Statements and Expressions

A C# statement contains one or more expressions ending with a **;**

An expression contains: **value operator value**

variable operator variable

An expression always produces a result.

Expressions in a statement are evaluated left to right based on Precedence (Order or Operations)

Order of operatations (TL;DR) process left to right:

1. Stuff in **()**

2. **++** (increment)
   -- (decrement)
   - (make negative)
   + (make positive)

3. ***** (multiply)
   / (divide)
   % (modulus - remainder of integer)

4. **+** (add)
   - (subtract)

5. All other operators

6. Assignment is always last

```
int aNum = 1 + 2 - 6 + 5;  // This is a statement with 3 expressions

What C# does:

  1. Evaluate the expression 1 + 2 (result 3)
  2. Evaluate the expression result-from-#1 - 6 (result -3)
  3. Evaluate the expression result-from-#2 + 5 (result +2)
  4. Assign the result from-from-#3 to aNum
  5. aNum contains the value-from-#4 (+2)

int aNum = 1 + 2 * 6 + 5;  // This is a statement with 3 expressions

What C# does:

  1. Evaluate the expression 2 * 6 (result 12)
  2. Evaluate the expression 1 + result-from-#1 (result 13)
  3. Evaluate the expression result-from-#2 + 5 (result 18)
  4. Assign the result from-from-#3 to aNum
  5. aNum contains the value-from-#4 (18)


int aNumber = int.Parse(Console.Readline());

 1. Run the Console.Readline() - result is a string
 2. Give the string-from-#1 to int.Parse() - result is numeric value of string
    3. Assign result-from-#2 to aNumber
```

The context in which something is used helps identify what it is:

```
  1 + 2  - add
  +1    - make positive
  "Tom" + "Jerry" - concatenate two strings
```

In English: How do you pronounce "read" - reed or red

What is offense? - Football group or ticks someone off