## Calling a function

When a program calls a function, the program control is transferred to the called function. A called function performs a defined task and when its return statement is executed or when its function-ending closing brace is reached, it returns the program control back to the main program.

To call a function, you simply need to pass the required parameters along with the function name, and if the function returns a value, then you can store the returned value.

Control of the program is transferred to the user-defined function by calling it.

**Syntax of function call**
functionName(argument1, argument2, ...);

In the above example, the function call is made using addNumbers(n1, n2); statement inside the main() function.

## Call by Value and Call by Reference

Functions can be invoked in two ways: **Call by Value** or **Call by Reference**. These two ways are generally differentiated by the type of values passed to them as parameters.

The parameters passed to function are called *actual parameters* whereas the parameters received by function are called *formal parameters*.

**Call By Value in C**: 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.

In other words, in this parameter passing method, values of actual parameters are copied to function's formal parameters, and the parameters are stored in different memory locations. So any changes made inside functions are not reflected in actual parameters of the caller.

### *Call by Value Example: Swapping the values of the two variables*

```
1. #include <stdio.h>
2. void swap(int , int); //prototype of the function
3. int main()
4. {
5. int a = 10;
6. int b = 20;
7. printf("Before swapping the values in main a = %d, b = %d\n",a,b);
8.  swap(a,b);
9. printf("After swapping values in main a = %d, b = %d\n",a,b);  }
```

```
10. void swap (int a, int b)
11. {
12. int temp;
13. temp = a;
14. a=b;
15. b=temp;
16. printf("After swapping values in function a = %d, b = %d\n",a,b);
17.  }
```

***Output***

```
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 10, b = 20
```

## Call by reference in C

Call by reference method copies the address of an argument into the formal parameter. In this method, the address is used to access the actual argument used in the function call. It means that changes made in the parameter alter the passing argument.

In this method, the memory allocation is the same as the actual parameters. All the operations in the function are performed on the value stored at the address of the actual parameter, and the modified value will be stored at the same address. Means, both the actual and formal parameters refer to same locations, so any changes made inside the function are actually reflected in actual parameters of caller.

***Call by reference Example: Swapping the values of the two variables***

```
1.  #include <stdio.h>
2.  void swap(int *, int *); //prototype of the function
3.  int main()
4.  {
5.     int a = 10;
6.     int b = 20;
7.     printf("Before swapping the values in main a = %d, b = %d\n",a,b);
8.     swap(&a,&b);
9.     printf("After swapping values in main a = %d, b = %d\n",a,b);
10.  }
11. void swap (int *a, int *b)
12. {
13.    int temp;
14.    temp = *a;
15.    *a=*b;
16.    *b=temp;
17.    printf("After swapping values in function a = %d, b = %d\n",*a,*b);
18.  }
```


By: Pragya Singh Tomar
ICS,Vikram University, Ujjain

***Output***

```
Before swapping the values in main a = 10, b = 20
After swapping values in function a = 20, b = 10
After swapping values in main a = 20, b = 10
```

| Call By Value | Call By Reference |
|---|---|
| While calling a function, we pass values of variables to it. Such functions are known as "Call By Values". | 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 method, the value of each variable in calling function is copied into corresponding dummy variables of the called function. | In this method, the address of actual variables in the calling function are copied into the dummy variables of the called function. |
| With this method, the changes made to the dummy variables in the called function have no effect on the values of actual variables in the calling function. | With this method, using addresses we would have an access to the actual variables and hence we would be able to manipulate them. |
| A copy of the value is passed into the function | An address of value is passed into the function |
| Actual and formal arguments are created at the different memory location | Actual and formal arguments are created at the same memory location |


By: Pragya Singh Tomar
ICS,Vikram University, Ujjain