Functions

Today’s Agenda

- Function
- Function Overloading and scope rules
- Difference b/w call by value, call by address and call by reference
- Recursion

Let’s Get Started-

Functions

A function is block of code which is used to perform a particular task.

Instead of doing a particular task several times, you can write these lines inside a function and call that function every time you want to perform that task.

This would make you code simple, readable and reusable.

Function example

Function

Function Declaration: You have seen that I have written the same program in two ways, in the first program I didn't have any function declaration and in the second program I have function declaration at the beginning of the program. The thing is that when you define the function before the main() function in your program then you don't need to do function declaration but if you are writing your function after the main() function like we did in the second program then you need to declare the function first, else you will get compilation error

Function

syntax of function declaration:

return_type function_name(parameter_list);

Function definition: Writing the full body of function is known as defining a function.

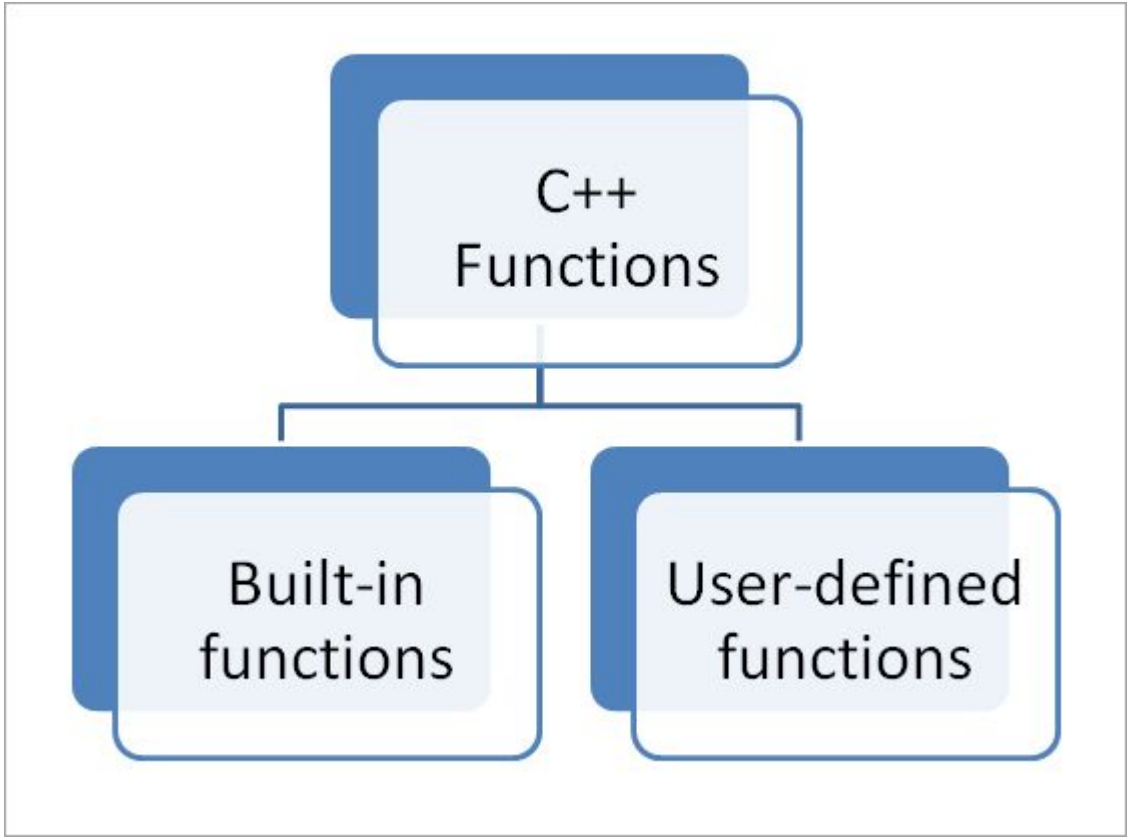
syntax of function definition:

```
return_type function_name(parameter_list) {
   //Statements inside function
}
```

Calling function: We can call the function like this:

function_name(parameters);

Function



Built in function example

User defined function example

Default Parameter List (Default Values for the parameters)

When you define a function, you can specify a default value for each of the last parameters. This value will be used if the corresponding argument is left blank when calling to the function.

