Lab 03

Objectives

After performing this lab, students will be able to

- Understand the basics of string data type and use it in programs
- Understand the if-else statement and create programs using it

Basics of string data type

The **string** is a class, different from the primitive data types discussed so far. We will discuss it in detail in the later part of this course.

Because a **char** variable can store only one character in its memory location, another data type is needed for a variable able to hold an entire string (collection of character/more than 1 characters). Although C++ does not have a built-in data type able to do this, standard C++ provides something called the **string** class that allows the programmer to create a **string** type variable.

The first step in using the **string** class is to #include the string header file. That can be done as shown below:

```
#include <string>
```

You have to use double quotes around the text to store it into a **string** variable. The following code snippet shows how to create a variable of type string:

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
  string name = "John Smith";
  cout << "Name is " << name << endl;

  return 0;
}
```

Output:

```
Name is John Smith
```

Getting Input into a string variable

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
  string name;
  cout << "Enter your name: ";
  cin >> name;
  cout << "Name is " << name << endl;

  return 0;
}
```

Output:

```
Enter your name: James Gosling
Name is James
```

Here, only the first word "James" is assigned to the variable. The text after the space after the first word is ignored. To get the entire line of text, we should use a built-in function **getline ( )** as shown below

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
  string name;
  cout << "Enter your name: ";
  getline(cin, name);
  cout << "Name is " << name << endl;

  return 0;
}
```

Output:

```
Enter your name: James Gosling
Name is James Gosling
```

Output with another value (consisting of three words):

```
Enter your name: John Anthony Charles
Name is John Anthony Charles
```

Flow of control (Control flow)

The order in which statements are executed is called as the flow of control (or control flow). Normally, the instructions are executed in a sequential order. **Branch** statements let program choose between two alternatives. For example, if the student has obtained 60 or above marks, then he/she can be declared as **pass**. These instructions are sometimes also called as **selection statements**. The selection statements allow to choose the set of instructions for execution depending upon a controlling expression's truth value (whether it is **true** or **false**). C++ provides following two types of selection statements:

- if (or if-else) statements
- switch statement

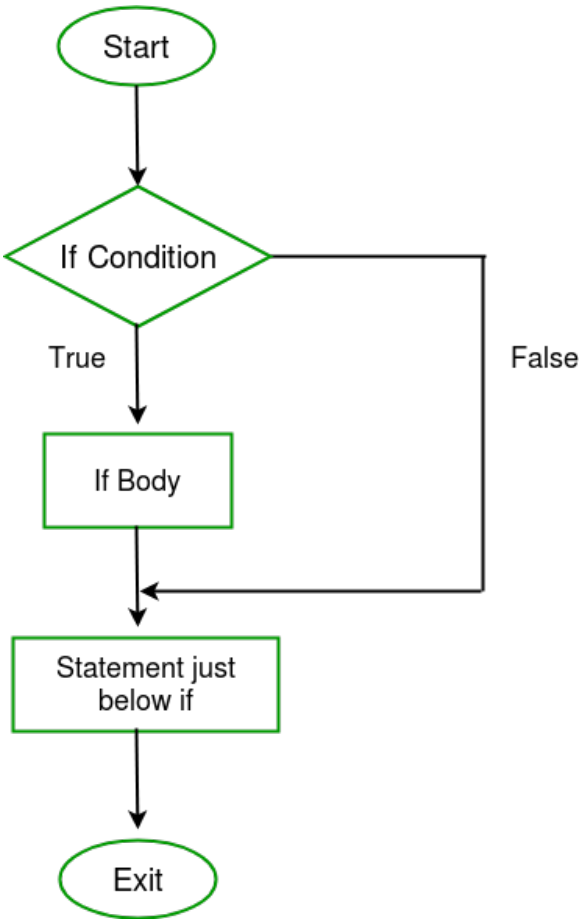
The if statement

In C++, ***if*** statement is used to perform branching (or sometimes also called as **jumping** because the code is not executed sequentially. Instead, the execution depends upon a condition). Its general syntax is

