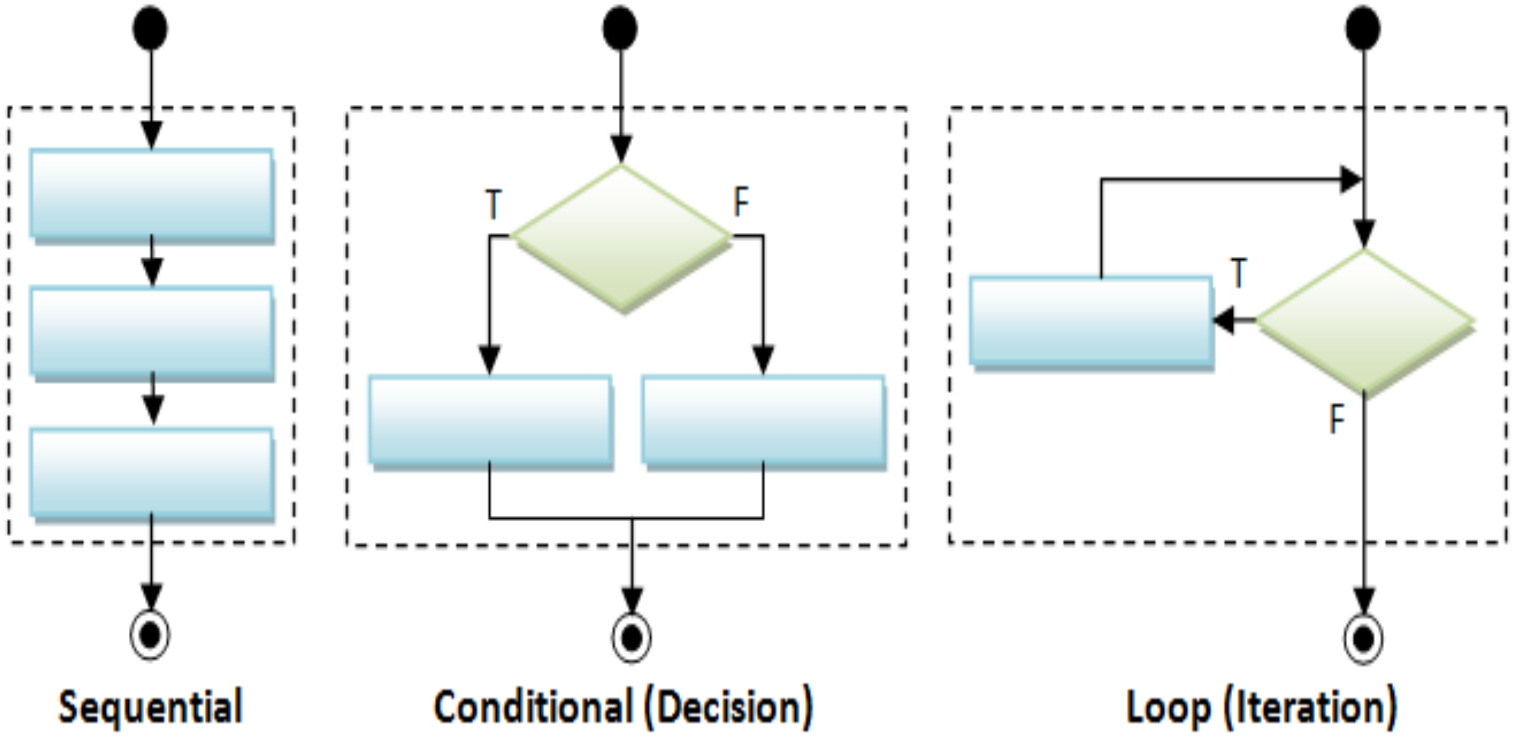
Fundamentals of Computer Programming

CS-110

Course Instructor: Dr. Momina Moetesum

Conditional Statements

Week 3



Learnning Objectives

01

To understand the flow of control in a C++ program

02

To use basic flowcharts to show the flow of control of the C++ program

03

To use selection statements like if, if-else, else-if, switch

04

To get familairization with the terniary conditional operator

Basic Flowcharts

Symbol	Name	Description
	Rectangular or action	A process or action such as calculation, input/output, assignment etc.
	Oval	Represents a complete algorithm Begin/Start, End/Stop.
	Diamond or decision	Which indicates that a decision to be made such as choice between YES or NO.
	Flowlines	Indicates the order in which the actions are to be performed.
	Small circle or connector symbol	When describing a portion of a complete algorithm continued from or will continue on.
	Input or output	The data or input/output.

- A flowchart is a graphical representation of an algorithm or a portion of an algorithm.
- It is drawn using certain special-purpose symbols such as rectangles, diamonds, ovals, and small circles.
- These symbols are connected by arrows called flow lines.
- Flowcharts can clearly show how control structures operate.

Flow Control in Programming

Program begins execution at the main() function.

Statements within the main() function are then executed from top to down style.

