Aaron McCarley
CS311
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.

The C# language is strongly typed, which means that when attempts are made to pass the wrong parameter as an argument or assigned value it will generate a compilation error. The purpose of this is to help avoid errors at runtime.

A declaration statement for a variable declares a new variable and has the option to initialize it. Since all variables have a declared type then an example of this in C# would be type varName = total; type (into or string) and varName is the name of the variable. The assigned values in this case is the value or total the equal sign assigns.

Example Code 1:

```
string name = "Bob";

Console.WriteLine(name);
```

Example Code 2:

```
int myNum = 35;

Console.WriteLine(myNum);
```

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

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

	float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits	
	double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits	
	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	(Predefined Data Types in C#).

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

IF Statement

An if statement checks the given condition. If the condition evaluates to be true then the block of code/statements with execute otherwise.

```
if(condition)
    {
        //code to be executed
    }
```

Example: // C# program to illustrate if statement

```
public class Town {
```

```
    public static void Main(string[] args)
    {
        string name = "Townsquare";
        if (name == "GTownsquare") {
            Console.WriteLine("TownforSquares");
        }
    }
}
```

IF-Else Statement

if statements evaluate the code if the condition is true, we use the else statement to tell the code what to do when the if condition is false.

```
    if(condition)
    {
      // code if condition is true
    }
    else
    {
      // code if condition is false
    }
```

Example:

```
// C# program to illustrate if-else statement

public class Town {

    public static void Main(string[] args)
    {
        string name = "Townsquare";
```

```
        if (name == "Townsquare") {
            Console.WriteLine("TownForSquares");
        }
        else {
            Console.WriteLine("Townsquare");
        }
    }
}
```

IF-ELSE-IF Statement

if-else-if is a ladder statement which executes one condition from multiple statements. The statement of if block will be executed which evaluates to be true. If none of the if condition evaluates to be true then the last else block is then evaluated.

```
if(condition1)
        {
            // code to be executed if condition1 is true
        }
        else if(condition2)
        {
            // code to be executed if condition2 is true
        }
        else if(condition3)
        {
            // code to be executed if condition3 is true
        }
...
        else
        {
```

```
        // code to be executed if all the conditions are false
    }
```

Example:

```
// C# program to illustrate if-else-if ladder
class Town {

  public static void Main(String[] args)
  {
    int t = 30;

    if ( t == 10)
      Console.WriteLine("t is 10");
    else if (t == 20)
      Console.WriteLine("t is 20");
    else if (t == 25)
      Console.WriteLine("t is 25");
    else
      Console.WriteLine("t is not present");
  }
}
```

Nested IF Statement

An if statement inside an if statement is known as nested if statement. The if statement is the target of another if statement, and is used when more than one condition needs to be true and one of the condition is the sub-condition of the parent condition.

```
if (condition1)
    {
      // code to be executed
      // if condition2 is true
      if (condition2)
```

```
        {
            // code to be executed
            // if condition2 is true
        }
    }
```

Example:

```
class Town {

  public static void Main(String[] args)
  {
    int i = 20;

    if (i == 20) {

      // Nested - if statement
      // Will only be executed if statement
      // above it is true
      if (i < 8)
        Console.WriteLine("i is smaller than 8 too");
      else
        Console.WriteLine("i is greater than 8");
    }
  }
}
```

Switch Statement

An alternative to long if-else-if ladders. The expression is checked for different cases and the one match is executed. break statement is used to move out of the switch. If the break is not used, the control will flow to all cases below it until break is found or switch comes to an end. There is default case (optional) at the end of switch, if none of the case matches then default case is executed.

	switch (expression)
	{
	case value1: // statement sequence
	break;
	case value2: // statement sequence
	break;
	.
	.
	.
	case valueN: // statement sequence
	break;
	default: // default statement sequence
	}
	
	// C# example for switch case
	public class Town
	{
	public static void Main(String[] args)
	{
	int number = 30;
	switch(number)
	{
	case 10: Console.WriteLine("case 10");
	break;
	case 20: Console.WriteLine("case 20");
	break;

```
        case 30: Console.WriteLine("case 30");
                 break;
        default: Console.WriteLine("None matches");
                 break;
        }
     }
  }
```

Nested switch

Nested Switch case are allowed in C# . In this case, switch is present inside other switch case. Inner switch is present in one of the cases in parent switch.

```
// C# example for nested switch case

public class Town
{
   public static void Main(String[] args)
   {
      int j = 5;

      switch (j)
      {
         case 5: Console.WriteLine(5);
                 switch (j - 1)
                 {
                 case 4: Console.WriteLine(4);
                         switch (j - 2)
                        {
                         case 3: Console.WriteLine(3);
                                 break;
                        }
```

	break;
	}
	break;
	case 10: Console.WriteLine(10);
	break;
	case 15: Console.WriteLine(15);
	break;
	default: Console.WriteLine(100);
	break;
	}
	}
	}

