Cheatsheets / **Beginner C#**

Learn C#: Numbers and Operators

`Console.ReadLine()`

The `Console.ReadLine()` method is used to get user input. The user input can be stored in a variable. This method can also be used to prompt the user to press `enter` on the keyboard.

```
Console.WriteLine("Enter your name: ");

string name = Console.ReadLine();
```

Comments

Comments are bits of text that are not executed. These lines can be used to leave notes and increase the readability of the program.

- Single line comments are created with two forward slashes `//` .
- Multi-line comments start with `/*` and end with `*/` . They are useful for commenting out large blocks of code.

```
// This is a single line comment

/* This is a multi-line comment
   and continues until the end
   of comment symbol is reached */
```

`Console.WriteLine()`

The `Console.WriteLine()` method is used to print text to the console. It can also be used to print other data types and values stored in variables.

```
Console.WriteLine("Hello, world!");

// Prints: Hello, world!
```

Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations on numerical values:

- `+` addition operator
- `-` subtraction operator
- `*` multiplication operator
- `/` division operator
- `%` modulo operator (returns the remainder)

```
int result;

result = 10 + 5;  // 15

result = 10 - 5;  // 5

result = 10 * 5;  // 50

result = 10 / 5;  // 2

result = 10 % 5;  // 0
```

String Interpolation in C#

String interpolation provides a more readable and convenient syntax to create formatted strings. It allows us to insert variable values and expressions in the middle of a string so that we don’t have to worry about punctuation or spaces.

```
int id = 100

// We can use an expression with a string interpolation.
string multipliedNumber = $"The multiplied ID is {id * 10}.";

Console.WriteLine(multipliedNumber);
// This code would output "The multiplied ID is 1000."
```

Built-in Math Methods

In C#, `Math` provides many built-in methods for performing advanced mathematical operations. Some common methods include:

- `Math.Abs()` – calculates the absolute value of a given number
- `Math.Sqrt()` – calculates the of a given number
- `Math.Floor()` – rounds the given number down to the nearest integer
- `Math.Min()` – takes 2 values of the same type and returns the value that is less

```
double x = -80;

double absValue = Math.Abs(x); // 80
double sqRoot = Math.Sqrt(absValue); // 8.94427190999916
double floored = Math.Floor(sqRoot); // 8
double smaller = Math.Min(x, floored); // -80
```

Operator Shortcuts

C# offers several shortcuts for condensing simple operations.
The addition (`+`) and subtraction (`-`) operators can be doubled to form the increment (`++`) and decrement (`--`) operators. `++` increases a variable's value by 1, and `--` lowers it by 1.
`+`, `-`, `*`, `/`, and `%` can all be combined with the equals sign (`=`) to form compound assignment operators, such as `+=`, the compound addition operator. These operators take a variable to the left and a value to the right, perform the specified arithmetic operation, and assign the result back into the variable.

```
int a = 0;

a++;     // a = 1
a += 1;  // a = 2
a *= 7;  // a = 14
a--;     // a = 13
a %= 7;  // a = 6
```

Print   Share ▼