Stephen Brom

10/22/2023

2023FA CS-311-ON

Assignment 1 Basic Structures of the C# Programming Language

Basic Structures of the C# Programming Language

Language Map for c#

Variable Declaration
Is this language strongly typed or dynamically typed? Provide at least three examples (with different data types or keywords) of how variables are declared in this language.
	This is a strongly typed language. This means variable types are checked at compile time, and any attempt to assign a value that isn't convertible will generate a compilation error.
	Explicitly Typed Variables: You specify the type of the variable and provide its name:
	string greeting = "Hello";
	int a = 3, b = 2, c = a + b;
	List<double> xs = new List<double>();
	Implicitly Typed Variables: You can let the compiler infer the type of the variable from its initialization expression. To do that, use the var keyword instead of a type's name:
	var greeting = "Hello";
	var a = 32;
	var xs = new List<double>();
	Constants: To declare a local constant, use the const keyword:
	const string Greeting = "Hello";
	const double MinLimit = -10.0, MaxLimit = -MinLimit;

Data Types *List all of the data types (and ranges) supported by this language.*	1. Integral Numeric Types: ○ 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 (computed at runtime) (Signed 32-bit or 64-bit integer) ○ nuint: Depends on platform (computed at runtime) (Unsigned 32-bit or 64-bit integer) 2. Floating Point Types: ○ float: Approximate range of ±1.5 x 10−4510−45 to ±3.4 x 10381038 with ~6-9 digits precision ○ double: Approximate range of ±5.0 x 10−32410−324 to ±1.7 x 1030810308 with ~15-17 digits precision 3. Decimal Type: ○ decimal: Approximate range of ±1.0 x 10−2810−28 to ±7.9228 x 10281028 with ~28-29 digits precision 4. Boolean Type: ○ bool: Can be either true or false 5. Character Type: ○ char: Unicode symbols used in text ranging from U+0000 to U+FFFF 6. String Type: ○ string: A sequence of characters

Selection Structures
Provide examples of all selection structures supported by this language (if, if else, etc.) ***Don't just list them, show code samples of how each would look in a real program.***
	1. If Statement
	
	int number = 10;
	if (number > 0)
	{
	Console.WriteLine("Number is positive.");
	}
	
	In this example, the message "Number is positive." will be printed if the number is greater than 0.
	
	2. If-Else Statement
	
	int number = -5;
	if (number > 0)
	{
	Console.WriteLine("Number is positive.");
	}
	else
	{
	Console.WriteLine("Number is not positive.");

	}
	
	In this example, since the number is not greater than 0, the message “Number is not positive.” will be printed.
	
	3. Else If Statement
	
	int number = 0;
	if (number > 0)
	{
	Console.WriteLine("Number is positive.");
	}
	else if (number < 0)
	{
	Console.WriteLine("Number is negative.");
	}
	else
	{
	Console.WriteLine("Number is zero.");
	}
	
	In this example, since the number is neither greater than 0 nor less than 0, the message “Number is zero.” will be printed.
	
	4. Switch Statement
	
	int dayOfWeek = 3;
	switch (dayOfWeek)
	{
	case 1:
	Console.WriteLine("Monday");
	break;
	case 2:
	Console.WriteLine("Tuesday");
	break;
	case 3:
	Console.WriteLine("Wednesday");
	break;
	// cases for the rest of the week can be added here.

	default: Console.WriteLine("Invalid day of the week"); break; } In this example, since dayOfWeek is 3, “Wednesday” will be printed. If dayOfWeek does not match any case, “Invalid day of the week” will be printed.

Repetition Structures
Provide examples of all repetition structures supported by this language (loops, etc.) ***Don’t just list them, show code samples of how each would look in a real program.***
	1. For Loop
	
	for (int i = 0; i < 5; i++)
	{
	Console.WriteLine(i);
	}
	
	This for loop will print the numbers 0 through 4.
	
	2. While Loop
	
	int i = 0;
	while (i < 5)
	{
	Console.WriteLine(i);
	i++;
	}
	
	This while loop will also print the numbers 0 through 4.
	
