**1. Length**

* **Definition**: Gets the number of characters in the string.
* **Example**:

csharp

int length = "Hello, World!".Length; // Output: 13

**2. string.Compare**

* **Definition**: Compares two specified string objects and returns an integer indicating their relative position in the sort order.
* **Example**:

csharp

int comparison1 = string.Compare("apple", "banana"); // Output: -1 (apple is less than banana)

** Compare 'a' (97 in Unicode) with 'b' (98 in Unicode).**

** Since 'a' < 'b', the result is -1**.

int comparison2 = string.Compare("banana", "apple"); // Output: 1 (banana is greater than apple)

int comparison3 = string.Compare("apple", "apple"); // Output: 0 (equal)

**3. CompareTo**

* **Definition**: Compares the current instance with another string and returns an integer indicating their relative position.
* **Example**:

csharp

int result1 = "apple".CompareTo("banana"); // Output: -1

int result2 = "banana".CompareTo("apple"); // Output: 1

int result3 = "apple".CompareTo("apple"); // Output: 0

**4. .Contains**

* **Definition**: Determines whether a string contains a specified substring.
* **Example**:

bool contains1 = "Hello, World!".Contains("World"); // Output: true

bool contains2 = "Hello, World!".Contains("world"); // Output: false (case-sensitive)

**5. EndsWith**

* **Definition**: Determines whether the end of this string instance matches a specified string.
* **Example**:

csharp

bool endsWith1 = "Hello, World!".EndsWith("World!"); // Output: true

bool endsWith2 = "Hello, World!".EndsWith("world!"); // Output: false (case-sensitive)

**6. Equals**

* **Definition**: Determines whether two specified string instances are equal.
* **Example**:

csharp

bool equals1 = "Hello".Equals("hello"); // Output: false (case-sensitive)

bool equals2 = "Hello".Equals("hello", StringComparison.OrdinalIgnoreCase); // Output: true

**7. IndexOf**

* **Definition**: Reports the zero-based index of the first occurrence of a specified substring within this string.
* **Example**:

csharp

Copy

int index1 = "Hello, World!".IndexOf("World"); // Output: 7

int index2 = "Hello, World!".IndexOf("world"); // Output: -1 (not found)

**8. LastIndexOf**

* **Definition**: Reports the zero-based index of the last occurrence of a specified substring within this string.
* **Example**:

csharp

Copy

int lastIndex1 = "Hello, World! World!".LastIndexOf("World"); // Output: 13

int lastIndex2 = "Hello, World!".LastIndexOf("world"); // Output: -1 (not found)

**9. Insert**

* **Definition**: Returns a new string in which a specified string is inserted at a specified index.
* **Example**:

csharp

string inserted = "Hello, !".Insert(7, "World"); // Output: "Hello, World!"

**10. Remove**

* **Definition**: Returns a new string in which a specified number of characters is removed from the current string.
* **Example**:

csharp

string removed = "Hello, World!".Remove(5); // Output: "Hello"

**11. Replace**

* **Definition**: Returns a new string in which all occurrences of a specified string are replaced with another specified string.
* **Example**:

csharp

string replaced1 = "Hello, World!".Replace("World", "C#"); // Output: "Hello, C#!"

string replaced2 = "Hello, World!".Replace("world", "C#"); // Output: "Hello, World!" (no change, case-sensitive)

**12. Split**

* **Definition**: Splits a string into substrings based on specified delimiters.
* **Example**:

csharp

string[] split1 = "apple,banana,cherry".Split(','); // Output: { "apple", "banana", "cherry" }

**13. Substring**

* **Definition**: Retrieves a substring from this instance.
* **Example**:

csharp

string substring1 = "Hello, World!".Substring(7, 5); // Output: "World"

**14. ToLower**

* **Definition**: Converts the string to lowercase.
* **Example**:

csharp

string lower = "Hello, World!".ToLower(); // Output: "hello, world!"

**15. ToUpper**

* **Definition**: Converts the string to uppercase.
* **Example**:

csharp

string upper = "Hello, World!".ToUpper(); // Output: "HELLO, WORLD!"

**16. Trim**

* **Definition**: Removes all leading and trailing white-space characters from the current string.
* **Example**:

csharp

string trimmed = " Hello, World! ".Trim(); // Output: "Hello, World!"

