Code Clock

Day 4: String Manipulation

Learn.to.code

Programming with C#

@ QUB

Introduction to the String Class in C#

In C#, **strings** are objects of the `String` class, which is part of the **System** namespace.

Strings in C# are **immutable**, meaning their values cannot be changed once they are created. If you modify a string, a new string object is created.

Creating a String

You can create strings in C# by using string literals:

```
string greeting = "Hello, World!";
```

You can also create a string from an array of characters:

```
char[] letters = { 'H', 'e', 'l', 'l', 'o' };
string word = new string(letters);  // "Hello"
```

Commonly Used Methods and Properties

Length Property

The `Length` property returns the number of characters in the string.

```
string greeting = "Hello, World!";
int length = greeting.Length;  // 13
```

Accessing Characters

You can access individual characters in a string using index notation:

```
char firstLetter = greeting[0];  // 'H'
```

String Concatenation

You can concatenate strings using the `+` operator or the `Concat` method:

```
string part1 = "Hello";
string part2 = "World";
string fullGreeting = part1 + ", " + part2 + "!";  // "Hello, World!"
```

You can also use `String.Concat`:

```
string fullGreeting = String.Concat(part1, ", ", part2, "!");  // "Hello, World!"
```

String Interpolation

String interpolation allows you to insert variable values directly into the string:

Method/Property	Description	Static/Instance	Example
METHODS			
ToLower()	Converts all characters to lowercase.	***Instance***	greeting.ToLower(); // "hello, world!"
ToUpper()	Converts all characters to uppercase.	***Instance***	greeting.ToUpper(); // "HELLO, WORLD!"
Contains(string value)	Determines whether the string contains the specified substring.	***Instance***	greeting.Contains("World"); // true
Substring(int startIndex)	Extracts a substring from the string starting at the specified index.	***Instance***	greeting.Substring(7); // "World!"
Substring(int startIndex, int length)	Extracts a substring with a specified length.	***Instance***	greeting.Substring(7, 5); // "World"
IndexOf(char value)	Returns the index of the first occurrence of the specified character.	***Instance***	greeting.IndexOf('W'); // 7
Trim()	Removes all leading and trailing whitespace characters.	***Instance***	greeting.Trim(); // "Hello, World!"
Replace(string oldValue, string newValue)	Replaces all occurrences of a specified string with another string.	***Instance***	greeting.Replace("World", "Alice"); // "Hello, Alice!"
Split(char separator)	Splits a string into an array of substrings based on a delimiter.	***Instance***	greeting.Split(','); // ["Hello", " World!"]
IsNullOrEmpty(string value)	Determines whether the specified string is null or empty.	***Static***	String.IsNullOrEmpty(greeting); // false
IsNullOrWhiteSpace(string value)	Determines whether the specified string is null or consists only of white-space characters.	***Static***	String.IsNullOrWhiteSpace(greeting); // false
PROPERTIES			
Length	Returns the number of characters in the string.	***Instance***	greeting.Length; // 13

Example Program: Working with Strings

```
class Program
{
    static void Main()
    {
        // Create a string
        string greeting = "  Hello, World!  ";

        // Trim leading and trailing whitespaces
        string trimmedGreeting = greeting.Trim();
        Console.WriteLine($"Trimmed: '{trimmedGreeting}'");

        // Convert to lowercase
        string lowerGreeting = trimmedGreeting.ToLower();
        Console.WriteLine($"Lowercase: '{lowerGreeting}'");

        // Find the index of a substring
        int indexOfWorld = lowerGreeting.IndexOf("world");
        Console.WriteLine($"Index of 'world': {indexOfWorld}");

        // Extract a substring
        string subGreeting = lowerGreeting.Substring(indexOfWorld, 5);
        Console.WriteLine($"Substring: '{subGreeting}'");

        // Replace a word in the string
        string replacedGreeting = lowerGreeting.Replace("world", "Alice");
        Console.WriteLine($"Replaced: '{replacedGreeting}'");

        // Check if the string contains "hello"
        bool containsHello = lowerGreeting.Contains("hello");
        Console.WriteLine($"Contains 'hello': {containsHello}");
    }
}
```

Summary:

- Strings in C# are **immutable** and require creating a new string whenever a modification is made.
- Common methods include `ToLower()`, `ToUpper()`, `Substring()`, and `Replace()`.
- Properties like `Length` and methods like `IndexOf()` are helpful for working with string manipulation.

The following 30 challenges will test your understanding of methods and properties of the string class

Basic

1. Create a program that takes a user's input and displays the length of the input string.
2. Write a program that concatenates two strings and displays the result.
3. Create a program that converts a string to all uppercase.
4. Write a program that checks if a string contains a specific word or phrase.
5. Create a program that replaces all occurrences of a word in a string with another word.
6. Write a program that trims whitespace from the beginning and end of a string.
7. Create a program that extracts a substring from a given string.
8. Write a program that checks if a string starts with a specific prefix.
9. Create a program that counts the number of vowels in a given string.
10. Write a program that reverses a given string.

Intermediate

11. Create a program that checks if a string is a palindrome (reads the same backward and forward).
12. Write a program that splits a comma-separated string into individual elements and displays them.
13. Create a program that finds the index of the first occurrence of a specific character in a string.
14. Write a program that removes duplicate characters from a string.
15. Create a program that formats a string as a phone number (e.g., "1234567890" becomes "(123) 456-7890").
16. Write a program that extracts all email addresses from a given string.
17. Create a program that counts the number of words in a given string.
18. Write a program that capitalizes the first letter of each word in a sentence.
19. Create a program that converts a string to a title case.
20. Write a program that replaces HTML tags in a string with appropriate placeholders.

Expert

21. Create a program that sorts a list of strings in alphabetical order.
22. Write a program that finds the longest word in a sentence.
23. Create a program that encrypts a string using a simple Caesar cipher.
24. Write a program that decodes a string encrypted with a Caesar cipher.
25. Create a program that extracts all URLs from a given string.
26. Write a program that checks if a string is a valid email address.
27. Create a program that parses a CSV (Comma-Separated Values) string into a list of records.
28. Write a program that generates a random password with a specified length.

29. Create a program that calculates the Levenshtein distance between two strings.
30. Write a program that tokenizes a sentence into words and then sorts the words based on their length.