NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES ISLAMABAD

PROGRAMMING FUNDAMENTALS LAB FALL 2019

Lab Manual 05
Control Structures (if else)

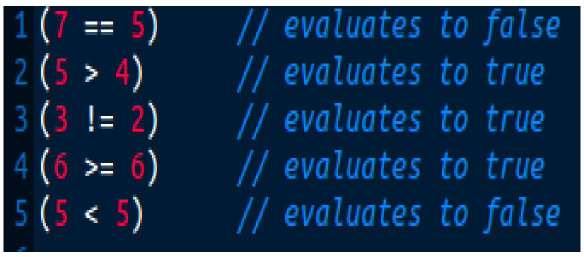
1 LOGICAL EXPRESSIONS

Relational and comparison operators

Two expressions can be compared using relational and equality operators. For example, to know if two values are equal or if one is greater than the other. C++ supports six comparison operators **==, !=, >, <, >=, <= .** Comparison operators always use two operands and result of such an operation is either true or false (i.e., a Boolean value 1 or 0).
The relational operators in C++ are:

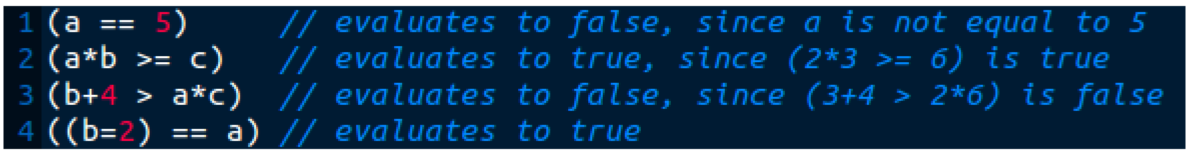
operator	description
==	Equal to
!=	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Examples:



```
(7 == 5)     // evaluates to false
(5 > 4)      // evaluates to true
(3 != 2)     // evaluates to true
(6 >= 6)     // evaluates to true
(5 < 5)      // evaluates to false
```

It's not just numeric constants that can be compared, but just any value, including variables. Suppose that a=2, b=3 and c=6, then:



```
(a == 5)     // evaluates to false, since a is not equal to 5
(a*b >= c)   // evaluates to true, since (2*3 >= 6) is true
(b+4 > a*c)  // evaluates to false, since (3+4 > 2*6) is false
((b=2) == a) // evaluates to true
```

Note: The assignment operator (operator =, with one equal sign) is not the same as the equality comparison operator (operator ==, with two equal signs); the first one

(=) assigns the value on the right-hand to the variable on its left, while the other (==) compares whether the values on both sides of the operator are equal. Therefore, in the last expression ((b=2) == a), we first assigned the value 2 to b and then we compared it to a (that also stores the value 2), yielding true.

Logical operators (!, & & , | |)
The operator ! is the C++ operator for the Boolean operation NOT. It has only one operand, to its right, and inverts it, producing false if its operand is true, and true if its operand is false. Basically, it returns the opposite Boolean value of evaluating its operand.
For example:

```
!(5 == 5)   // evaluates to false because the expression at its right (5 == 5) is true
!(6 <= 4)   // evaluates to true because (6 <= 4) would be false
!true       // evaluates to false
!false      // evaluates to true
```

The logical operators & & and | | are used when evaluating two expressions to obtain a single relational result. The operator & & corresponds to the Boolean logical operation AND, which yields true if both its operands are true, and false otherwise. The following panel shows the result of operator & & evaluating the expression a& & b:

&& OPERATOR (and)		
a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

The operator | | corresponds to the Boolean logical operation OR, which yields true if either of its operands is true, thus being false only when both operands are false. Here are the possible results of a| | b:

|| OPERATOR (or)		
a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

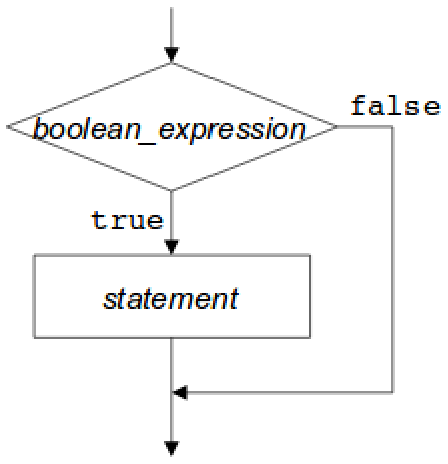
```
( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

2 Control Structures

If statement
The if keyword is used to execute a statement or block, if, and only if, a condition is fulfilled. Its syntax is:

```
if (boolean_expression)
    statement
```

The if Flowchart



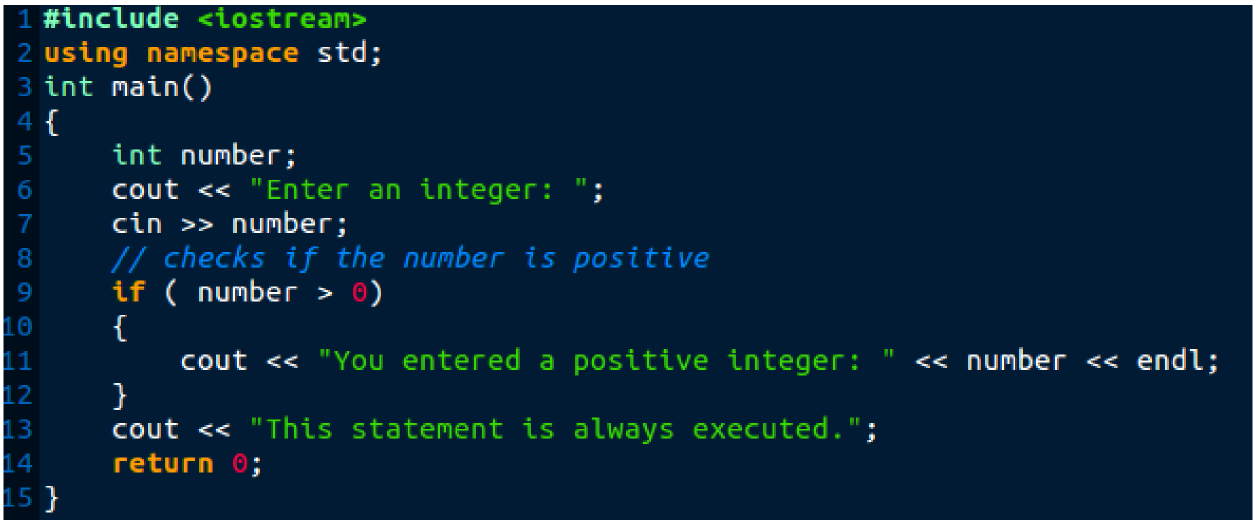
Here, the condition (boolean_expression) is an expression that is being evaluated. If this condition is true, statement is executed. If it is false, the statement is not executed (it is simply ignored), and the program continues right after the entire selection statement. For example, the following code fragment prints the message (x is 100), only if the value stored in the x variable is indeed 100:

```cpp
if (x == 100)
  cout << "x is 100";
```

If x is not exactly 100, this statement is ignored, and nothing is printed.
If you want to include more than a single statement to be executed when the condition is fulfilled, these statements shall be enclosed in braces ({ }), forming a block:

```cpp
if (x == 100)
{
   cout << "x is ";
   cout << x;
}
```

Example

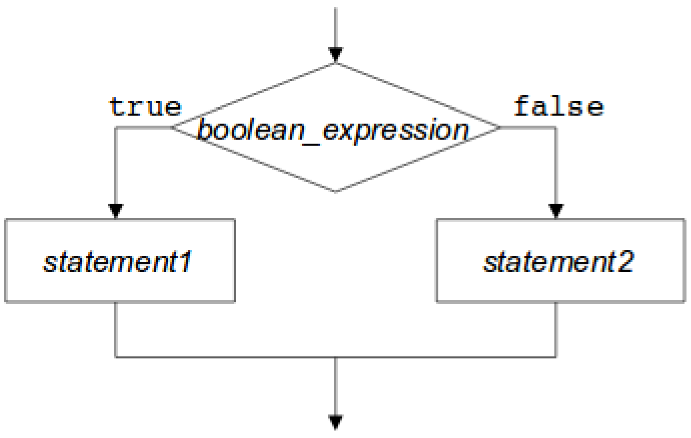