**17. TrimStart**

* **Definition**: Removes all leading white-space characters from the current string.
* **Example**:

csharp

string trimmedStart = " Hello, World! ".TrimStart(); // Output: "Hello, World! "

**18. TrimEnd**

* **Definition**: Removes all trailing white-space characters from the current string.
* **Example**:

csharp

string trimmedEnd = " Hello, World! ".TrimEnd(); // Output: " Hello, World!"

**19. ToCharArray**

* **Definition**: Copies the characters in the string to a Unicode character array.
* **Example**:

csharp

char[] charArray = "Hello".ToCharArray(); // Output: { 'H', 'e', 'l', 'l', 'o' }

**20. string.Join**

* **Definition**: Concatenates the elements of a string array, using the specified separator between each element.
* **Example**:

csharp

string joined = string.Join(", ", new[] { "apple", "banana", "cherry" }); // Output: "apple, banana, cherry"

string[] fruits = { "apple", "banana", "cherry" };

string result = string.Join(", ", fruits); // Output: "apple, banana, cherry"

**21. Format**

* **Definition**: Formats the specified string by substituting the specified objects' values in place of the format items in the string.
* **Example**:

csharp

string formatted = string.Format("Hello, {0}!", "World"); // Output: "Hello, World!"

**22. string.Concat**

* **Definition**: Concatenates one or more instances of String or the string representations of the specified objects.
* **Example**:

csharp

string concatenated = string.Concat("Hello, ", "World!"); // Output: "Hello, World!"

string[] words = { "Hello", "world", "!" };

string sentence = String.Concat(words); // Output: "Helloworld!"

Console.WriteLine(sentence);

**23. PadLeft**

* **Definition**: Pads the string on the left with spaces or a specified character to the specified total length.
* **Example**:

csharp

string paddedLeft = "42".PadLeft(5, '0'); // Output: "00042"

**24. PadRight**

* **Definition**: Pads the string on the right with spaces or a specified character to the specified total length.
* **Example**:

csharp

string paddedRight = "42".PadRight(5, '0'); // Output: "42000"

**25. StartsWith**

* **Definition**: Determines whether the beginning of this string instance matches the specified string.
* **Example**:

csharp

bool startsWith1 = "Hello, World!".StartsWith("Hello"); // Output: true

bool startsWith2 = "Hello, World!".StartsWith("hello"); // Output: false (case-sensitive)

**26. CopyTo**

* **Definition**: Copies the characters in this instance to a specified array of Unicode characters, starting at a specified index.
* **Example**:

csharp

char[] array = new char[5]; "Hello".CopyTo(0, array, 0, 5); // array now contains { 'H', 'e', 'l', 'l', 'o' }

**27. Normalize**

* **Definition**: Returns a new string in which all the characters in this instance have been normalized using the specified normalization form.
* **Example**:

csharp

string normalized = "é".Normalize(); // Normalized string (depends on the normalization form)

**28. GetHashCode**

* **Definition**: Returns the hash code for this instance.
* **Example**:

csharp

int hashCode = "Hello".GetHashCode(); // Output: Hash code (integer value, varies)

**29. string.IsNullOrEmpty**

* **Definition**: Returns a value indicating whether the specified string is null or an empty string ("").
* **Example**:

csharp

bool isNullOrEmpty1 = string.IsNullOrEmpty(""); // Output: true

bool isNullOrEmpty2 = string.IsNullOrEmpty("Hello"); // Output: false

**30. string.IsNullOrWhiteSpace**

* **Definition**: Returns a value indicating whether the specified string is null, empty, or consists only of white-space characters.
* **Example**:

csharp

bool isNullOrWhiteSpace1 = string.IsNullOrWhiteSpace(" "); // Output: true

bool isNullOrWhiteSpace2 = string.IsNullOrWhiteSpace("Hello"); // Output: false

Array

**1. Array.Clear**

* **Definition**: Sets a range of elements in the array to their default value.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

Array.Clear(numbers, 1, 3); // Clears elements at index 1 to 3

// Output: { 1, 0, 0, 0, 5 }

**2. Array.Copy**

* **Definition**: Copies a section of one array to another array.
* **Example**:

csharp

int[] source = { 1, 2, 3, 4, 5 };

int[] destination = new int[5];