Repetition Structures
Provide examples of all repetition structures supported by this language (loops, etc.)
	While Loop
	
	The while loop is probably the most simple one, so we will start with that. The while loop simply executes a block of code as long as the condition you give it is true. A small example, and then some more explanation:
	
	using System;
	
	namespace ConsoleApplication1
	{
	class Program
	{
	static void Main(string[] args)
	{
	int number = 0;
	
	while(number < 5)
	{
	Console.WriteLine(number);
	number = number + 1;
	}
	
	Console.ReadLine();
	}
	}
	}

Do Loop

The do loop evaluates the condition after the loop has been executed, this ensures the code block is always executed at least once.

```
int number = 0;
do
{
   Console.WriteLine(number);
   number = number + 1;
} while(number < 6);
```

The output is the same, once the number is more than 6, the loop is resolved or exited.

For Loop

The for loop is preferred when you know how many iterations you want to perform, or because you have a variable containing
A set amount.

```
namespace ConsoleApplication1
{
   class Program
   {
   static void Main(string[] args)
   {
      int number = 5;

      for(int i = 0; i < number; i++)
      Console.WriteLine(i);

      Console.ReadLine();
   }
   }
}
```

The first part, where we define the i variable and set it to 0, is only executed once, before the loop starts.
The last 2 parts are executed for each iteration of the loop. Each time, i is compared to our number variable

	- if i is smaller than number, the loop runs one more time. After that, i is increased by one.
	
	Foreach Loop
	
	the foreach loop. It operates on collections of items, for instance arrays or other built-in list types.
	In our example we will use one of the simple lists, called an ArrayList. It works much like an array,
	but don't worry, we will look into it in a later chapter.
	
	using System;
	using System.Collections;
	
	namespace ConsoleApplication1
	{
	class Program
	{
	static void Main(string[] args)
	{
	ArrayList list = new ArrayList();
	list.Add("John Doe");
	list.Add("Jane Doe");
	list.Add("Someone Else");
	
	foreach(string name in list)
	Console.WriteLine(name);
	
	Console.ReadLine();
	}
	}
	}
	
	Okay, so we create an instance of an ArrayList, and then we add some string items to it. We use the foreach
	loop to run through each item, setting the name variable to the item we have reached each time. That way,
	we have a named variable to output. As you can see, we declare the name variable to be of the string type
	– you always need to tell the foreach loop which datatype you are expecting to pull out of the collection.
	In case you have a list of various types, you may use the object class instead of a specific class,
	to pull out each item as an object

Arrays *If this language supports arrays, provide an example*	**An array can be declared using by specifying the type of its elements with square brackets.** int[] countedNums; // integer array

of creating an array with a primitive data type (e.g. float, int, etc.)	string[] towns; // string array
	
	The following declares and adds values into an array in a single statement.
	
	int[] evenNums = new int[5]{ 2, 4, 6, 8, 10 };
	
	string[] cities = new string[3]{ "Mumbai", "London", "New York" };
	
	Above, evenNums array can store up to five integers.
	
	Arrays type variables can be declared using var without square brackets.
	
	Example:
	var evenNums = new int[]{ 2, 4, 6, 8, 10};
	
	var cities = new string[]{ "Mumbai", "London", "New York" };
	
	If you are adding array elements at the time of declaration, then size is optional.
	
	The compiler will infer its size based on the number of elements inside curly braces, as shown below.
	
	Accessing Array Elements
	
	Array elements can be accessed using an index.
	
	An index is a number associated with each array element, starting with
	
	index 0 and ending with array size - 1.
	
	Example:
	
	int[] evenNums = new int[5];
	evenNums[0] = 2;
	evenNums[1] = 4;
	//evenNums[6] = 12; //Throws run-time exception IndexOutOfRange
	
	Console.WriteLine(evenNums[0]); //prints 2
	Console.WriteLine(evenNums[1]); //prints 4

All the arrays in C# are derived from an abstract base class System.Array.
The Array class implements the IEnumerable interface, so you can
LINQ extension methods such as Max(), Min(), Sum(), reverse(), etc.
Example:

```
int[] nums = new int[5]{ 10, 15, 16, 8, 6 };

nums.Max(); // returns 16
nums.Min(); // returns 6
nums.Sum(); // returns 55
nums.Average(); // returns 55
```

Passing Array as an Argument

```
public static void Main(){
   int[] nums = { 1, 2, 3, 4, 5 };

   UpdateArray(nums);

   foreach(var item in nums)
      Console.WriteLine(item);
}

public static void UpdateArray(int[] arr)
{
   for(int i = 0; i < arr.Length; i++)
      arr[i] = arr[i] + 10;
}
```

Data Structures

If this language provides a standard set of data structures, provide a list of the data structures and their Big-Oh complexity.

