National University of computer and emerging sciences

LAB MANUAL 5

CL118-PROGRAMMING FUNDAMENTALS

BSCS-FALL-2020

SCHOOL OF COMPUTING FAST-NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES, ISLAMABAD CAMPUS

National University of computer and emerging sciences

LAB 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 is 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 (!, & & **,** j j **)**

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:

```
1 !(5 == 5)   // evaluates to false because the expression at its right (5 == 5) is true
2 !(6 <= 4)   // evaluates to true because (6 <= 4) would be false
3 !true       // evaluates to false
4 !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

```
1 ( (5 == 5) && (3 > 6) )  // evaluates to false ( true && false )
2 ( (5 == 5) || (3 > 6) )  // evaluates to true ( true || false )
```

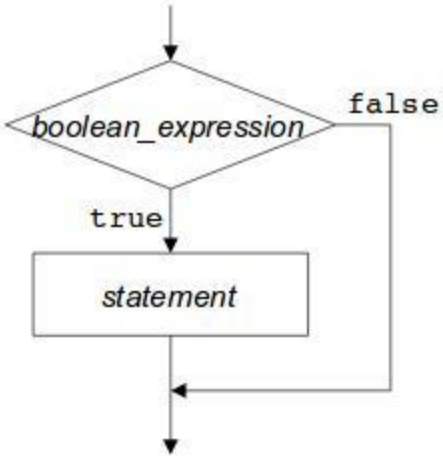
2 CONTROL STRUCTURES

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:

```
1 if (x == 100)
2   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 (f g), forming a block:

```
1 if (x == 100)
2 {
3    cout << "x is ";
4    cout << x;
5 }
```

National University of computer and emerging sciences

TASKS

Problem 1:
Write a program that reads two integers and outputs the largest.

Problem 2:
Write a program that reads three integers and outputs the smallest.

Problem 3:
Write a program that reads the user's age and then outputs "You are a child." if the age < 18, "You are an adult." if age < 65, and "You are a senior citizen." if age >= 65.

Problem 4:
Write a program that input an integer as an argument and returns True if the argument is an **even number**

Problem 5:
Write a program that reads the score of a student in a subject and displays his grade according to the following criteria:

Score	Grade
>= 90	A+
80 – 89	A
70 – 79	B
60 – 69	C
50 – 59	D
< 50	F