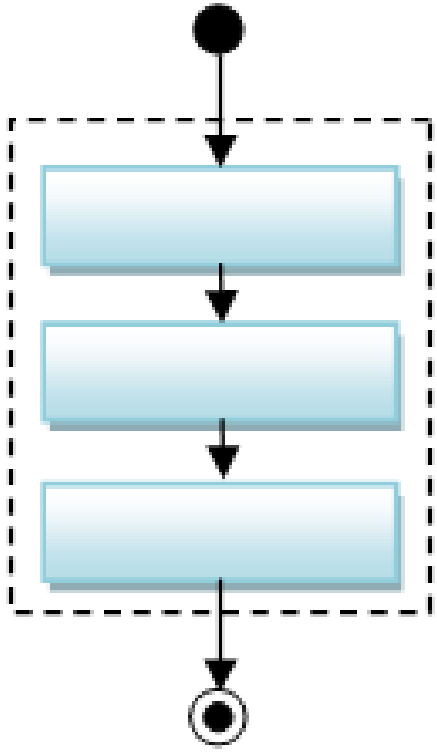
The first statement, then the second and so forth, until the end of the main() function is reached.

However, this order is rarely encountered in real C/C++ program.

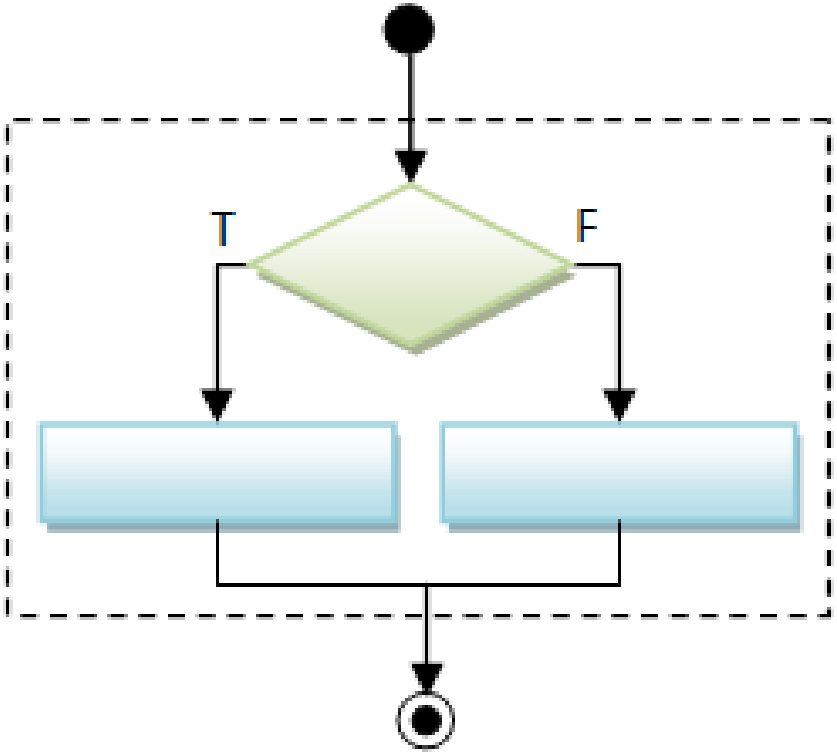
The order of the execution of the statements within the main() body may be redirected, not in sequence anymore.

This concept of changing the order in which statements are executed is called program control and is accomplished by using program control statements. This is how we can control the program flows.

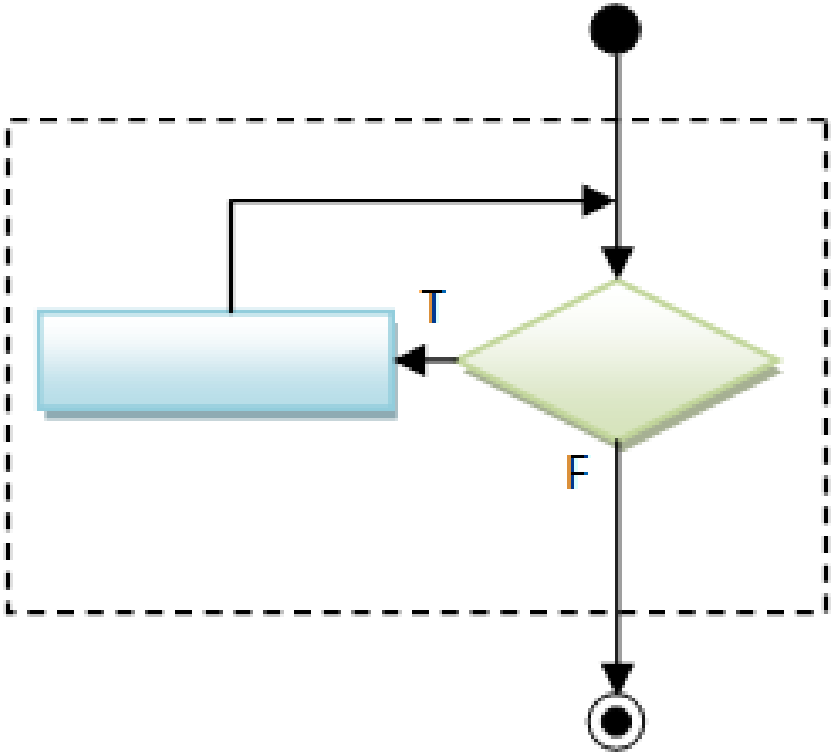
Control Structures in C++



Sequential



Conditional (Decision)



