**Data Type Selection:**

- If performance is a factor, choosing a data type that takes up less memory will result in a faster application  
- Consider if your application will require a whole number or a decimal in it

**Integer:**

- A whole value integer like 4, 100, or -2349 used to count the individual units of things

A screenshot of a computer

Description automatically generated

**Decimals:**

- To capture a decimal value you can use *double, float,* or *decimal*- *double* is the best option as it is more precise that *float* (15-16 significant digits compared to *float* 7 significant digits) but faster than *decimal*- For financial applications use *decimal* as it is the most precise (28-29 significant digits)

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

- Don’t forget the m character after the number when creating a decimal! This character tells C# that we’re defining a decimal and not a double. Similarly, when creating a float, the number must be followed by the character f:  
- *double* can use a trailing d but it is not required

A screenshot of a computer

Description automatically generated A screenshot of a computer

Description automatically generated

**Arithmetic Operators:**

- When using two similar data types the resulting value will always be the same data type  
- When using different data types the resulting value will always be the type that is *more precise*

*A screenshot of a computer

Description automatically generated A screen shot of a computer code

Description automatically generated*

**Order of Operations:**

- C# follows a specific order of operations:   
- When assessing operators without parentheses, it will do so left to right

A black screen with white text

Description automatically generated A black background with white text

Description automatically generated  
A computer screen shot of text

Description automatically generated

**Operator Shortcuts:**

- Can update a variable by modifying it using an arithmetic expression and then resaving it to the same variable name  
A computer screen with white and orange text

Description automatically generated with medium confidence  
  
- ++ is used as shorthand to add +1 to a variable (**increment operator)  
-** -- is used as shorthand to subtract -1 from a variable (decrement operator)  
A black background with white text

Description automatically generated

- To increment by a value other than 1, use the += operator  
- Can also use -= to decrement by 3  
- Also applies to \*= and /=  
A computer screen shot

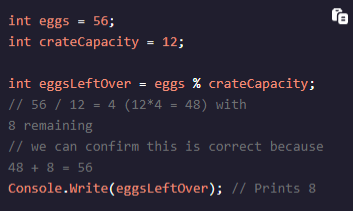
Description automatically generated A black background with white text

Description automatically generated

**Modulo (%):**

- Returns a *remainder*, the value left over when we divide a number by another number  
- Useful when we want to know if a number fits into another number   
- Can also be used to check if a number is odd or even

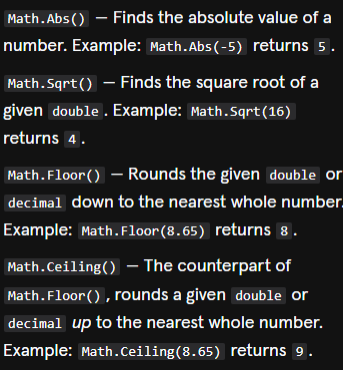
A computer screen shot of white text

Description automatically generated

A screenshot of a computer code

Description automatically generated

**Built-In Methods:**

- C# has several built-in methods to manipulate data and perform more complex mathematical calculations  
- All methods will output a number that is the same type as the input or a more precise number  
  
 A black and white text on a black background

Description automatically generated

- Can combine multiple arithmetic methods by nesting them within each other  
- Order of operations matters here: Parenthesis, Multiplication, Addition