	3. Do-While Loop
	
	int i = 0;
	do
	{
	Console.WriteLine(i);
	i++;
	} while (i < 5);

	This do-while loop will also print the numbers 0 through 4.
	
	4. Foreach Loop
	
	string[] names = { "John", "Jane", "Jim" };
	foreach (string name in names)
	{
	Console.WriteLine(name);
	}
	
	This foreach loop will print each name in the names array.

Arrays *If this language supports arrays, provide **at least two examples** of creating an array with a primitive or String data types (e.g. float, int, String, etc.) If the language supports declaring arrays in multiple ways, provide an example of way.*	1. Integer Array int[] numbers = new int[5] { 1, 2, 3, 4, 5 }; This creates an integer array named numbers with the elements 1, 2, 3, 4, and 5. 2. String Array string[] names = new string[3] { "John", "Jane", "Jim" }; This creates a string array named names with the elements “John”, “Jane”, and “Jim”. 3. Implicitly Typed Array var numbers = new[] { 1, 2, 3, 4, 5 }; This creates an implicitly typed integer array named numbers with the elements 1, 2, 3, 4, and 5. The compiler infers the type of the array from the elements specified. Remember that arrays in C# are zero-indexed, which means the first element is at index 0. You can access individual elements of the array using their index. For example, numbers[0] would give you the first element of the numbers array.

Data Structures
If this language provides a standard set of data structures, provide a list of the data structures and their Big-Oh complexity (identify what the complexity represents).
	1. Array
	   - Access: O(1)
	   - Search: O(n)
	   - Insertion: O(n)
	   - Deletion: O(n)
	2. List
	   - Access: O(1)
	   - Search: O(n)
	   - Insertion: O(n)
	   - Deletion: O(n)
	3. LinkedList
	   - Access: O(n)
	   - Search: O(n)
	   - Insertion: O(1)
	   - Deletion: O(1)
	4. Stack
	   - Access: O(n)
	   - Search: O(n)
	   - Insertion: O(1)
	   - Deletion: O(1)
	5. Queue
	   - Access: O(n)
	   - Search: O(n)
	   - Insertion: O(1)
	   - Deletion: O(1)
	6. HashSet
	   - Access: N/A
	   - Search: O(1)
	   - Insertion: O(1)
	   - Deletion: O(1)
	7. Dictionary (HashMap)
	   - Access: N/A
	   - Search: O(1)
	   - Insertion: O(1)
	   - Deletion: O(1)

	Please note that these are average-case complexities. The actual time complexity can vary depending on the specific operation and data involved. In the above list: • "Access" refers to getting an element in the data structure. • "Search" refers to finding an element in the data structure. • "Insertion" refers to adding an element to the data structure. • "Deletion" refers to removing an element from the data structure. Remember that different data structures are suited for different tasks, so it's important to choose the right one based on your specific needs. In computer science, Big O notation is used to classify algorithms according to how their running time or space requirements grow as the input size grows. Here's what each complexity in the list represents: 1. O(1): Constant Time Complexity o The running time of the operation does not change with the size of the input data set. Whether it's 10 items or 10,000 items, it will take the same amount of time. 2. O(n): Linear Time Complexity o The running time of the operation will grow linearly with the size of the input data set. If you double the number of items, the running time will roughly double as well. 3. O(n^2): Quadratic Time Complexity o The running time of the operation is directly proportional to the square of the size of the input data set. This is common with algorithms that involve nested iterations over the data set. 4. O(log n): Logarithmic Time Complexity o The running time of the operation grows logarithmically with the size of the input data set. The operation becomes slower as the data set size increases, but once the data set is sorted, it speeds up dramatically. 5. O(n log n): Linear Logarithmic Time Complexity o The running time of the operation grows linearly and logarithmically at the same time. This is common with more efficient sorting algorithms. Remember that Big O notation represents an upper bound on the time complexity, describing the worst-case scenario. An algorithm may run faster for some inputs than what its Big O notation suggests.