This is done by using the assignment operator and assigning values for the arguments in the function definition. If a value for that parameter is not passed when the function is called, the default given value is used, but if a value is specified, this default value is ignored and the passed value is used instead.

Consider the following example −

Function with default values example

Function Overloading

Function overloading is a C++ programming feature that allows us to have more than one function having same name but different parameter list,

when I say parameter list, it means the data type and sequence of the parameters,

Function overloading example

Advantage

The main advantage of function overloading is to the improve the **code readability** and allows **code reusability**. In the example 1, we have seen how we were able to have more than one function for the same task(addition) with different parameters, this allowed us to add two integer numbers as well as three integer numbers, if we wanted we could have some more functions with same name and four or five arguments.

Imagine if we didn't have function overloading, we either have the limitation to add only two integers or we had to write different name functions for the same task addition, this would reduce the code readability and reusability

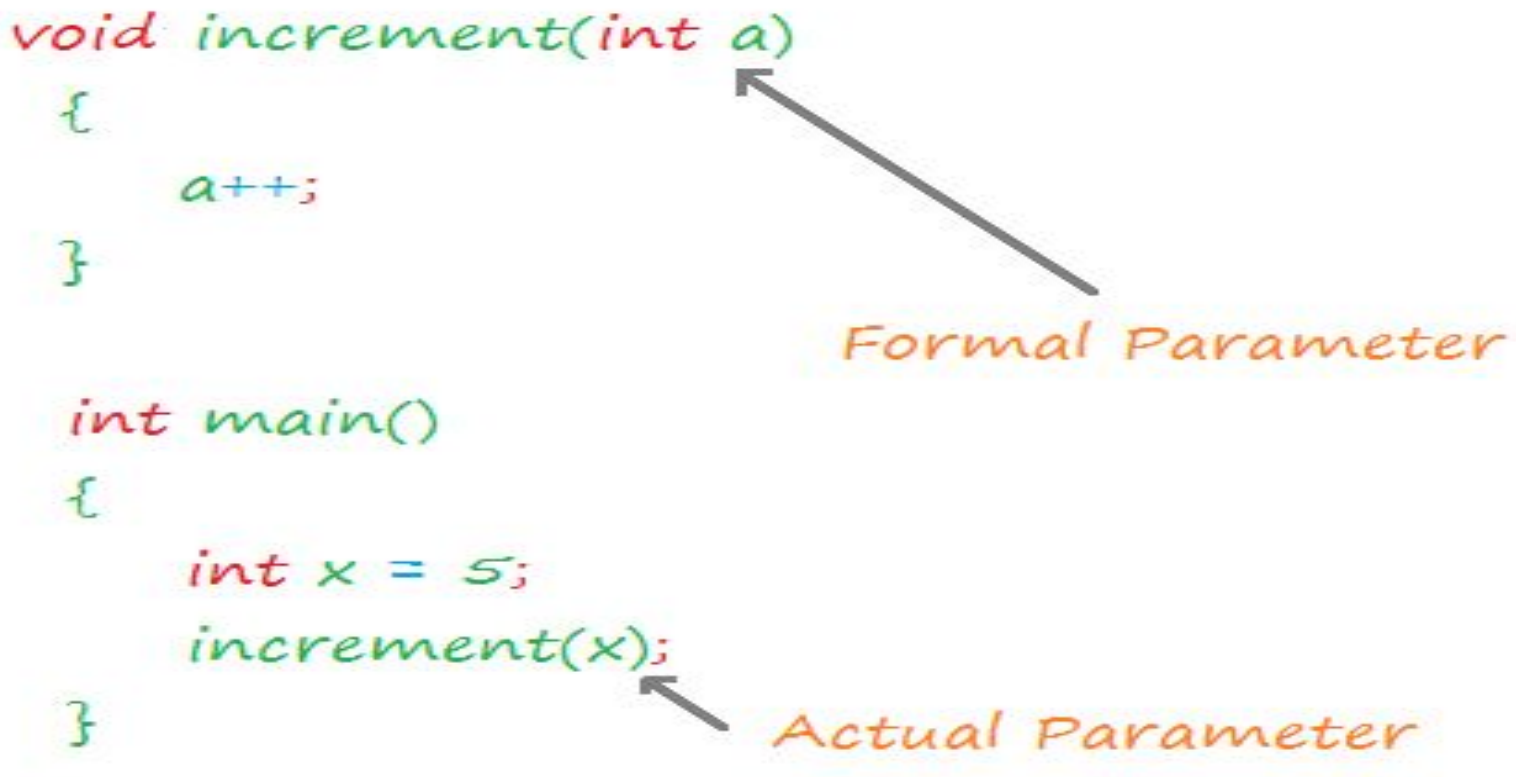
Call by value, Call by reference and Call by address.