Loop (Iteration)

Control Structures in C++

Sequential

Conditional

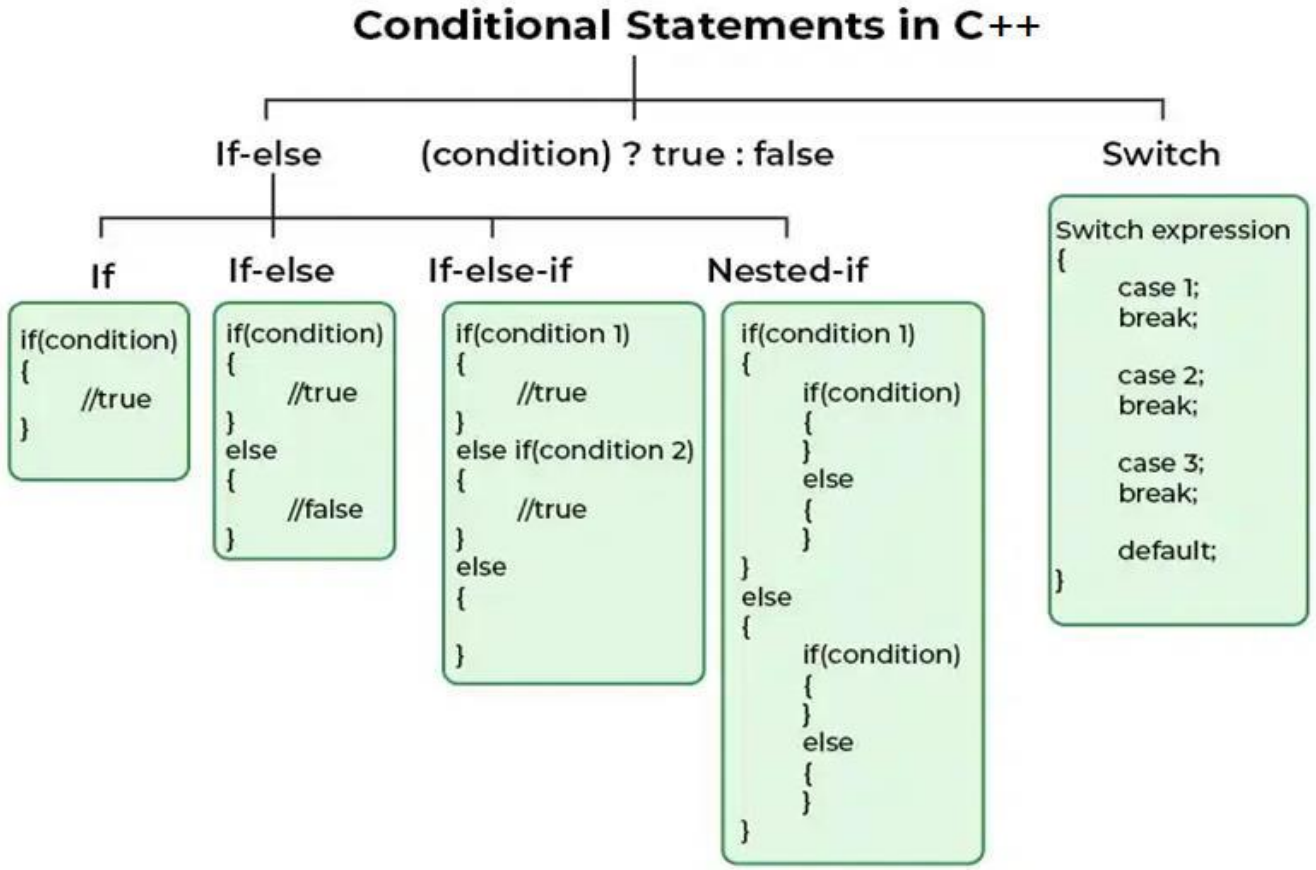
- If-else
- Else-if
- Switch
- Ternary Conditional Operator
- Nested conditional

