• Relational expression/ relational operations and its priorities/ formulate Relational expression • Logical expression/ logical operation and its priorities/ formulate Logical expression • Compound expression/ priorities table of public operations/ deferent examples	الرابع

Relational operators and relational expressions

"A relational operator with constants and variables makes relational expression or An expressions in which relational operators are use is called relational expression."

Points about relational operators

1.Relational operators are use to compare values.

- 2.All the expressions evaluates from left to right.
- 3.There are six relational operators in C++ programming (>,< ,>=,<=,==,!=).
- 4.These operators evaluates results true or false.
- 5.False statement represent by 0 and True statement represent by 1.
- 6.These operators evaluate at statement level and has no preference.

Operator	Meaning
>	Greater than
<	Less than
>=	Greater than equal than
<=	Less than equal than
!=	not equal to
= =	Equal to (conditional operator)

Logical Expression and Logical Operators
Logical operators

- 1.There are three logical operators And(&&),or(||) these two both are binary operator and not(!) is u nary operator.
- 2.More than one relation expression are combine by using logical operators.
- 3.The expression will evaluate from left to right if more than one relation expression are use.

And operator (&&)

- It produces true result if all the expression or conditions are true.
- It produces false if any one expression is false.
- Below table shows evaluation method of and operator.
- 1 represent True 0 represent false.

Example to understanding of And (**&&**) operator.
a=10,b=5

Exp-1	Exp-2	Result
1	1	1

Exp-1	Exp-2	Result
(a>b) it evaluates 1	(b<a) it will evaluate 1	1
(a>b) it evaluates 1	(b>a) it will evaluate 0	0
(a<b) it evaluates 0	(b<a) it will evaluate 1	0
(a<b) it evaluates 0	(b>a) it will evaluate 0	0

Or operator(||)

- 1.It produces true if any expression is true.
- 2.It produces false if all the conditions are false.
- Below table shows evaluation method of Or(||) operator.

Exp-1	Exp-2	Result
1	1	1
1	0	1
0	1	1
0	0	0

Example to understanding of And (||) operator.
a=10,b=5,

Exp-1	Exp-2	Result
(a>b) it evaluates 1	(b<a) it will evaluate 1	1
(a>b) it evaluates 1	(b>a) it will evaluate 0	1
(a<b) it evaluates 0	(b<a) it will evaluate 1	1
(a<b) it evaluates 0	(b>a) it will evaluate 0	0

Not operator(!)

- 1.If expression provides true result it convert it into false.
- 2.If expression provides false result it convert it into true.

Evaluation method.

Result	**! Result**
1	0
0	1

Example to understanding of And (!) operator.

a=10,b=5

Result	! Result
(a>b) it will evaluates 1	0
(a<b) it will evaluate 0	1

Logical Operators

There are following logical operators supported by C++ language.

Assume variable A holds 1 and variable B holds 0, then −

Show Examples

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	!(A && B) is true.

Assume if A = 60; and B = 13; now in binary format they will be as follows −

A = 0011 1100

B = 0000 1101

A&B = 0000 1100

A|B = 0011 1101

A^B = 0011 0001

~A = 1100 0011

The Bitwise operators supported by C++ language are listed in the following table. Assume variable A holds 60 and variable B holds 13, then –

Operator	Description	Example
&	Binary AND Operator copies a bit to the result if it exists in both operands.	(A & B) will give 12 which is 0000 1100
|	Binary OR Operator copies a bit if it exists in either operand.	(A | B) will give 61 which is 0011 1101
^	Binary XOR Operator copies the bit if it is set in one operand but not both.	(A ^ B) will give 49 which is 0011 0001
~	Binary Ones Complement Operator is unary and has the effect of 'flipping' bits.	(~A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number.
<<	Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand.	A << 2 will give 240 which is 1111 0000
>>	Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.	A >> 2 will give 15 which is 0000 1111

Show Example:

Try the following example to understand all the logical operators available in C++.

Copy and paste the following C++ program in test.cpp file and compile and run this program.

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

main() {
   int a = 5;
   int b = 20;
   int c ;

   if(a && b) {
      cout << "Line 1 - Condition is true"<< endl ;
   }

   if(a || b) {
      cout << "Line 2 - Condition is true"<< endl ;
   }

   /* Let's change the values of  a and b */
   a = 0;
   b = 10;

   if(a && b) {
      cout << "Line 3 - Condition is true"<< endl ;
   } else {
      cout << "Line 4 - Condition is not true"<< endl ;
   }

   if(!(a && b)) {
      cout << "Line 5 - Condition is true"<< endl ;
   }

   return 0;
}
```

When the above code is compiled and executed, it produces the following result −

```
Line 1 - Condition is true
Line 2 - Condition is true
Line 4 - Condition is not true
Line 5 - Condition is true
```