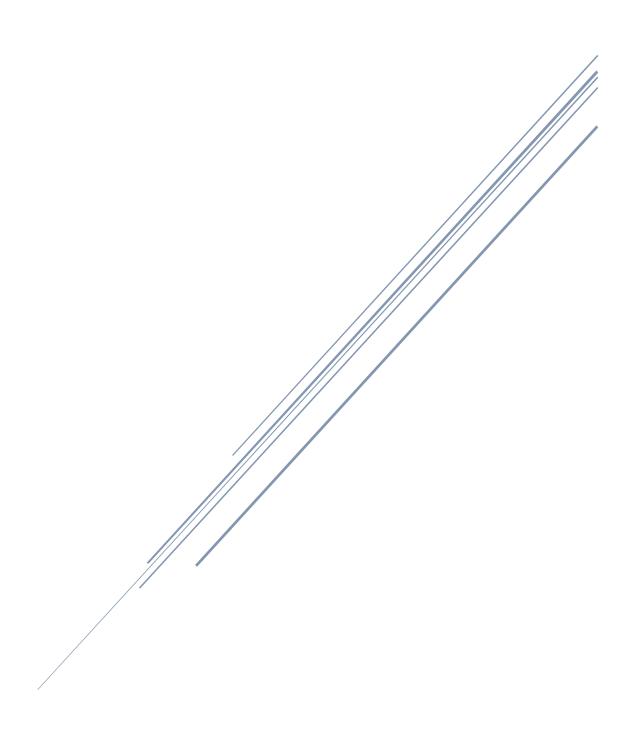
FLOW CONTROL, FUNCTION & LOOPS



INDEX

Table of Contents

Switch Statement	2
Function	
Formal & Actual Parameters	
Pass by value & Pass by Reference	
Default Argument in Function	
Function Overloading	

Switch Statement

Remember:

The expression used in case should be constant. We can't use variable in case.

We cannot use float, double, string or user defined data type in the switch statement.

```
#include <iostream>
using namespace std;
int main() {
    int x = 0; // Initial x-coordinate
    int y = 0; // Initial y-coordinate
    char move;
    cout << "Enter movement ('1', 'r', 'u', 'd'): ";</pre>
    cin >> move;
    switch (move) {
            y--;
            break;
            y++;
            break;
            X--;
            break;
            χ++;
            break;
        default:
            cout << "Invalid movement!" << endl;</pre>
            return 1; // Exit program with error code
    3
    cout << "Updated x-coordinate: " << x << endl;</pre>
    cout << "Updated y-coordinate: " << y << endl;</pre>
    return 0;
```

Function

In short, a function in C++ is a named block of code that can be invoked (called) from other parts of the program to perform a specific action or computation, and it can optionally return a value.

```
< return-type > < function-name > (< set-of-arguments >)
{
    //block of statements
}
```

Skip

C++ Advanced Function Topics

Formal & Actual Parameters

The parameters passed to function are called actual parameters.

The parameters received by function are called formal parameters.

```
#include <iostream>
using namespace std;
// Function declaration with formal parameters (int a, int b)
void printSum(int a, int b);
int main() {
    int x = 5, y = 3;
    // Call the function printSum with actual parameters (x and y)
    printSum(x, y);
    return 0;
}
// Function definition with formal parameters (int a, int b)
void printSum(int a, int b) {
    // Calculate the sum of formal parameters (a and b)
    int sum = a + b;
    // Print the calculated sum
    cout << "Sum: " << sum << endl;</pre>
```

```
Sum: 8
```

```
#include <iostream>
using namespace std;
// Function to modify a value by passing by value
void modifyValue(int x) {
    x = 10; // Modifying the copy of x
}
// Function to modify a value by passing by reference
void modifyReference(int &x) {
    x = 10; // Modifying the original value of x
}
int main() {
    int num1 = 5;
    int num2 = 5;
    cout << "Original value of num1: " << num1 << endl;</pre>
    modifyValue(num1);
    cout << "Value of num1 after modifyValue: " << num1 << end1;</pre>
    cout << "Original value of num2: " << num2 << end1;</pre>
    modifyReference(num2);
    cout << "Value of num2 after modifyReference: " << num2 << endl;</pre>
    return 0;
}
```

```
Original value of num1: 5

Value of num1 after modifyValue: 5

Original value of num2: 5

Value of num2 after modifyReference: 10
```

Remember:

All Default argument must appear at the end, otherwise it will give compiler error.

Default argument can be given in function declaration, and If default argument provided in function declaration, then no need to write again in function definition, otherwise it will give **compiler error**.

```
#include <iostream>
using namespace std;
void printDetails(int id, string name = "NA", string address = "NA")
{
    cout << "Id: " << id << '\n';
    cout << "Name: " << name << '\n';</pre>
    cout << "Address: " << address << '\n';</pre>
}
int main()
{
    printDetails(101, "Sandeep", "Noida");
    cout << '\n';
    printDetails(201, "Shivam");
    cout << '\n';
    printDetails(301);
    return 0;
}
```

```
Id: 101
Name: Sandeep
Address: Noida

Id: 201
Name: Shivam
Address: NA

Id: 301
Name: NA
Address: NA
```

Function Overloading

Remember:

Function can't be overloaded based on return type only.

```
#include <iostream>
using namespace std;
int add(int a, int b) {
    return a + b;
}
// This is not allowed due to the same parameter types and order
float add(int a, int b) {
    return static_cast<float>(a + b);
}
int main() {
    int result1 = add(3, 4);
    float result2 = add(3, 4);
    cout << "Result 1: " << result1 << endl;</pre>
    cout << "Result 2: " << result2 << endl;</pre>
    return 0;
3
```