# Language Map for C#

## Variable Declaration

*Is this language strongly typed or dynamically typed? Provide an example of how variables are declared in this language.*

C# is a strongly typed language.

In C#, begin a variable declaraton with the C# keyword for the data type followed by the name of the variable. Follow the rules and conventions for naming variables. Variables may or may not be initialized when they are declared. For example, the following declares a string variable named *firstName* and initializes it to *"Bob":*

```
string firstName = "Bob";
```

## Data Types

*List all of the data types (and ranges) supported by this language.*

(w3schools.com)

| Data Type | Size | Description |
|---|---|---|
| bool | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter, surrounded by single quotes |
| string | 2 bytes per character | Stores a sequence of characters, surrounded by double quotes |

## Data Types (cont.)

C# supports the following predefined integral types (docs.microsoft.com):

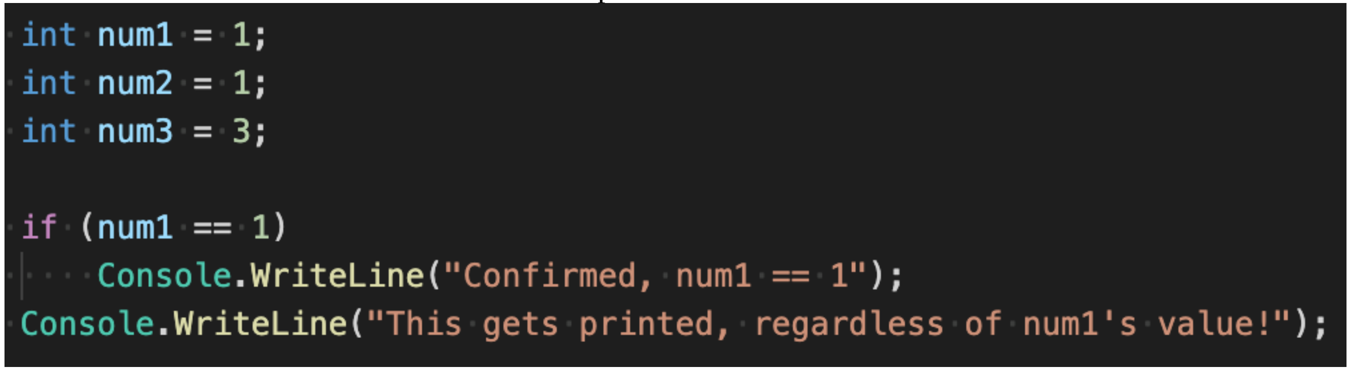
| C# type/ keyword | Range | Size |
|---|---|---|
| sbyte | -128 to 127 | Signed 8-bit integer |
| byte | 0 to 255 | Unsigned 8-bit integer |
| short | -32,768 to 32,767 | Signed 16-bit integer |
| ushort | 0 to 65,535 | Unsigned 16-bit integer |
| int | -2,147,483,648 to 2,147,483,647 | Signed 32-bit integer |
| uint | 0 to 4,294,967,295 | Unsigned 32-bit integer |
| long | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | Signed 64-bit integer |
| ulong | 0 to 18,446,744,073,709,551,615 | Unsigned 64-bit integer |
| nint | Depends on platform | Signed 32-bit or 64-bit integer |
| nuint | Depends on platform | Unsigned 32-bit or 64-bit integer |

C# supports the following predefined floating-point types (http://docs.microsoft.com/):

| C# type/keyword | Approximate range | Precision | Size |
|---|---|---|---|
| float | $\pm1.5 \times 10^{-45}$ to $\pm3.4 \times 10^{38}$ | ~6-9 digits | 4 bytes |
| double | $\pm5.0 \times 10^{-324}$ to $\pm1.7 \times 10^{308}$ | ~15-17 digits | 8 bytes |
| decimal | $\pm1.0 \times 10^{-28}$ to $\pm7.9228 \times 10^{28}$ | 28-29 digits | 16 bytes |

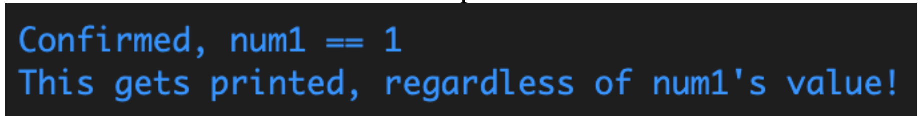
**Selection Structures**
*Provide examples of all selection structures supported by this language (if, if else, etc.)*