```cpp
#include <iostream>
using namespace std;
int main()
{
    int number;
    cout << "Enter an integer: ";
    cin >> number;
    // checks if the number is positive
    if ( number > 0)
    {
        cout << "You entered a positive integer: " << number << endl;
    }
    cout << "This statement is always executed.";
    return 0;
}
```

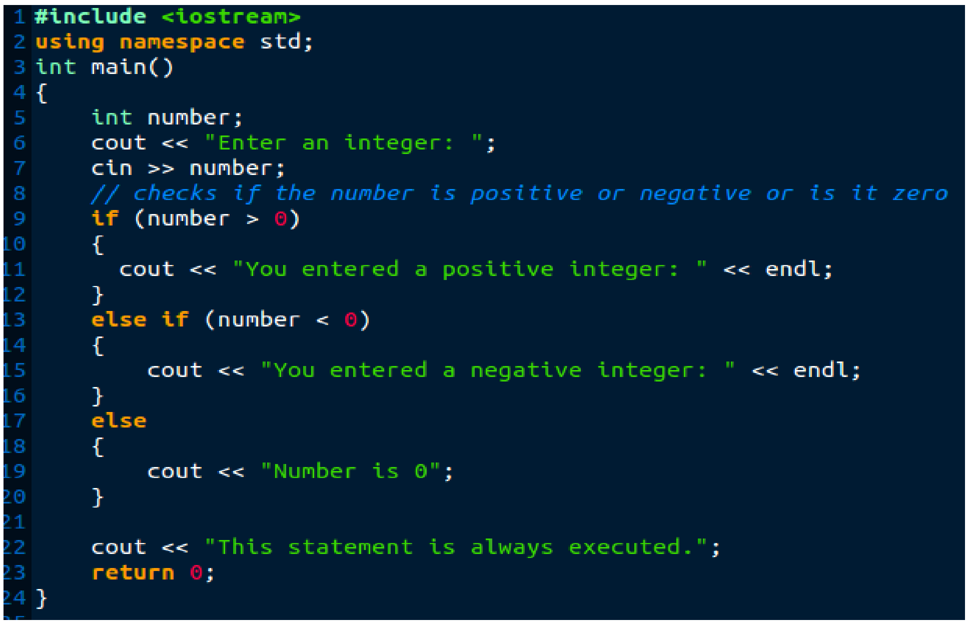
If else statement
If we want to choose between two alternative we use the if/else statement:

```
if (boolean_expression)
    statement1
else
    statement2
```

The if else Flowchart



Example:



```cpp
#include <iostream>
using namespace std;
int main()
{
    int number;
    cout << "Enter an integer: ";
    cin >> number;
    // checks if the number is positive or negative or is it zero
    if (number > 0)
    {
      cout << "You entered a positive integer: " << endl;
    }
    else if (number < 0)
    {
        cout << "You entered a negative integer: " << endl;
    }
    else
    {
        cout << "Number is 0";
    }

    cout << "This statement is always executed.";
    return 0;
}
```

<u>Lab Tasks</u>

Problem 01

Write a program that takes mileage per gallon of car from user, if mileage is equal or greater than 40 km per gallon then print "Your car has got excellent mileage" if less then print "your car has got poor mileage". Use only if statement

Problem 02

Take two integers input from the user. Check whether the first number is completely divisible by the second number. If yes, then print the following,
The number a is completely divisible by b
However, if the number is not completely divisible, then print:
The number is a not completely divisible by b

Problem 03

Write a program to take an integer input from the user between 1 and 99999 and Display how many digit number the user entered.
Example:
if user enters 10 the program should display “You have Entered a 2 Digit Number”
if the user enters 978 the program should display “You have Entered a 3 Digit Number” and so on.
if the user enters any other number Display “Invalid Input”.

Problem 04

Write a program to determine which character is entered by the user from the keyboard i.e., an uppercase character, a lowercase character, a digit or a special symbol.
Hint: use **char** data type i.e. char variable='a'; or char variable= '% ';

Problem 05

Write a program to take "**Age**" **.** as input from the user and Display the "**Stage Description**" as given below.

0–2 "**Baby/infant**"
3-12 "**Child**"
13-17 "**Teenager**"
18-35 "**Young Adult**"
36-45 "**Adult**"
45-65 "**Middle Age**"
65-75 "**Senior**"
75+ "**Elderly**"

Problem 06

In this program, we will display a menu to the user. The user can choose any option from the given menu. The menu would be something like this:

Welcome to the World of Control Structures

Press (A) to add two integers
Press (S) to subtract two integers
Press (M) to multiply two integers
Press (E) to exit the program

Please choose one of the above options:

The above menu is pretty much self explanatory. If the user inputs 'a' or 'A', you will give him the option to input two integers. Once he/she has done that, you will display the sum of the two numbers. If the user presses 'e' or 'E', the program should exit. If the user inputs anything else, you should display an error message saying "You have input an invalid command" .

Submission Instructions:

1. Save all **.cpp** files with your roll no and task number
   e.g. i19XXXX_Task05.cpp
2. Now create a new folder with name *ROLLNO_LAB05* **e.g. i19XXXX_LAB05**
3. Move all of your .cpp files to this newly created directory and compress it into **.zip file**.
4. Now you have to submit this zipped file on Slate.

THE END