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)
	(Algorithmic complexity of collections)				

Objects
If this language support object-orientation, provide an example of how to create a simple object with a default constructor.
	C# is an object-oriented programming language. The four basic principles of object-oriented programming are:
	
	Abstraction. Modeling the relevant attributes and interactions of entities as classes to define an abstract representation of a system.
	
	Encapsulation. Hiding the internal state and functionality of an object and only allowing access through a public set of functions.
	
	Inheritance. Ability to create new abstractions based on existing abstractions.
	
	Polymorphism. Ability to implement inherited properties or methods in different ways across multiple abstractions.
	
	(Object-Oriented programming (C#))
	
	Polymorphism Example:
	
	All three of these account types must have an action which takes places at the end of each month.
	
	public BankAccount(string name, decimal initBalance)
	
	public IntEarnAccount(string name, decimal initlBalance) : base(name, initBalance)
	{
	}
	
	public virtual void PerfMonthEndTransactions() { }
	
	public override void PerfMonthEndTransactions()
	{
	  if (Balance > 500m)
	  {
	  decimal interest = Balance * 0.05m;				

<table>
<tr><td></td><td>
<pre>
        Deposit(interest, DateTime, "apply monthly interest");
    }
}

public override void MonthEndTransactions()
{
    if (Balance < 0)
    {
        // Negate the balance to get a positive interest charge:
        decimal interest = -Balance * 0.07m;
        MakeWithdrawal(interest, DateTime.Now, "Charge monthly interest");
    }
}
</pre>
</td></tr>
<tr><td>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?</td><td>C# is designed to be run on the Common Language Runtime (CLR).which is the basic and Virtual Machine component of the . NET Framework. It is the run-time environment in the .NET Framework that runs the codes and helps in making the development process easier by providing various services. A common language runtime (CLR) routine is an external routine created by executing a CREATE PROCEDURE or CREATE FUNCTION statement that references a .NET assembly as its external code body.

Visual Basics, Visual or Managed C++, J# all use CLR. But it can be used with any language that can be complied in an intermediate language.

(Common Language Runtime (CLR) in C#.)</td></tr>
<tr><td>Libraries/Frameworks
What are the popular libraries or frameworks used by programmers for this language? List at least three (3).</td><td>.NET has a very large standard set of class libraries, these are referred too
As either the base class libraries (core set) or framework class libraries (complete set).

For framework class libraries – System.Object, System.Int16 (shot), SystemInt32 (float). See below.
(Framework Libraries)
(NET Framework Class Library)

<table>
<tr><th>Namespaces</th><th>Description</th></tr>
<tr><td>System</td><td>It includes all common datatypes, string values, arrays and methods for data conversion.</td></tr>
</table>
</td></tr>
</table>

	<table><tr><td>System.Data, System.Data.Common, System.Data.OleDb, System.Data.SqlClient, System.Data.SqlTypes</td><td>These are used to access a database, perform commands on a database and retrieve database.</td></tr><tr><td>System.IO, System.DirectoryServices, System.IO.IsolatedStorage</td><td>These are used to access, read and write files.</td></tr><tr><td>System.Diagnostics</td><td>It is used to debug and trace the execution of an application.</td></tr><tr><td>System.Net, System.Net.Sockets</td><td>These are used to communicate over the Internet when creating peer-to-peer applications.</td></tr></table>
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# was a programming language created by Microsoft for Microsoft, so they are the biggest example that I know of a business using this language. Accenture is another large corporation that uses C#, as well as Intuit. There are many companies that use C# for a variety of reasons. Why do they use it? Because they could develop specific applications and programs to meet the needs of the Microsoft platform. How so? You can save a tremendous amount of time using C# compared to different languages by providing an simple And efficient way to scale a business, whether that developing games, websites, mobile apps, desktop apps, cloud based services etc. (A tour of the C# Language)

References

1) Predefined Data Types in C#. https://www.tutorialsteacher.com/csharp/csharp-data-types. Copyright TutorialsTeacher 2022
2) Algorithmic complexity of collections. https://learn.microsoft.com/en-us/dotnet/standard/collections/. Copyright Microsoft 2022.
3) Object-Oriented programming (C#). https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/oop. Copyright Microsoft 2022.

4) Common Language Runtime (CLR) in C#. https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/. By Anshul Aggarwal. 24 Nov. 2021.
5) .NET Framework Class Library, https://www.javatpoint.com/net-framework-class-library. Copyright 2011-2021, Developed by JavaTpoint.
6) Framework Libraries. https://learn.microsoft.com/en-us/dotnet/standard/framework-libraries. Copyright Microsoft 2022.
7) A tour of the C# Language. https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/. Copyright Microsoft 2022.