To understand the importance of each type of call and their difference, we must know, what the actual and formal parameters are:

Actual Parameters are the parameters that appear in the function call statement.

Formal Parameters are the parameters that appear in the declaration of the function which has been called.

Call by value, Call by reference and Call by address.



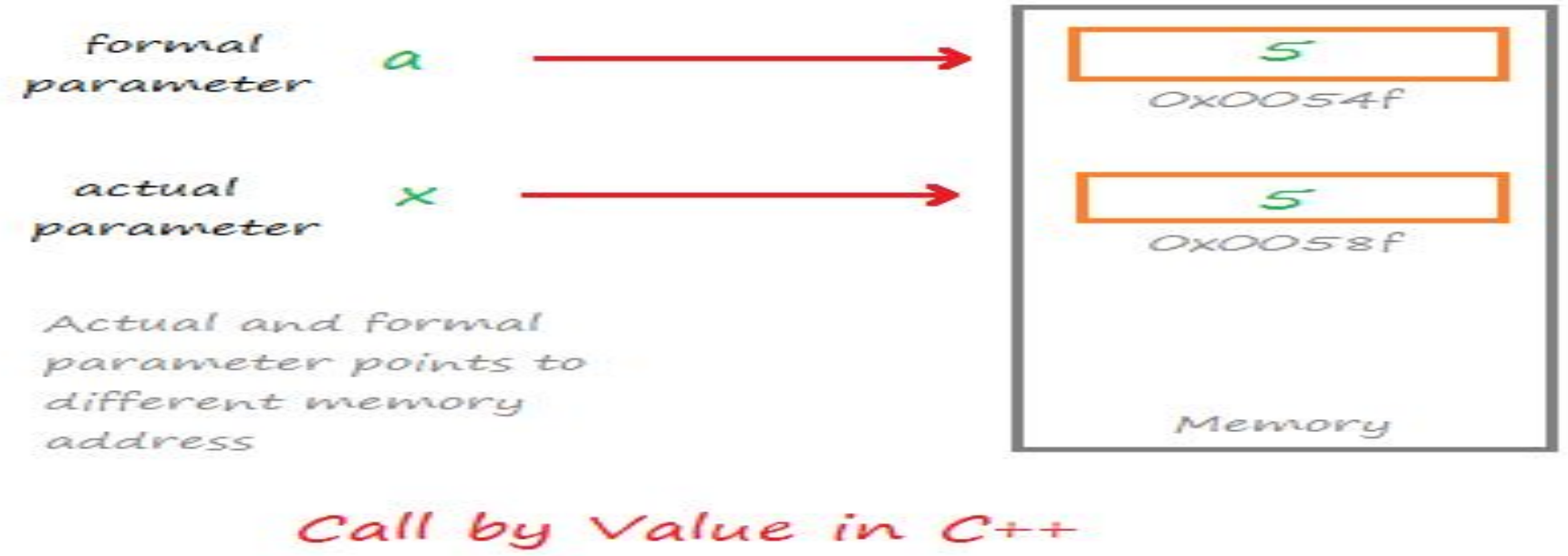
Call by value

When a function is called in the call by value, the value of the actual parameters is copied into formal parameters.

Both the actual and formal parameters have their own copies of values, therefore any change in one of the types of parameters will not be reflected by the other.

