Thus the value of x and y declared in the main function will remain unchanged after the execution of swap_f function.

==========

*/

```
{
   int t=x;
   x=y;
   y=t;
   /* ============================
    printf ("\n x=%d and y=%d", x, y);
   Output of the above printf statement:
     x=5 and y=2
    ============================*/
}
```



Exchange of contents at appropriate addresses.

Swapping will be effective.

```
void swap_a(int *p, int *q)
/*
```

==========

Here, p and q are both the pointers to integers (address of integers). Corresponding function call in main is as follows:

swap_a (&x, &y);

Finally, p and q are replaced by &x (address of x) and &y (address of y).

***p = Content at the address pointed by p.**

***q = Content at the address pointed by q.**

This swapping will be effective as changes were performed at the actual addresses of x and y.

==========

*/

```
{
   int t=*p;
   *p=*q;
   *q=t;
}
```

Output:

x=2 and y=5

x=2 and y=5

x=5 and y=2

Important Concept of C Language to be Remembered:

In C Language, whenever a *variable is modified using a C function*, the *address of the variable to be modified* must be passed as an *input argument to the C function*.

Or in other words, no function in C can change the value of its input arguments.

Call By Value

In this parameter passing method, values of actual parameters are copied to function's formal parameters and the two types of parameters are stored in different memory locations.

So any changes made inside functions are not reflected in actual parameters of caller.

Call by Reference

While calling a function, instead of passing the values of variables, we pass address of variables (location of variables) to the function known as "Call By References.

In this case the changes made inside functions are reflected in the targeted variables (whose addresses were passed) of caller.

****Actually in true sense there is nothing called 'Call by Reference' in C Language. The second approach is also 'Call by Value' as the values of the address were passed in the function.**