```
if (condition) {
  some code
}
```

If the ***condition*** evaluates to a non-zero value (or ***true***), then the statements within **if** are executed. Otherwise, the statements are skipped, and the program continues on.

This can be illustrated by the following figure:



Example:

```
int marks;
cout << "Enter marks: ";
cin >> marks;

if (marks >= 60)
  cout << "You have passed" << endl;
```

Output 1:

```
Enter marks: 60
You have passed
```

Entered marks are equal to 60, hence the output is “You have passed”

Output 2:

```
Enter marks: 55
```

Now, the entered marks are less than 60. So, the condition of if statement (marks >= 60) is not true. Therefore, the code inside the **if** block does not run (does not print anything).

If you have only one line of code that should be executed when the condition true, then you can write the code as shown above (i.e., no need to write the curly brackets). However, it is a good practice (and a recommended way) to use the brackets, as shown below:

```
if (marks >= 60) {
    cout << "You have passed" << endl;
}
```

The code inside the brackets is called the ***body of if statement***.

If multiple lines of codes need to be executed inside the if statement, then you **must** write those multiple lines inside the curly brackets. Otherwise, only the first line after the if statement will be considered its part. For example,

```
if (marks >= 60) {
    cout << "Congratulations! ";
    cout << "You have passed" << endl;
}
```

Output:

```
Enter marks: 75
Congratulations! You have passed
```

Here, the code works as expected. Nevertheless, if we do not use the curly brackets, like

```
if (marks >= 60)
    cout << "Congratulations! ";
    cout << "You have passed" << endl;
```

Output 1:

```
Enter marks: 77
Congratulations! You have passed
```

Even though now the output looks correct. But it is not. Only the first line (printing "Congratulations!") is part of the ***if*** statement. The second line (saying "You have passed") is the normal line and will be executed regardless of whether the condition is true or false.

Output 2:

```
Enter marks: 34
You have passed
```

Even though now when marks are less than 60, it says "You have passed" because that line does not depend on the condition (because it is NOT the part of ***if*** statement. Only the line saying

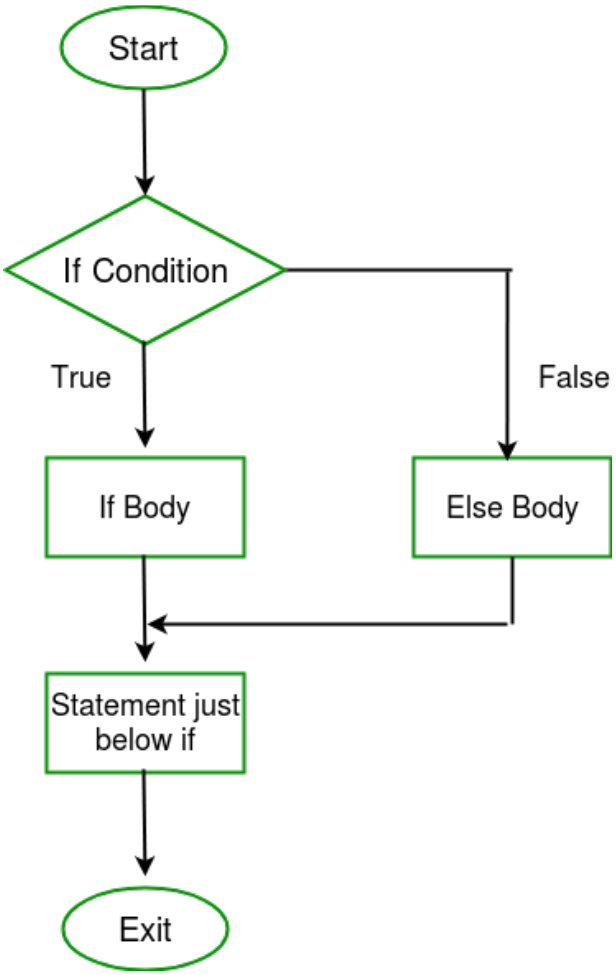
"Congratulations!" is skipped because that was considered to be the part of ***if*** statement. Hence, it is recommended to use curly brackets (even if there is only one line.)

The if-else statement

If the situation is such that there are two options/possibilities. If the condition becomes true, then one statement should execute. On the other hand, if the condition becomes false, then the second statement should execute. In such case, ***if-else*** statement should be used. Its general syntax is:

