15. Logical Operators

- Shivam Malhotra

Find Output

```
#include <iostream.h>
int main()
{
        int x, y;
        x = (( y = 2) + 1);
        cout << x << endl << y;
        return 0;
}
```

Find Output

```
#include <iostream.h>
int main()
{
        int x, y;
        x = (( y = 2) + 1);
        cout << x << endl << y;
        return 0;
}
```

- First assign value 2 to y
- The overall value of expression (y = 2) is 2
- Next, the value 3 is assigned to x

Find Output

```
#include <iostream.h>
int main()
{
        int x, y;
        x = (( y = 2) + 1);
        cout << x << endl << y;
        return 0;
}
```

Final Output :

3

2

Logical Operators

Logical operators are used combine multiple expressions.

There are 3 logical operators

- Logical OR : *(expr1)* **||** *(expr2)*
- Logical AND : *(expr1)* **&&** *(expr2)*
- Logical NOT : **!***(expr)*

The value of logical expression is 0 if false and 1 if true.

Logical OR

- Logical OR is a binary operator which takes two expressions as its operands

(expression1) **||** *(expression2)*

Operation : Computes the two expressions

Value : 1 (true) if **either** of the expression is true, else 0 (false)

Expression	Value
`(4 == 3) || (9 <= 9)`	1
`(2 <= 1) || (1 != 1)`	0

Logical AND

- Logical AND is also a binary operator which takes two expressions as its operands

(expression1) **&&** *(expression2)*

Operation : Computes the two expressions

Value : 1 (true) if **both** of the expressions are true, else 0 (false)

Expression	Value
`(4 == 3) && (9 <= 9)`	0
`(2 >= 1) && (1 != 2)`	1

Logical NOT

- Logical NOT is a unary operator which takes only one expression as the operand

! (expression1)

Operation : Computes the expression

Value : 1 (true) if the expression is false, else 0 (false)

Expression	Value
`!(9 <= 9)`	0
`!(2 <= 1)`	1

Examples :

Let us use logical and relational operators to write expressions for the mentioned scenarios :

1. Check if the number is even.

(Assume that we have variable "*number*" storing the value)

```
((number%2) == 0)
```

2. Check if the weight is greater than 80 kg

(Assume that we have variable "*weight*" storing the weight in kilograms)

```
(weight > 80)
```

Examples :

Let us use logical and relational operators to write expressions for the mentioned scenarios :

3. Check if age is in the range 20-30 (20 and 30 included).

(Assume that we have variable "*age*")

```
((age >= 20) && (age <= 30))
```

Note that we cannot write it as : (20 <= age <= 30) 

You can try checking the value of this expression for value of "*age*" as 50.

Examples :

Let us use logical and relational operators to write expressions for the mentioned scenarios :

4. Check if either the age is less than or equal to 24 or salary is greater than 1000.

(Assume that we have variables "*age*" and "salary")

```
((age <= 24) || (salary > 1000))
```

It is important to use brackets () properly to clearly indicate the operands for each operator.

Comma Operator

- Comma operator is a binary operator which takes two expressions as its operands

(expression1) , *(expression2)*

Operation : Computes *expression1* first and then *expression2*

Value : Same as the value of *expression2*

Expression	Value
`(x = 5, x + 2)`	7
`(y = 5, y = 3)`	3

What’s ahead?

In the next video, we will study about conditional (ternary) operator