Objects
If this language support object-orientation, provide an example of how you would write a simple object with a default constructor and then how you would instantiate it.
	In C#, you can define an object by creating a class. Here's an example of a simple class with a default constructor:
	
	public class Person
	{
	  // Fields
	  public string Name;
	  public int Age;
	
	  // Default constructor
	  public Person()
	  {
	  Name = "Unknown";
	  Age = 0;
	  }
	}
	
	In this example, Person is a class, which is a blueprint for creating objects. This class has two fields: Name and Age. The Person() method is a special method called a constructor, which is called when an object is created from the class.
	
	You can create (instantiate) an object from this class like this:
	
	Person person1 = new Person();
	
	Now person1 is an object of the Person class. You can access the fields of the object using the dot operator:
	
	Console.WriteLine(person1.Name); // Outputs "Unknown"
	Console.WriteLine(person1.Age); // Outputs "0"
	
	Remember that this is a very basic example. In real-world programs, you would typically use properties and methods to encapsulate the fields of the class, and you might have multiple constructors for more

	flexibility when creating objects. But this should give you a good starting point for understanding objects in C#.
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? If so, what these other languages?*	C# compiles to the Common Language Runtime (CLR), which is a component of the .NET Framework. CLR is a virtual machine that not only executes the code but also provides services that make the development process easier. The .NET Framework also includes a .NET SDK which is used to create .NET apps and libraries. The .NET runtime is always installed with the SDK. As for other languages that compile to the same runtime as C#, there are several languages that can be compiled to run on the CLR. These include, but are not limited to, Visual Basic .NET and F#. On the other hand, Java compiles to the Java Virtual Machine (JVM). Apart from Java, there are several other languages that can run on the JVM. These include, but are not limited to, Scala, Kotlin, Groovy, and Clojure. It's worth noting that these languages are compatible with each other to some extent. For example, Scala libraries can be used with Java programs and vice versa.
Libraries/Frameworks *What are the popular libraries or frameworks used by programmers for this language? List at least three (3) and describe what they are used for.*	1. Entity Framework Core: It's a lightweight, extensible, open-source, and cross-platform version of Entity Framework data access technology[3]. 2. AutoMapper: It's a convention-based object-object mapper in .NET. It's used to get rid of code that mapped one object to another[2]. 3. FluentValidation: It's a popular .NET validation library for building strongly-typed validation rules[2]. 4. MediatR: It's a simple, unambitious mediator implementation in .NET for in-process messaging with no dependencies[2]. 5. Autofac: It's an IoC container for Microsoft .NET. It manages the dependencies between classes so that applications stay easy to change as they grow in size and complexity[2]. 6. Hangfire: It's an easy way to perform background job processing in your .NET and .NET Core applications[2].

	7. Swashbuckle (Swagger — OpenAPI): It simplifies API development for users, teams, and enterprises with the Swagger open source and professional toolset[2]. 8. Serilog: It's a diagnostic logging library for .NET applications. It is easy to set up, has a clean API, and runs on all recent .NET platforms[2].
Domains *What industries or domains use this programming language? Provide at least three specific examples of companies that use this language and what they use it for****. E.g. Company X uses C# for its line of business applications.***	C# is a versatile language used across various industries for different purposes. Here are some specific examples of companies that use C# and what they use it for: 1. Microsoft: Microsoft, the creator of C#, uses it extensively for web and game development1. 2. Stack Overflow: This popular platform for developers uses C# for app development and web services1. 3. Trustpilot: This online review platform uses C# for web services and app development1. 4. Service Titan: This software company uses C# for web services and Android app development1. 5. Slack, Insightly, Pinterest, Tableau, The World Bank, Cinemark, NBCUniversal, FoxSports, Aviva, and Taxfyle: These companies have mobile apps written in C#2. These are just a few examples. Many other companies across different industries use C# for a variety of applications such as web development, game development, cloud-based services, Windows apps, and more. It's worth noting that the demand for C# is continuously increasing with 2245 companies reportedly using C# in their tech stacks.