Simple if statement:



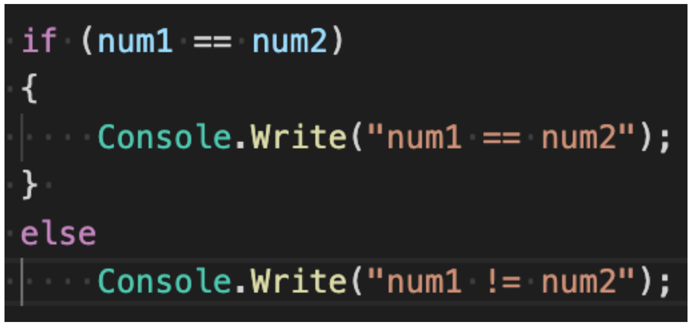
```
int num1 = 1;
int num2 = 1;
int num3 = 3;

if (num1 == 1)
    Console.WriteLine("Confirmed, num1 == 1");
Console.WriteLine("This gets printed, regardless of num1's value!");
```

Output:



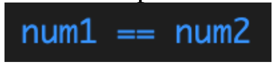
```
Confirmed, num1 == 1
This gets printed, regardless of num1's value!
```

If else statement:

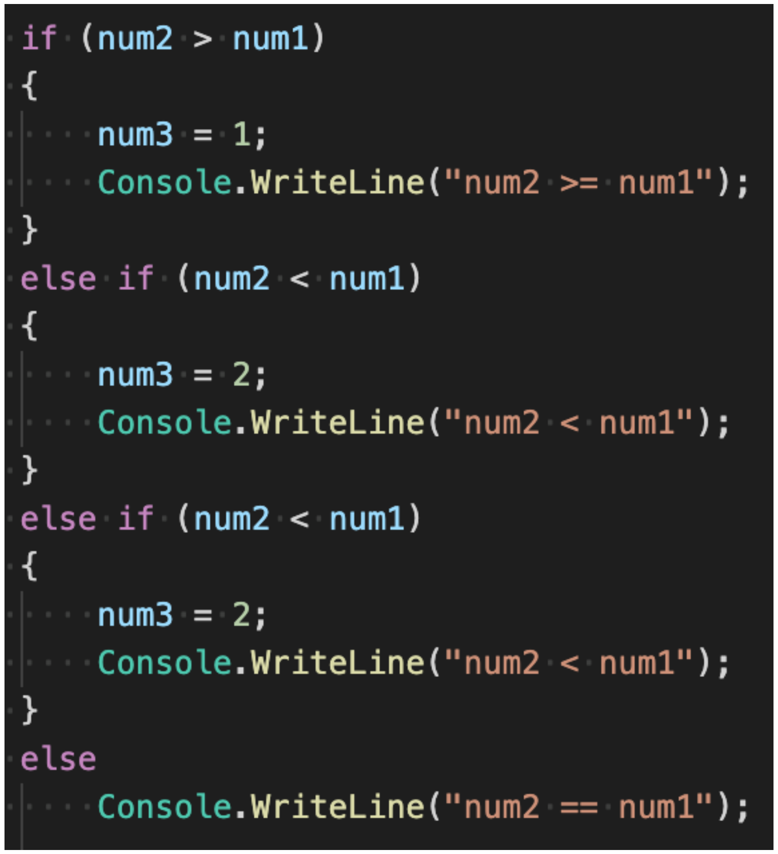
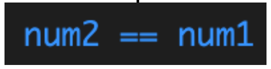


```
if (num1 == num2)
{
    Console.Write("num1 == num2");
}
else
    Console.Write("num1 != num2");
```

Output:

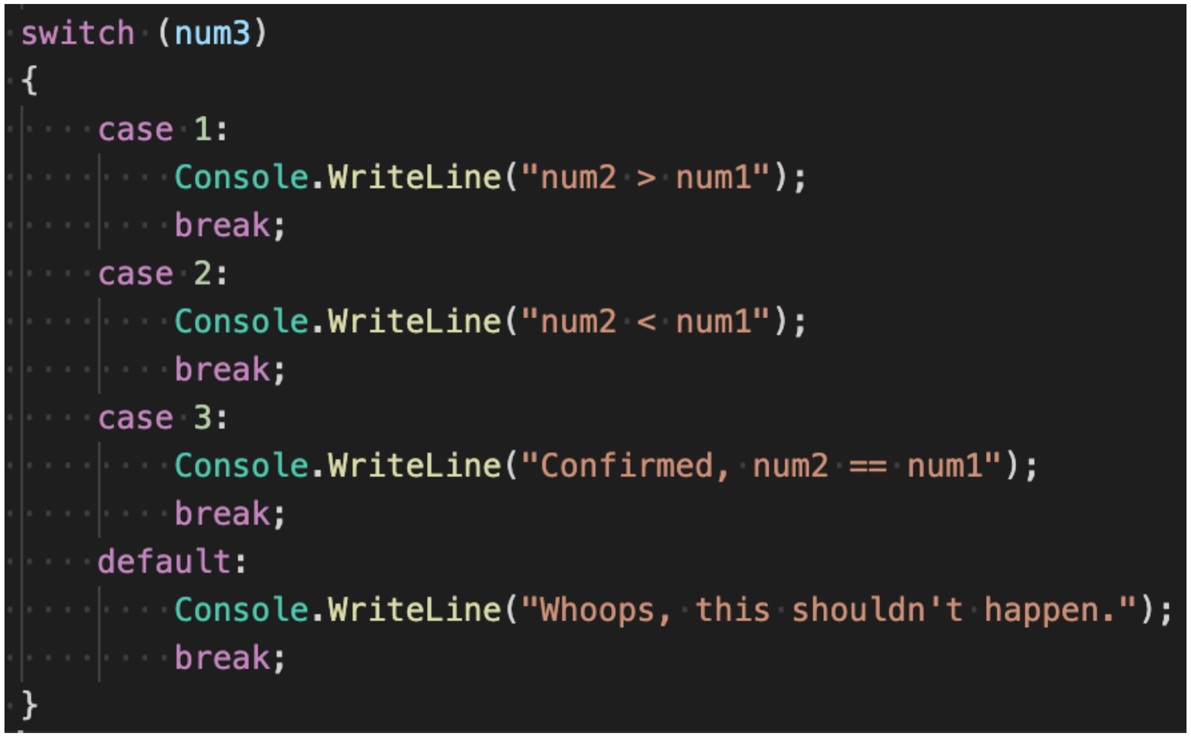


```
num1 == num2
```

| | |
|---|---|
| **Selection Structures** | If else if statement:  <pre>if (num2 > num1)<br>{<br>    num3 = 1;<br>    Console.WriteLine("num2 >= num1");<br>}<br>else if (num2 < num1)<br>{<br>    num3 = 2;<br>    Console.WriteLine("num2 < num1");<br>}<br>else if (num2 < num1)<br>{<br>    num3 = 2;<br>    Console.WriteLine("num2 < num1");<br>}<br>else<br>    Console.WriteLine("num2 == num1");</pre> Output:  <pre>num2 == num1</pre> |

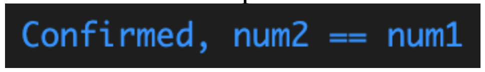
**Selection Structures (cont.)**

Switch statement:



```
switch (num3)
{
    case 1:
        Console.WriteLine("num2 > num1");
        break;
    case 2:
        Console.WriteLine("num2 < num1");
        break;
    case 3:
        Console.WriteLine("Confirmed, num2 == num1");
        break;
    default:
        Console.WriteLine("Whoops, this shouldn't happen.");
        break;
}
```

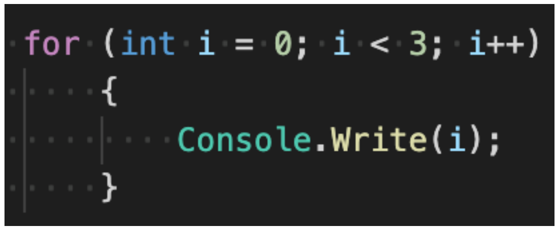
Output:



```
Confirmed, num2 == num1
```

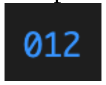
**Repetition Structures**
*Provide examples of all repetition structures supported by this language (loops, etc.)*

For statement:



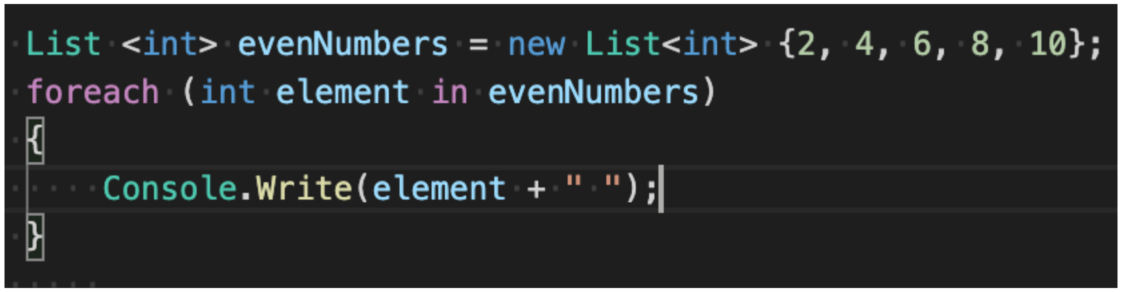
```
for (int i = 0; i < 3; i++)
    {
        Console.Write(i);
    }
```

Output:



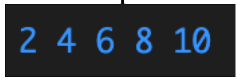
```
012
```

Foreach statement:



```
List <int> evenNumbers = new List<int> {2, 4, 6, 8, 10};
foreach (int element in evenNumbers)
{
    Console.Write(element + " ");
}
```

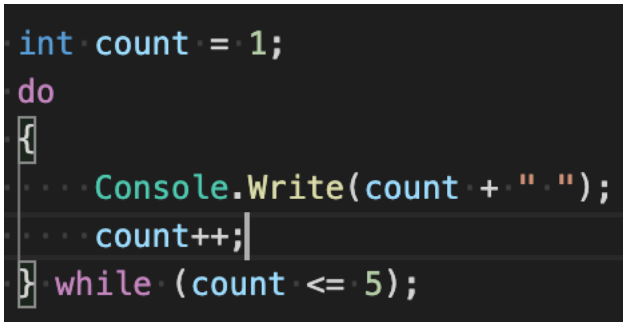
Output:



```
2 4 6 8 10
```

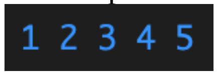
**Repetition Structures (cont.)**

Do statement:



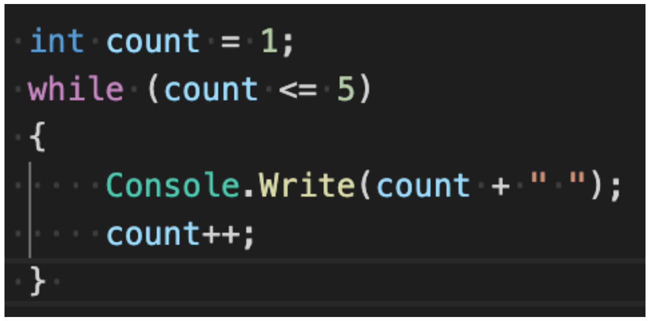
```
int count = 1;
do
{
    Console.Write(count + " ");
    count++;
} while (count <= 5);
```

Output:



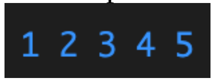
```
1 2 3 4 5
```

While statement:



```
int count = 1;
while (count <= 5)
{
    Console.Write(count + " ");
    count++;
}
```

Output:



```
1 2 3 4 5
```

| | |
|---|---|
| **Arrays**<br>*If this language supports arrays, provide an example of creating an array with a primitive data type (e.g. float, int, etc.)* | The following creates and initializes an array of strings named *firstNames*:<br>`string[] firstNames = { "Bob", "Larry", "Sue" };`<br><br>And the following creates an empty array of integers of length 3:<br>`int[] myNumbers = new int[3];` |
| **Data Structures**<br>*If this language provides a standard set of data structures, provide a list of the data structures and their Big-Oh complexity.* | Data structures in C# are categorized as collections.<br>https://docs.microsoft.com/en-us/dotnet/standard/collections/ |

| **Generic collection options** | **Non-generic collection options** | **Thread-safe or immutable collection options** |
|---|---|---|
| Dictionary<TKey,TValue> | Hashtable<br><br>(A collection of key/value pairs that are organized based on the hash code of the key.) | ConcurrentDictionary<TKey,TValue><br><br>ReadOnlyDictionary<TKey,TValue><br><br>ImmutableDictionary<TKey,TValue> |
| List<T> | Array<br><br>ArrayList | ImmutableList<T><br><br>ImmutableArray |
| Queue<T> | Queue | ConcurrentQueue<T><br><br>ImmutableQueue<T> |
| Stack<T> | Stack | ConcurrentStack<T><br><br>ImmutableStack<T> |
| LinkedList<T> | No recommendation | No recommendation |