Iteration

- While Loop
- Do-while Loop
- For Loop
- Nested Loops

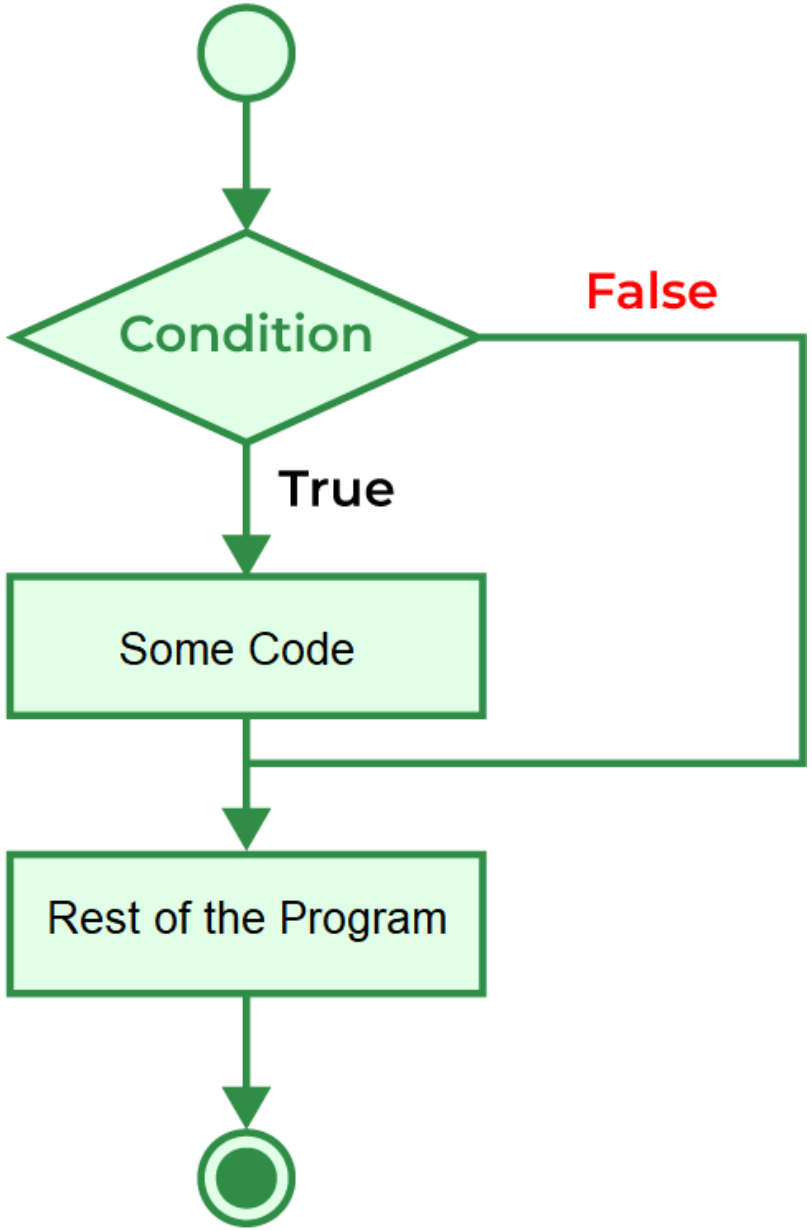
Conditional Statements

- There are situations in real life when we need to make some decisions and based on these decisions, we decide what should we do next.
- Similar situations arise in programming also where we need to make some decisions and based on these decisions, we will execute the next block of code.



If-Statement

- The if statement is the most simple decision-making statement.
- It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statements is executed otherwise not.



Example

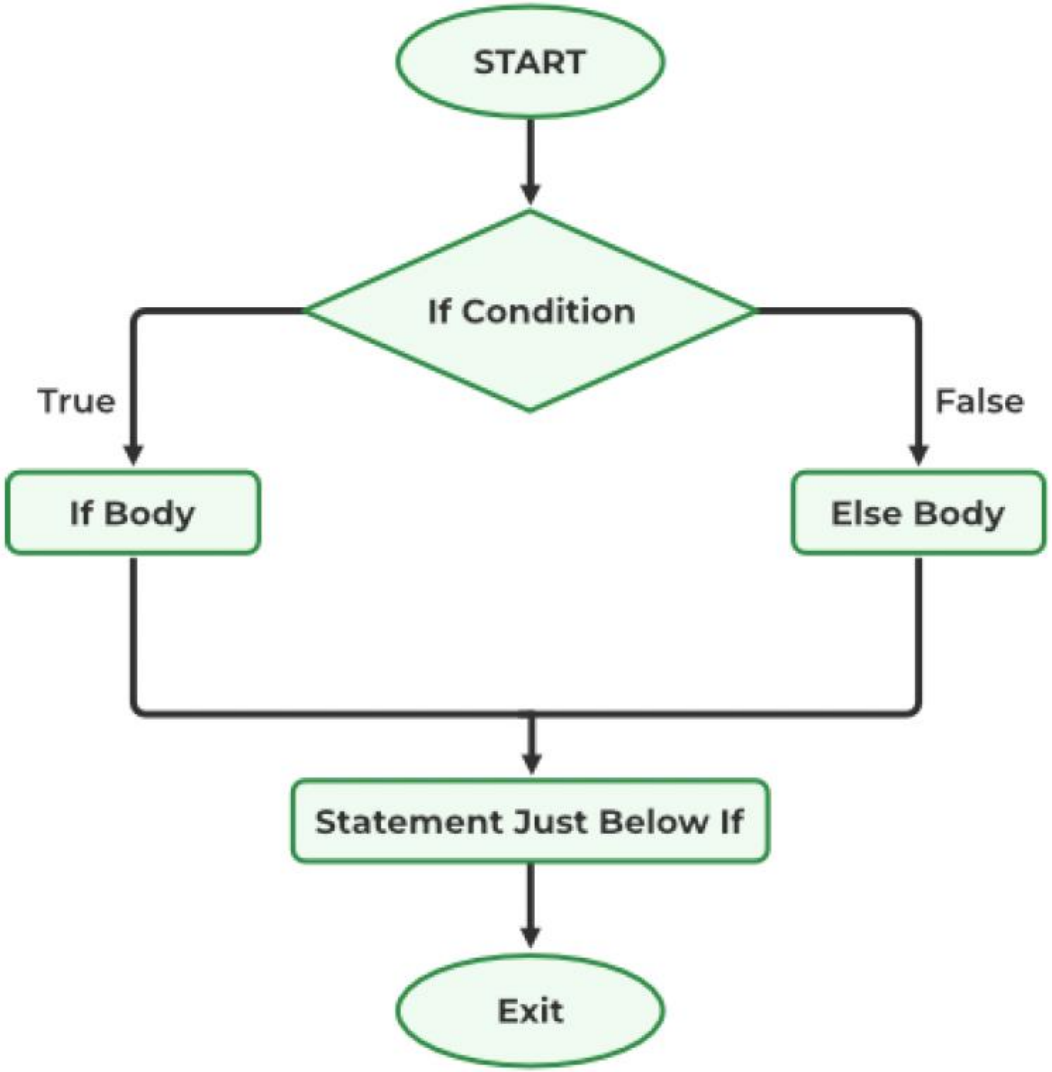
```cpp
// C++ program to illustrate If statement
#include <iostream>
using namespace std;

int main()
{
    int i = 10;

    if (i > 15) {
        cout << "10 is greater than 15";
    }

    cout << "I am Not in if";
}
```

