Logical operators which are used to couple the conditions, can also be used in statements.

Consider the following program:

```
#include <stdio.h>
int main()
{
        int i = 5, j = 4, k = -1, a;
        a = ++i && ++j || ++k;
        printf("%d %d %d %d", a, i, j, k);
        return 0;
}
```

The question is how will the expression for 'a' will be evaluated ?

i.e. a = (++i && ++j) || ++k;

or a = ++i && (++j || ++k);

The confusion is created because the parenthesis are missing. So the solution is obtained from the precedence order of operators where priority of AND is more than OR. So the former one is the correct expression and AND will be evaluated first.

Output: **0 6 5 -1**

Here i and j are incremented by 1 but k remains same even though we have used ++k.

When the expression is executed, compiler gets a = 1 || ++k. Now compiler thinks whatever be the value of ++k, the result is for sure to be true, so why to evaluate ++k? Therefore, ++k is not evaluated and remains -1 only.

So in logical operators, expressions are evaluated only until the fate of the condition is decided. Once it is decided, the remaining expressions won't be evaluated at all. This is known as '**short – circuit evaluation**'.