Boolean Expressions

Often times when making a decision, you have to evaluate a number of different conditions, each of which may be TRUE or FALSE. For example,

- You can sleep in, **if** it is a Saturday **or** a Sunday.
- You want the buzzer in your car to sound **if** the keys are in the ignition **and** the door is open.
- You want the fire alarm in your house to sound **if** it senses heat **or** smoke.
- There is an election coming up. You are allowed to vote **if** you are a citizen **and** you are 18 **or** older.

In C# a condition, such as one of those above, is written as a Boolean expression. When evaluated, a Boolean expression is always either true or it is false. For example, we might ask

```
is a < b?
```

Either a is less than b (the expression is true) or it is not (the expression is false).

In C#, the symbol < is what is called a Relational Operator.

Relational Operators

The relational operators in C# are defined in the following table:

Operator	Description	Example	Meaning
==	equal to	a == b	Is a equal to b?
!=	not equal to	a != b	Is a not equal to b?
<	less than	a < b	Is a less than b?
>	greater	a > b	Is a greater than b?

	than		
<=	less than or equal to	a <= b	Is a less than or equal to b?
>=	greater than or equal to	a >= b	Is a greater than or equal to b?

Boolean Operators

In the examples above, we presented situations in which a combination of conditions had to be evaluated in order to make a decision.
Combinations of such conditions are formed with Boolean Operators.
Boolean operators take Boolean values (true or false) and yield a Boolean value.

Operator	Description	Example
!	Not	! (a < b)
&&	And	a < b && x < y
||	Or	a > b || x < y

The definition of Boolean operators is often expressed in what is called a Truth Table.

NOT Truth Table

Expression	! Expression
true	false
false	true

AND Truth Table

Expression 1	Expression 2	Expression 1 && Expression 2
true	true	true

true	false	false
false	true	false
false	false	false

OR Truth Table

Expression 1	Expression 2	Expression 1 || Expression 2
true	true	true
true	false	true
false	true	true
false	false	false

Short Circuit Evaluation

When Boolean expressions are combined with either the && operator or the || operator, C# uses what is called Short Circuit Evaluation. Consider, for example, the expression

```
(x > y) || (x == 5)
```

The computer first evaluates the left hand portion of this expression,

```
(x > y).
```

If this part is true, then the OR of the two parts must also be true. Consult the truth tables above if you are not convinced of this. Since the computer knows that the complete expression must be true, it does not bother to evaluate the right hand part

```
(x == 5).
```

If, on the other hand,

```
(x > y)
```

were false, then we don't know if the OR of the two parts is true until we know whether or not

```
(x == 5)
```

is true. So, in this case the computer must evaluate both parts of the expression.

Now, consider, the expression

```
(x > y) && (x == 5)
```

The computer first evaluates the left hand portion of this expression,

```
(x > y).
```

If this part is false, then the AND of the two parts must also be false. Consult the truth tables above if you are not convinced of this. Since the computer knows that the complete expression must be false, it does not bother to evaluate the right hand part

```
(x == 5).
```

If, on the other hand,

```
(x > y)
```

were true, then we don't know if the AND of the two parts is true until we know whether or not

```
(x == 5)
```

is true. So, in this case the computer must evaluate both parts of the expression.