codecademy

Data Types and Variables

Variables and Types

A variable is a way to store data in the computer's memory to be used later in the program. C# is a type-safe language, meaning that when variables are declared it is necessary to define their data type.
Declaring the types of variables allows the compiler to stop the program from being run when variables are used incorrectly, i.e, an `int` being used when a `string` is needed or vice versa.

```
string foo = "Hello";
string bar = "How are you?";
int x = 5;

Console.WriteLine(foo);
// Prints: Hello
```

`Math.Sqrt()`

`Math.Sqrt()` is a `Math` class method which is used to calculate the square root of the specified value.

```
double x = 81;

Console.Write(Math.Sqrt(x));

// Prints: 9
```

Arithmetic Operators

Arithmetic operators are used to modify 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
```

Unary Operator

Operators can be combined to create shorter statements and quickly modify existing variables. Two common examples:

- `++` operator increments a value.
- `--` operator decrements a value.

```
int a = 10;
a++;

Console.WriteLine(a);
// Prints: 11
```

Math.Pow()

`Math.Pow()` is a `Math` class method that is used to raise a number to a specified power. It returns a number of double type.

```
 double pow_ab = Math.Pow(6, 2);

Console.WriteLine(pow_ab);

// Prints: 36
```

.toUpper() in C#

In C#, `.ToUpper()` is a string method that converts every character in a string to uppercase. If a character does not have an uppercase equivalent, it remains unchanged. For example, special symbols remain unchanged.

```
string str2 = "This is C# Program
xsdd_$#%";

// string converted to Upper case
string upperstr2 = str2.ToUpper();

//upperstr2 contains "THIS IS C# PROGRAM
XSDD_$#%"
```

IndexOf() in C#

In C#, the `IndexOf()` method is a string method used to find the index position of a specified character in a string. The method returns -1 if the character isn't found.

```
string str = "Divyesh";

// Finding the index of character
// which is present in string and
// this will show the value 5
int index1 = str.IndexOf('s');

Console.WriteLine("The Index Value of
character 's' is " + index1);
//The Index Value of character 's' is 5
```

Bracket Notation

Strings contain characters. One way these char values can be accessed is with bracket notation. We can even store these chars in separate variables.

We access a specific character by using the square brackets on the string, putting the index position of the desired character between the brackets. For example, to get the first character, you can specify `variable[0]`. To get the last character, you can subtract one from the length of the string.

```
// Get values from this string.
string value = "Dot Net Perls";

//variable first contains letter D
char first = value[0];

//Second contains letter o
char second = value[1];

//last contains letter s
char last = value[value.Length - 1];
```

Escape Character Sequences in C#

In C#, an escape sequence refers to a combination of characters beginning with a back slash \ followed by letters or digits. It's used to make sure that the program reads certain characters as part of a string. For example, it can be used to include quotation marks within a string that you would like to print to console. Escape sequences can do other things using specific characters. `\n` is used to create a new line.

Substring() in C#

In C#, Substring() is a string method used to retrieve part of a string while keeping the original data intact. The substring that you retrieve can be stored in a variable for use elsewhere in your program.

```
string myString = "Divyesh";
string test1 = myString.Substring(2);
```

String Concatenation in C#

Concatenation is the process of appending one string to the end of another string. The simplest method of adding two strings in C# is using the + operator.

```
// Declare strings
string firstName = "Divyesh";
string lastName = "Goardnan";

// Concatenate two string variables
string name = firstName + " "
+ lastName;
Console.WriteLine(name);
//Ths code will output Divyesh Goardnan
```

.ToLower() in C#

In C#, `.ToLower()` is a string method that converts every character to lowercase. If a character does not have a lowercase equivalent, it remains unchanged. For example, special symbols remain unchanged.

```
string mixedCase = "This is a MIXED case string.";

// Call ToLower instance method, which returns a new copy.
string lower = mixedCase.ToLower();

//variable lower contains "this is a mixed case string."
```

String Length in C#

The string class has a Length property, which returns the number of characters in the string.

```
string a = "One example";
Console.WriteLine("LENGTH: " + a.Length);
// This code outputs 11
```

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."
```

String New-Line

The character combination `\n` represents a newline character when inside a C# `string`.
For example passing `"Hello\nWorld"` to `Console.WriteLine()` would print `Hello` and `World` on separate lines in the console.

```
Console.WriteLine("Hello\nWorld");

// The console output will look like:
// Hello
// World
```

`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: ");

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!
```