FLOW CONTROL, FUNCTION & LOOPS
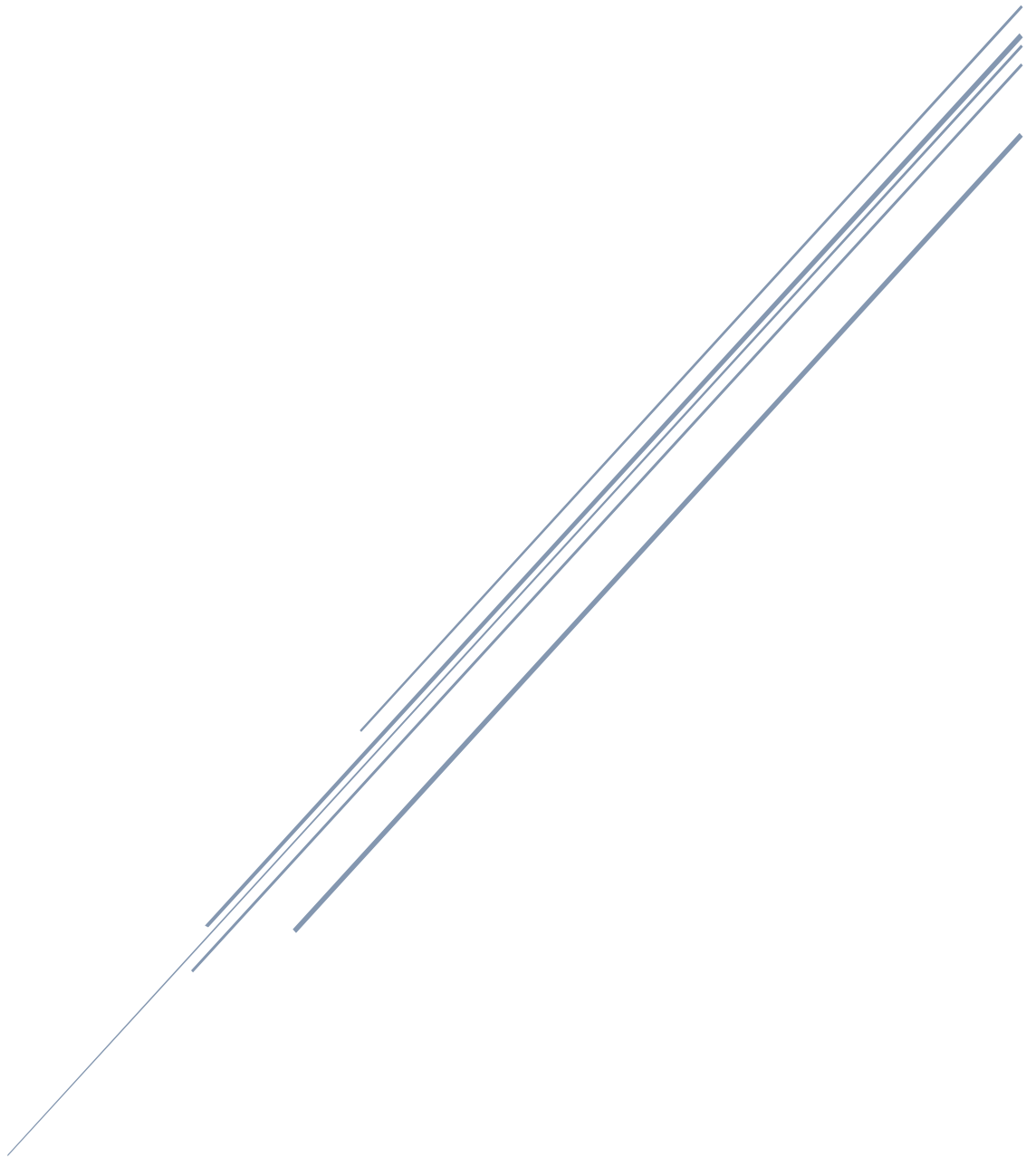
INDEX

Table of Contents

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.

```cpp
#include <iostream>
using namespace std;

int main() {
    int x = 0; // Initial x-coordinate
    int y = 0; // Initial y-coordinate

    char move;

    cout << "Enter movement ('l', 'r', 'u', 'd'): ";
    cin >> move;

    switch (move) {
        case 'l':
            y--;
            break;
        case 'r':
            y++;
            break;
        case 'u':
            x--;
            break;
        case 'd':
            x++;
            break;
        default:
            cout << "Invalid movement!" << endl;
            return 1; // Exit program with error code
    }

    cout << "Updated x-coordinate: " << x << endl;
    cout << "Updated y-coordinate: " << y << endl;

    return 0;
}
```

Default Argument in Function

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**.

```cpp
#include <iostream>
using namespace std;

void printDetails(int id, string name = "NA", string address = "NA")
{
    cout << "Id: " << id << '\n';
    cout << "Name: " << name << '\n';
    cout << "Address: " << address << '\n';
}

int main()
{
    printDetails(101, "Sandeep", "Noida");
    cout << '\n';
    printDetails(201, "Shivam");
    cout << '\n';
    printDetails(301);
    return 0;
}
```

OP

```
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.

```cpp
#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;
    cout << "Result 2: " << result2 << endl;

    return 0;
}
```

OP

```
main.cpp:9:7: error: ambiguating new declaration of 'float add(int, int)'
    9 | float add(int a, int b) {
      |       ^~~
```