C++ Functions

The function in C++ language is also known as procedure or subroutine in other programming languages.

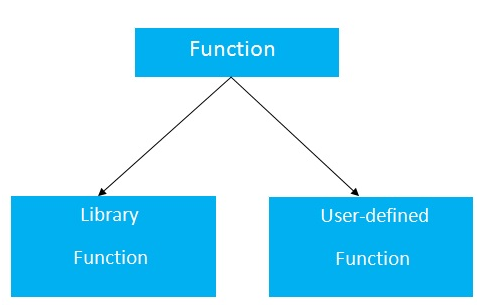
To perform any task, we can create function. A function can be called many times. It provides modularity and code reusability.

## **Types of Functions**

There are two types of functions in C programming:

**1. Library Functions:** are the functions which are declared in the C++ header files such as ceil(x), cos(x), exp(x), etc.

**2. User-defined functions:** are the functions which are created by the C++ programmer, so that he/she can use it many times. It reduces complexity of a big program and optimizes the code.



## **Declaration of a function**

The syntax of creating function in C++ language is given below:

1. return\_type function\_name(data\_type parameter...)
2. {
3. //code to be executed
4. }

**Example**

|  |  |
| --- | --- |
| 1. #include <iostream> 2. using namespace std; 3. void fuck(int x){ 4. int j = 1, k; 5. while(j <= x){ 6. cout << j << "\n"; 7. { 8. for (k = 1; k <= x; k++){ 9. cout << "\t" << k << "\n"; 10. if ( k == 8) 11. break; 12. } 13. } 14. j++; 15. if (j == 6){ 16. break; 17. } 18. } 19. } 20. int main () { 21. int x; 22. cout << "type the fuckin number"; 23. cin >> x; 24. fuck(x); 25. } | 1  1  2  3  2  1  2  3  3  1  2  3 |

# Call by value and call by reference in C++

|  |  |
| --- | --- |
| Call by value | Call by reference |
| #include <iostream>using namespace std;void change(int data);int main(){int data = 3;change(data);cout << "Value of the data is: " << data<< endl;return 0;}void change(int data){data = 5;}//3 | #include<iostream>using namespace std;void swap(int \*x, int \*y){int swap;swap=\*x;\*x=\*y;\*y=swap;}int main(){int x=500, y=100;swap(&x, &y); // passing value to functioncout<<"Value of x is: "<<x<<endl;cout<<"Value of y is: "<<y<<endl;return 0;}/\*Value of x is: 100Value of y is: 500\*/ |

## **Difference between call by value and call by reference in C++**

|  |  |  |
| --- | --- | --- |
| **No.** | **Call by value** | **Call by reference** |
| 1 | A copy of value is passed to the function | An address of value is passed to the function |
| 2 | Changes made inside the function is not reflected on other functions | Changes made inside the function is reflected outside the function also |
| 3 | Actual and formal arguments will be created in different memory location | Actual and formal arguments will be created in same memory location |