Basic Operators

This lesson will introduce you to types of operators in C# such as relational, unary, binary etc

WE'LL COVER THE FOLLOWING

Until now we've used hardcoded values to output information to the screen. Now we'll use math on those values to force the computer to perform some *mathematical* grunt work.

A computer uses the same laws for math as we do. The **C#** language follows the rule of PEMDAS hence the *operations* on the **left** are executed before the operations to the **right**.

Table of Operators

The following table describes the allowable *operators*, their *precedence*, and *associativity*.

Category (by precedence)	Operator(s)	Associativity
Primary	`x.y` `f(x)` `a[x]` `x++` `x–` new typeof default checked unchecked delegate	**left**
Unary	`+` `–` `!` `~` `++x` `–x` `(T)x`	**right**
Multiplicative	`*` `/` `%`	**left**

Additive	`+` `-`	**left**
Shift	`<<` `>>`	**left**
Relational	`<` `>` `<=` `>=` `is` `as`	**left**
Equality	`==` `!=`	**right**
Logical AND	`&`	**left**
Logical XOR	`^`	**left**
Logical OR	`|`	**left**
Conditional AND	`&&`	**left**
Conditional OR	`||`	**left**
Null Coalescing	`??`	**left**
Ternary	`?:`	**right**
Assignment	`=` `*=` `/=` `%=` `+=` `-=` `<<=` `>>=` `&=` `^=` `|=` `=>`	**right**

> **Note: Left** *associativity* means that *operations* are evaluated from **left** to **right**. **Right** *associativity* means all *operations* occur from **right** to **left**, such as *assignment operators* where everything to the **right** is evaluated before the result is placed into the variable on the **left**.

Now that you've taken a look at the basic *operators* in **C#** lets take a look at **unary** *operators* in detail in the next lesson.