Output

```
I am Not in if
```

If-Else Statement

- The if-else statement consists of two blocks:
  - One for false expression and
  - One for true expression



Example

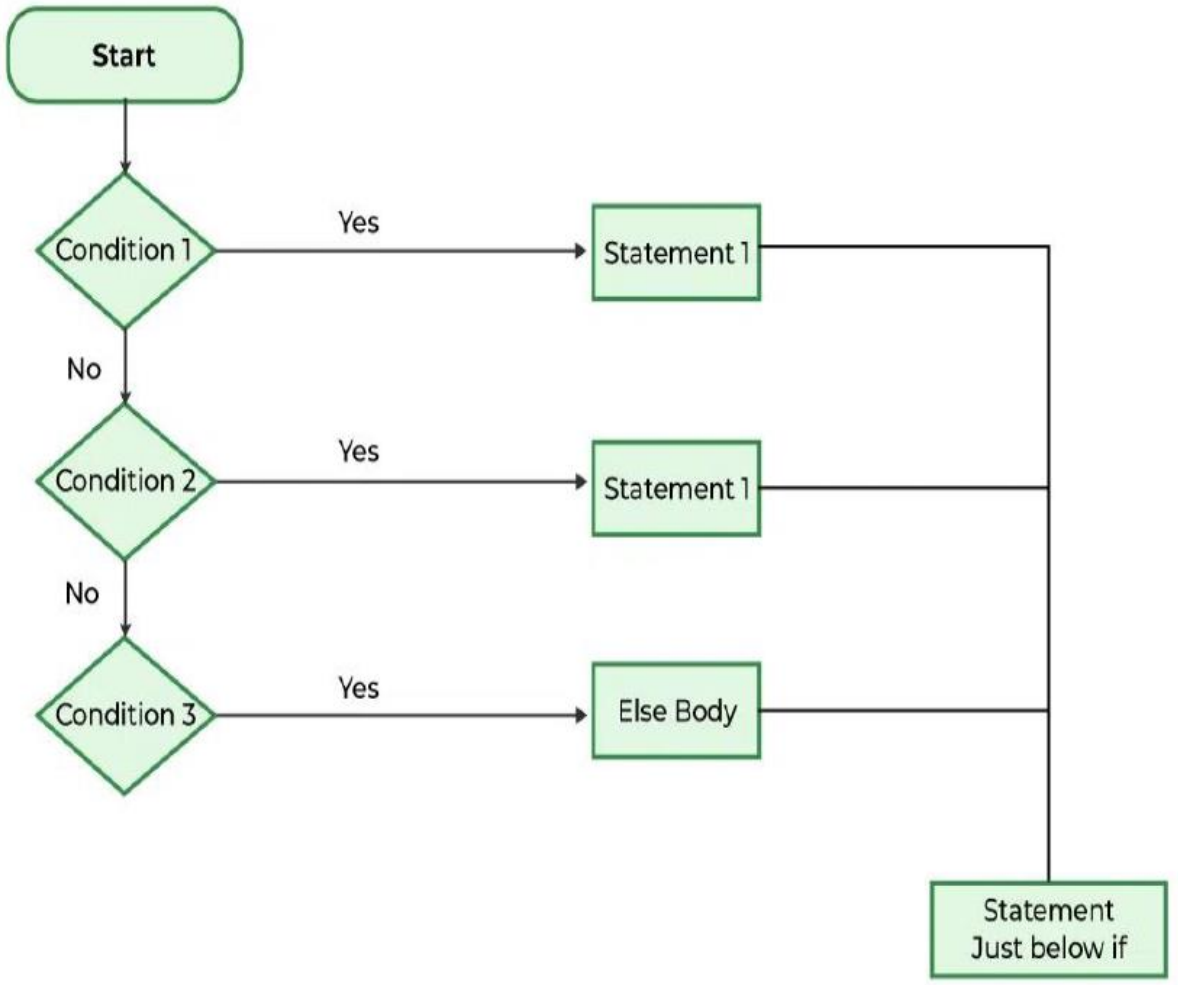
```cpp
// C++ program to illustrate if-else statement
#include <iostream>
using namespace std;

int main()
{
    int i = 20;

    if (i < 15)
        cout << "i is smaller than 15";
    else
        cout << "i is greater than 15";

    return 0;
}
```

Output

```
i is greater than 15
```

Else-if Statement

- These are used when the user has to decide among multiple options.
- The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the else-if ladder is bypassed.
- If none of the conditions is true, then the final else statement will be executed.