**Data Structures (cont.)**

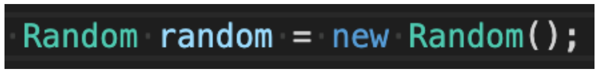
| | | |
|---|---|---|
| ObservableCollection<T> | No recommendation | No recommendation |
| SortedList<TKey,TValue> | SortedList | ImmutableSortedDictionary<TKey,TValue> |
| | | ImmutableSortedSet<T> |
| HashSet<T> | No recommendation | ImmutableHashSet<T> |
| SortedSet<T> | | ImmutableSortedSet<T> |

Algorithmic complexity of collections:

| **Mutable** | **Amortized** | **Worst Case** | **Immutable** | **Complexity** |
|---|---|---|---|---|
| `Stack<T>.Push` | O(1) | O(n) | `ImmutableStack<T>.Push` | O(1) |
| `Queue<T>.Enqueue` | O(1) | O(n) | `ImmutableQueue<T>.Enqueue` | O(1) |
| `List<T>.Add` | O(1) | O(n) | `ImmutableList<T>.Add` | O(log n) |
| `List<T>.Item[Int32]` | O(1) | O(1) | `ImmutableList<T>.Item[Int32]` | O(log n) |
| `List<T>.Enumerator` | O(n) | O(n) | `ImmutableList<T>.Enumerator` | O(n) |
| `HashSet<T>.Add`, lookup | O(1) | O(n) | `ImmutableHashSet<T>.Add` | O(log n) |
| `SortedSet<T>.Add` | O(log n) | O(n) | `ImmutableSortedSet<T>.Add` | O(log n) |
| `Dictionary<T>.Add` | O(1) | O(n) | `ImmutableDictionary<T>.Add` | O(log n) |
| `Dictionary<T>` lookup | O(1) | O(1) – or strictly O(n) | `ImmutableDictionary<T>` lookup | O(log n) |
| `SortedDictionary<T>.Add` | O(log n) | O(n log n) | `ImmutableSortedDictionary<T>.Add` | O(log n) |

**Objects**

*If this language support object-orientation, provide an example of how to create a simple object with a default constructor.*



```
Random random = new Random();
```

| | |
|---|---|
| **Runtime Environment** *What runtime environment does this language compile to? For example, Java compiles to the Java Virtual Machine. Do other languages also compile to this runtime?* | C# compiles to the .NET runtime.<br><br>Dot Net is a huge software framework, which contains 32 programming languages. Some of the prominent languages are C#, F#, and Visual Basic. The other programming languages are A Sharp (.NET), Boo, Cobra, Fantom, IronPython, Sharp, Oxygene, etc. to name a few.<br><br>https://www.clariontech.com/blog/top-5-things-you-should-know-about-.net-framework |
| **Libraries/ Frameworks** *What are the popular libraries or frameworks used by programmers for this language? List at least three (3).* | Top 10 C# libraries:<br><br>1. Newtonsoft.<br>2. NzbDrone, a personal video recorder (PVR) for newsgroup users, which climbed its way up from No. 17 in 2018.<br>3. Abp, ASP.NET Boilerplate, for Microsoft's framework designed for web applications.<br>4. dnSpy, a .NET debugger and assembly editor that moved up from No. 9 in 2018.<br>5. JetBrains' ReSharper, the Visual Studio extension for .NET.<br>6. Cake, a cross-platform build automation system with a C# DSL.<br>7. DotNetNuke, a web content management system that jumped from No. 14 in 2018 to No. 7 in 2019.<br>8. Osu! an open-source rhythm game.<br>9. NUnit, the unit testing library.<br>10. Xunit, the open-source unit testing tool that dropped from the 4th place in the 2018 top libraries edition.<br><br>https://visualstudiomagazine.com/articles/2019/12/03/top-csharp-libraries.aspx |
| **Domains** *What industries or domains use this programming language? Provide specific examples of companies that use this language and what they use it for.* | C# is a popular, high-level, object-oriented language developed and maintained by Microsoft. It is easy to learn and highly scalable. It is particularly strong for building Windows desktop applications, games, and web applications. It is becoming increasingly popular for mobile development as well. StackOverflow, a highly respected community of software experts on the web was constructed in C#, something of which I was not aware until doing the research for this assignment. (It was also interesting to learn that C# developers are in big demand in the job market.) And, of course C# is used extensively by |

| | |
|---|---|
| **Domains (cont.)** | Microsoft for development of its own web services, applications and games. Finally, a very specific example of an industrial application of C# is Intuit's online .NET SDK that allows developers to interface their .NET web apps with the online Quickbooks API using C#. |