This is because both actual and formal parameters point to different locations in memory (i.e. they both have different memory addresses).

Call by value



Call by value method is useful when we do not want the values of the actual parameters to be changed by the function that has been invoked.

Call by value example

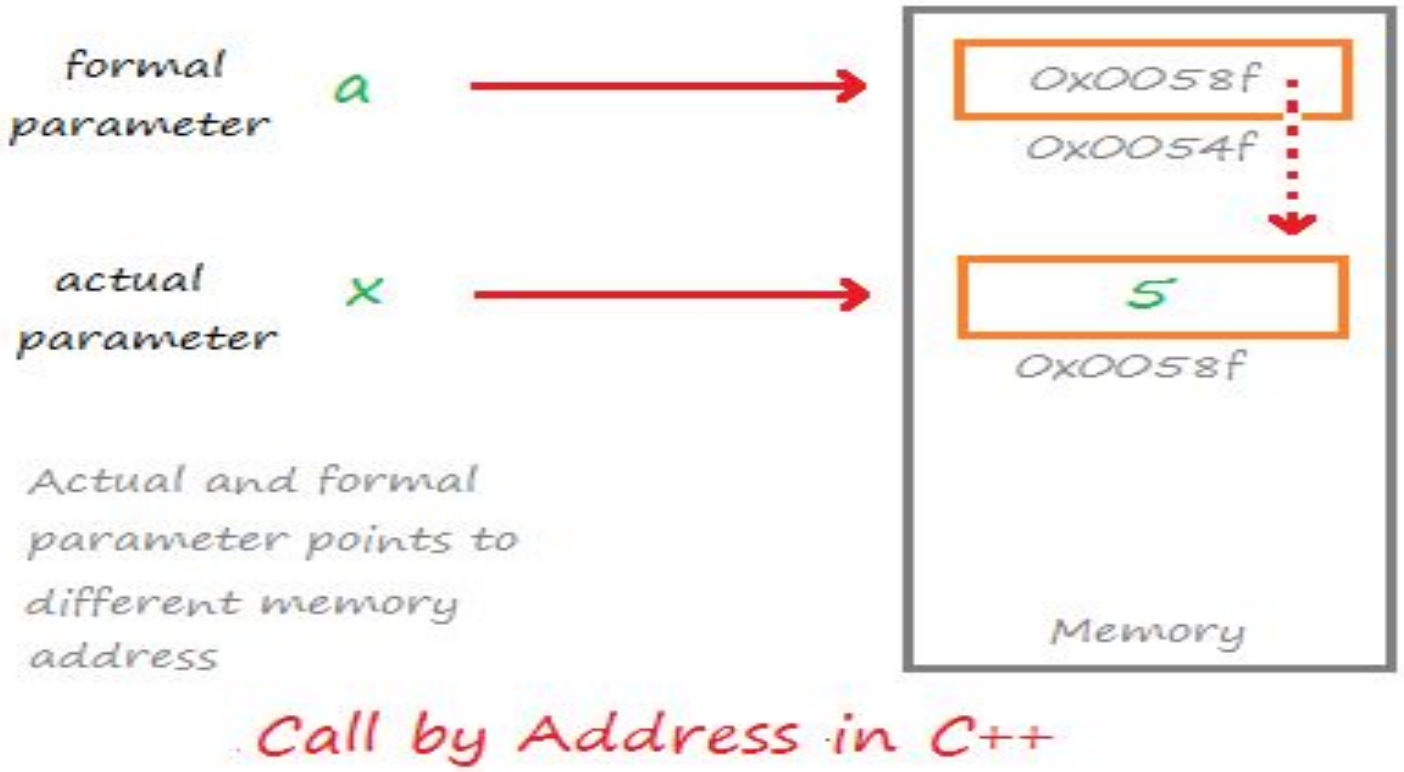
Call by Address

In the call by address method, both actual and formal parameters indirectly share the same variable.

In this type of call mechanism, pointer variables are used as formal parameters.

The formal pointer variable holds the address of the actual parameter, hence the changes done by the formal parameter is also reflected in the actual parameter.