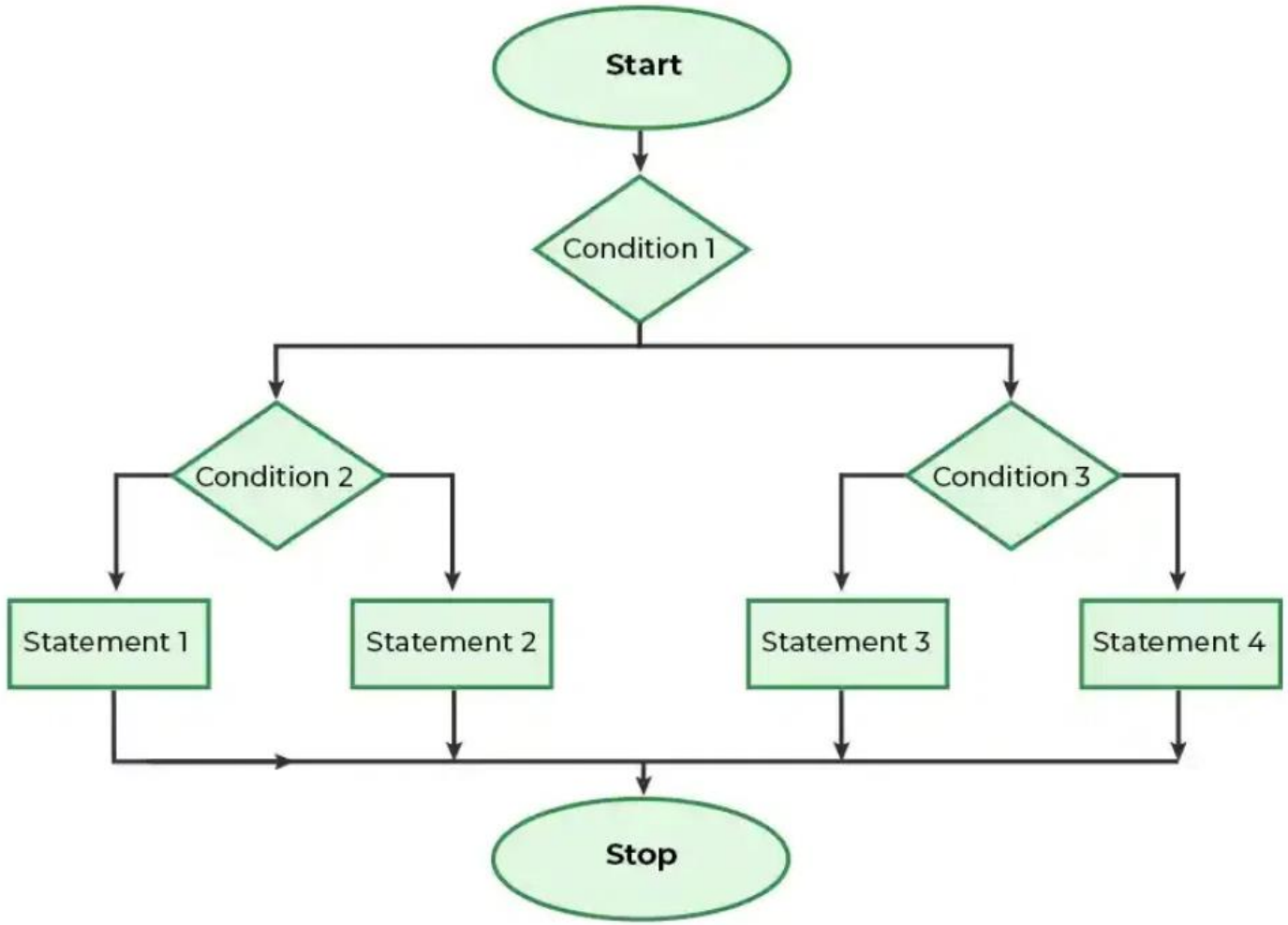
Example

```cpp
// C++ program to illustrate if-else-if ladder
#include <iostream>
using namespace std;

int main()
{
    int i = 20;

    if (i == 10)
        cout << "i is 10";
    else if (i == 15)
        cout << "i is 15";
    else if (i == 20)
        cout << "i is 20";
    else
        cout << "i is not present";
}
```

Output

```
i is 20
```

Nested if-else Statements



Example

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

int main()
{
    int i = 10;

    if (i == 10) {
        // First if statement
        if (i < 15)
            cout << "i is smaller than 15\n";

