Comparison Operators

This lesson introduces different comparison operators such as ==,!=,>,< etc that can be used in C++ and which data types they can be applied to.

We'll cover the following

As the name implies, *conditional statements* specify whether another *statement* or *block* of statements should be executed or not. These are often called **"selection constructs"**. The two general types are:

- "if...then"
- the "switch...case" construct

Note that there is no looping involved here, but that conditionals are involved in loops.

Comparison Operators Syntax

The conditions tested are specified using **comparison operators**. These *operators* cause the **immediate** statement in which they are contained to return a **Boolean** value of either `true` or `false`.

> **Note:** In certain circumstances they may evaluate to `0` or `1`; be careful combining *conditional* statements with arithmetic.

The following comparison operators are available:

- **Equality:** `==`, or **Inequality:** `!=` of any primitive data type (`int`, `char`, `float`, `bool`, etc.) These are *binary* operators (take **two** operands) and are specified using *infix* notation (which means that the operator goes in between the two operands).

- **Greater-than:** `>`, **Greater than or equal to:** `>=`, **Less-than:** `<` and **Less than or equal to:** `<=` are also *binary* operators using *infix* notation. Use only with *numeric* data types; there are specific functions for comparing other data types.
- **Negation:** `!` is a *unary operator*, and *prefixes* the operand.

Examples

Statement	Result
5 == 5	true
7 != 5	true
`a` == `b`	false
6 > 9	false
4 <= 4	true
! `true`	false
`true` !	syntax error

Now that you're familiar with the *comparison* operators let's look at the *conditional* statements in the next lesson.