```
if (Boolean_Expression)
    Yes_Statement
else
    No_Statement
```

Here, any of these two parts will run. If the condition is true, then **only** the body of ***if*** statement will run (and else will NOT run). Otherwise, if the condition is false, then **only** the body of ***else*** statement will run (because the body of ***if*** statement will be skipped). See the following figure:



For example,

```
int marks;
cout << "Enter your marks: ";
cin >> marks;

if(marks >= 60) {
    cout << "Congrats! You have passed";
}
else {
    cout << "Sorry! You have failed";
}
```

Output 1:

```
Enter your marks: 61
Congrats! You have passed
```

Output 2:

```
Enter your marks: 59
Sorry! You have failed
```

Exercise

1. Write a program to accept two integers and print whether they are equal or not.
2. Write a program that should ask the user to input his/her percentage. Then, it should tell the grade according to his/her percentage. Details/rules of the grades are given below:

Percentage	Grade
90 (or more)	A1
80 to 89	A
70 to 79	B
60 to 69	C
Less than 60	F

3. Extend the program written in the previous lab's exercise, where you asked the user to input his/her marks for five subjects and calculated his/her obtained marks and percentage. So, after calculating the percentage, you should also tell his/her grade according to the rules given in the previous task's table.
4. Write a program that asks the user to input an integer value. Then, it should tell whether that number is odd or even. **Note/Hint**: A number is considered to be an **even** number if it is fully divided by 2. Otherwise, it is considered to be an **odd** number.
5. Write a program that prompts the user to input a number. The program should then output the number and a message saying whether the number is positive, negative, or zero.
6. Write a program that asks the use to input any month number in integer (like 1, 2, …, 12) and display the respective month name in the word (like January, February, …, December).
7. Write a C++ program that asks the user to input two number and tell
    a. if the first number is greater than the second number
    b. if the second number is greater than the first one
    c. or both numbers are equal

8. Write a C++ program that asks a user to input three numbers and tell which one is the largest of them. See the sample outputs below:

Case 1

```
Enter first number: 15
Enter second number: 45
Enter third number: 42
The second number is the greatest
```

Case 2

```
Enter first number: 20
Enter second number: 15
Enter third number: 18
The first number is the greatest
```

Case 3

```
Enter first number: 23
Enter second number: 12
Enter third number: 34
The third number is the greatest
```

9. Write a C++ program that asks a user to input two numbers and then the operator (+ or – or * or / etc.) to show which operation is to be performed on those two numbers. Like shown below:

Case 1

```
Enter first number: 12
Enter second number: 6
Now enter any of the operators (+, -, *, /): +
Operator was plus and the result is 18
```

Case 2

```
Enter first number: 12
Enter second number: 8
Now enter any of the operators (+, -, *, /): *
Operator was multplication and the result is 96
```

Case 3

```
Enter first number: 10
Enter second number: 5
Now enter any of the operators (+, -, *, /): $
Invalid operator
```

10. Write a program to calculate and print the Electricity bill of a given customer. The customer ID, name and unit consumed by the user should be taken input from the keyboard. Then, display the total amount to be paid by the customer. The unit charges are to be charged as per the rules given below:

Number of units consumed	Charge per unit
1 to 199	5 Rs.
200 to 399	10 Rs.
400 to 599	15 Rs.
600 or more	20 Rs.