        // Nested - if statement
        // Will only be executed if
        // statement above is true
        if (i < 12)
            cout << "i is smaller than 12 too\n";
        else
            cout << "i is greater than 15";
    }

    return 0;
}
```

Output

```
i is smaller than 15
i is smaller than 12 too
```

Ternary Conditional Operator (?:)

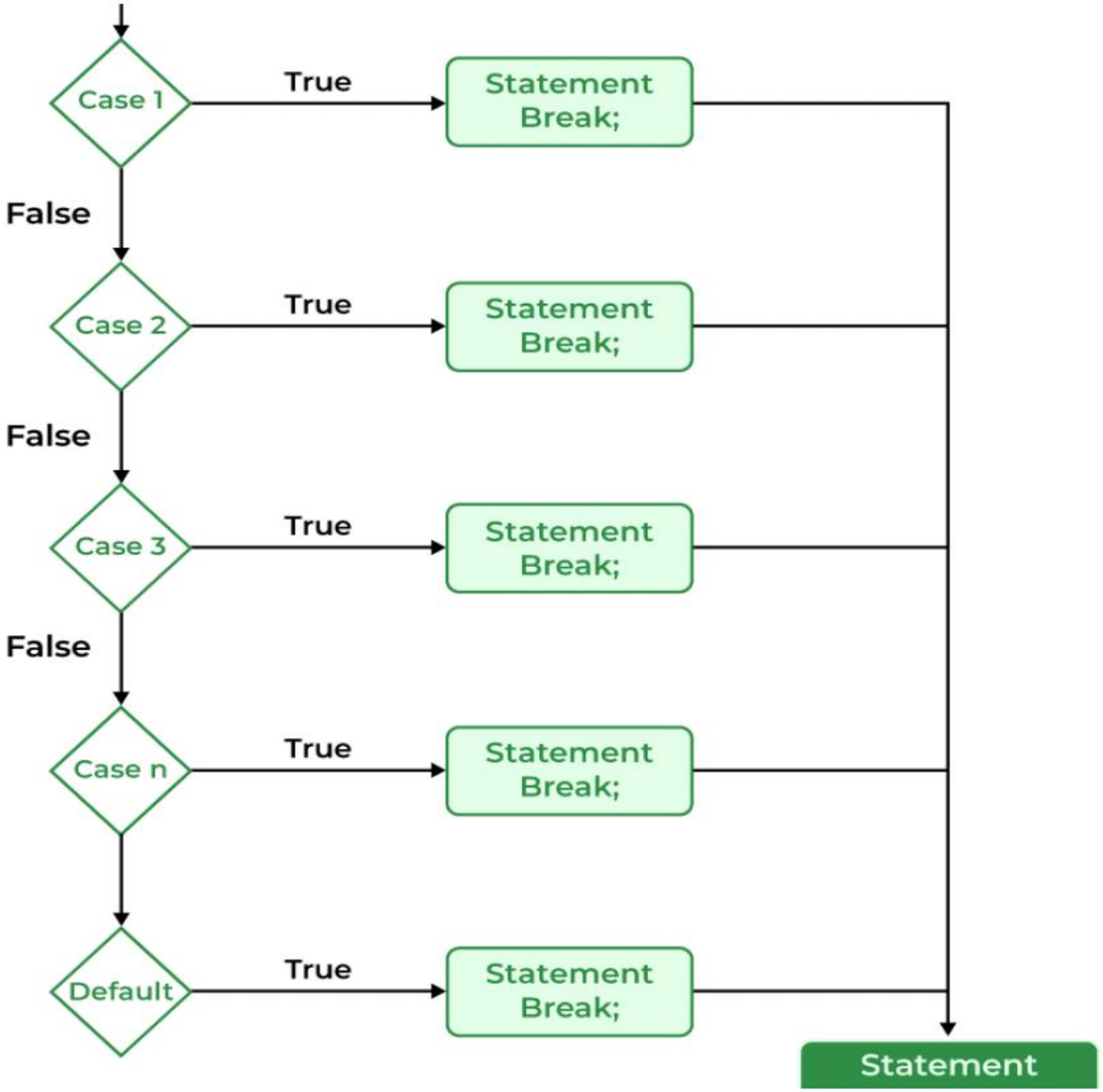
```cpp
#include <iostream>
#include <string>
using namespace std;

int main() {
  double marks;

  // take input from users
  cout << "Enter your marks: ";
  cin >> marks;