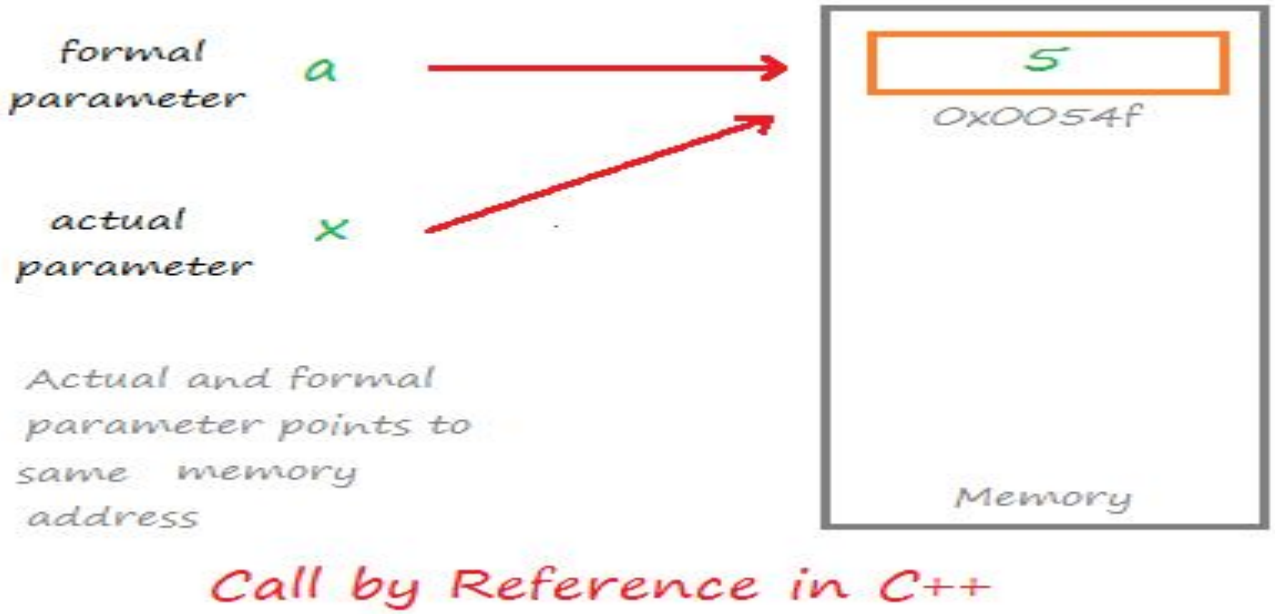
Call by Address



Call by address example

Call by Reference

In the call by reference, both formal and actual parameters share the same value. Both the actual and formal parameter points to the same address in the memory.



Call by Reference

That means any change on one type of parameter will also be reflected by other.
Calls by reference are preferred in cases where we do not want to make copies of objects or variables, but rather we want all operations to be performed on the same copy.

Call by reference example

Conclusion

As a conclusion, we can say that call by value should be used in cases where we do not want the value of the actual parameter to be disturbed by other functions and call by reference and call by address should be used in cases where we want to maintain a variable or a copy of the object throughout the program.

Recursion

When function is called within the same function, it is known as recursion in C++. The function which calls the same function, is known as recursive function.
A function that calls itself, and doesn't perform any task after function call, is known as tail recursion. In tail recursion, we generally call the same function with return statement.

Advantage of Recursion

1. It makes our code shorter and cleaner.

2. Recursion is required in problems concerning data structures and advanced algorithms, such as Graph and Tree Traversal.

Disadvantage of Recursion

1. It takes a lot of stack space compared to an iterative program.
2. It uses more processor time.
3. It can be more difficult to debug compared to an equivalent iterative program.
4. Stack might overflow

Calculator program example

Assignment

- A person is elligible to vote if his/her age is greater than or equal to 18. Define a function to find out if he/she is elligible to vote.

- Write a program which will ask the user to enter his/her marks (out of 100). Define a function that will display grades according to the marks entered as below:

Marks	Grade
91-100	AA
81-90	AB
71-80	BB
61-70	BC
51-60	CD
41-50	DD
<=40	Fail

QNA Time

Any Questions??

Thank You!

See you guys in next class.