  // ternary operator checks if
  // marks is greater than 40
  string result = (marks >= 40) ? "passed" : "failed";

  cout << "You " << result << " the exam.";

  return 0;
}
```

Output 1

```
Enter your marks: 80
You passed the exam.
```

Switch Statement

- The Switch Case Statement is an alternative to the if else if ladder that can be used to execute the conditional code based on the value of the variable specified in the switch statement.
- The switch block consists of cases to be executed based on the value of the switch variable.



Example

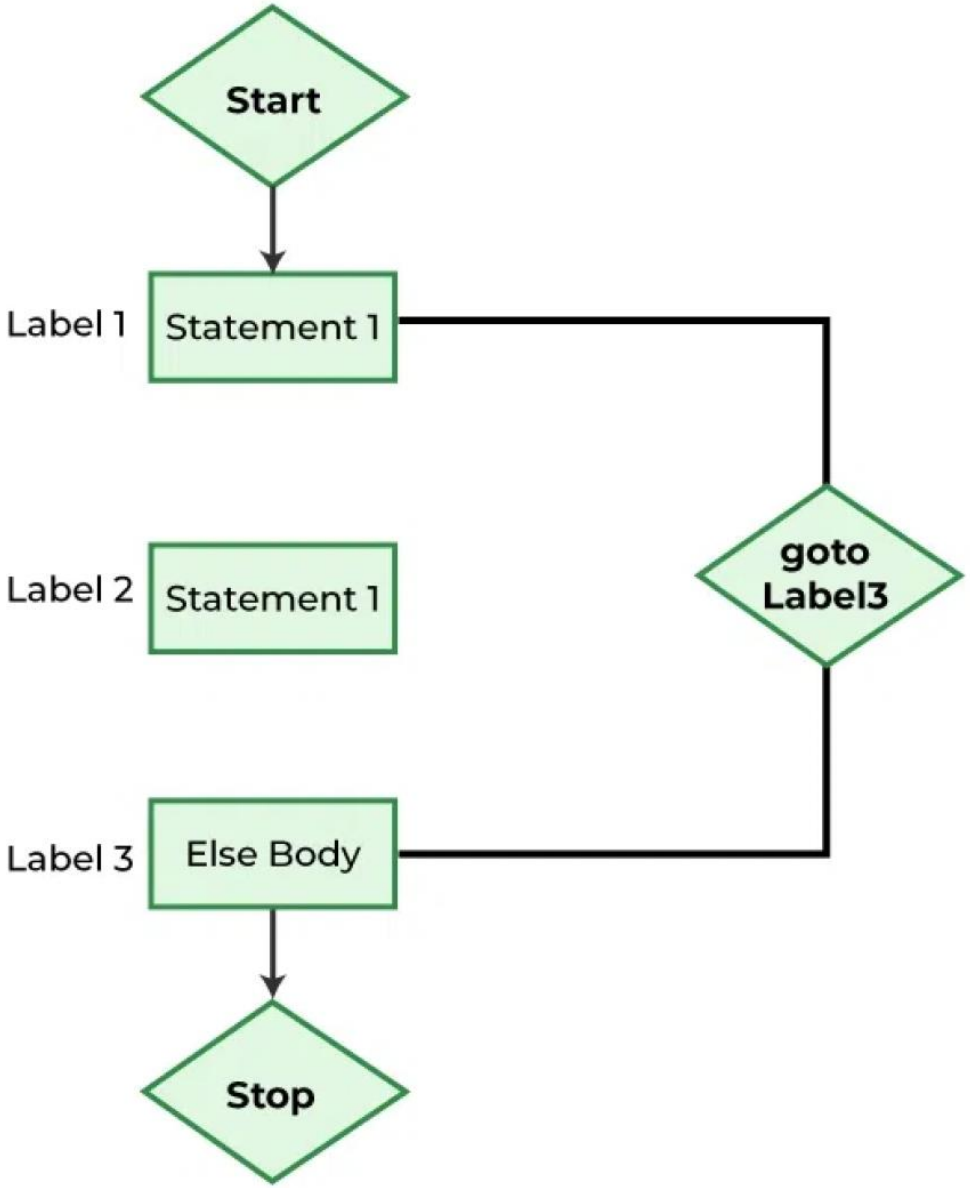
```
// C Program to illustrate the use of switch statement
#include <iostream>
using namespace std;

// driver code
int main()
{
    // variable to be used in switch statement
    int var = 2;

    // declaring switch cases
    switch (var) {
    case 1:
        cout << "Case 1 is executed";
        break;
    case 2:
        cout << "Case 2 is executed";
        break;
    default:
        cout << "Default Case is executed";
        break;
    }

    return 0;
```

Output

```
Case 2 is executed
```

Jump Statements: Goto

- The **goto statement** is a jump statement which is sometimes also referred to as an **unconditional jump** statement.
- The **goto statement** can be used to jump from anywhere to anywhere within a function.



Example

```cpp
// C++ program to check if a number is
// even or not using goto statement
#include <iostream>

int main()
{
    int num = 27;
    if (num % 2 == 0)
        // jump to even
        goto even;
    else
        // jump to odd
        goto odd;

even:
    std::cout<<num<<" is even";
    return 0;
odd:
    std::cout<<num<<" is odd";
    return 0;
```

Disadvantages of Using a Goto Statement

- The use of the goto statement is highly discouraged as it makes the program logic very complex.
- The use of goto makes tracing the flow of the program very difficult.
- The use of goto makes the task of analyzing and verifying the correctness of programs (particularly those involving loops) very difficult.
- The use of goto can be simply avoided by using break and continue statements.

Acknowledgment

- Content of these slides are taken from:
  - https://www.geeksforgeeks.org/
  - https://www.tutorialspoint.com/
  - https://www.programiz.com/