Array.Copy(source, destination, source.Length); // Copies entire source array

// Output: destination = { 1, 2, 3, 4, 5 }

**3. Array.IndexOf**

* **Definition**: Searches for the specified object and returns the index of its first occurrence in the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

int index = Array.IndexOf(numbers, 3); // Output: 2

return -1 if not present

**4. Array.LastIndexOf**

* **Definition**: Searches for the specified object and returns the index of its last occurrence in the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 2, 5 };

int lastIndex = Array.LastIndexOf(numbers, 2); // Output: 3

**5. Array.Sort**

* **Definition**: Sorts the elements in the entire one-dimensional array using the default comparer.
* **Example**:

csharp

int[] numbers = { 5, 2, 4, 1, 3 };

Array.Sort(numbers); // Sorts the array in ascending order

// Output: { 1, 2, 3, 4, 5 }

**6. Array.Reverse**

* **Definition**: Reverses the order of the elements in the entire one-dimensional array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

Array.Reverse(numbers); // Reverses the array

// Output: { 5, 4, 3, 2, 1 }

**7. Array. ze**

* **Definition**: Changes the number of elements of a specified array to the specified new size.
* **Example**:

csharp

int[] numbers = { 1, 2, 3 };

Array.Resize(ref numbers, 5); // Resizes array to hold 5 elements

// Output: { 1, 2, 3, 0, 0 }

**8. Array.Find**

* **Definition**: Searches for an element that matches the conditions defined by the specified predicate and returns the first occurrence within the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

int result = Array.Find(numbers, n => n > 3); // Output: 4

**9. Array.FindAll**

* **Definition**: Searches for all elements that match the conditions defined by the specified predicate and returns an array of all matching elements.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

int[] results = Array.FindAll(numbers, n => n > 2); // Output: { 3, 4, 5 }

**10. Array.ForEach**

* **Definition**: Performs the specified action on each element of the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

Array.ForEach(numbers, n => Console.WriteLine(n)); // Output: 1 2 3 4 5

**11. Array.Exists**

* **Definition**: Determines whether the specified array contains elements that match the conditions defined by the specified predicate.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 };

bool exists = Array.Exists(numbers, n => n == 3); // Output: true

**12. Array.Fill**

* **Definition**: Fills the entire array with the specified value.
* **Example**:

csharp

int[] numbers = new int[5];

Array.Fill(numbers, 7); // Fills the array with 7

// Output: { 7, 7, 7, 7, 7 }

**13. Array.BinarySearch**

* **Definition**: Searches a one-dimensional sorted array for a specific object and returns the zero-based index of the object.
* **Example**:

csharp

int[] numbers = { 1, 2, 3, 4, 5 }; // Must be sorted

int index = Array.BinarySearch(numbers, 3); // Output: 2

**14. Array.CopyTo**

* **Definition**: Copies all the elements of the current one-dimensional array to the specified one-dimensional array starting at the specified destination array index.
* **Example**:

csharp

int[] source = { 1, 2, 3 };

int[] destination = new int[3];

source.CopyTo(destination, 0); // Copies elements to destination

// Output: destination = { 1, 2, 3 }

**15. Array.GetLength**

* **Definition**: Gets the number of elements in the specified dimension of the array.
* **Example**:

csharp

int[,] matrix = { { 1, 2 }, { 3, 4 }, { 5, 6 } };

int length = matrix.GetLength(0); // Output: 3 (number of rows)

int columns = matrix.GetLength(1); // Output: 2 (number of columns)

**16. Array.GetUpperBound**

* **Definition**: Returns the index of the last element in the specified dimension of the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3 };

int upperBound = numbers.GetUpperBound(0); // Output: 2

**17. Array.GetLowerBound**

* **Definition**: Returns the index of the first element in the specified dimension of the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3 };

int lowerBound = numbers.GetLowerBound(0); // Output: 0

**18. Array.SetValue**

* **Definition**: Sets the value of the specified element in the array.
* **Example**:

csharp

int[] numbers = new int[3];

numbers.SetValue(5, 1); // Sets the second element to 5

// Output: numbers = { 0, 5, 0 }

**19. .GetValue**

* **Definition**: Gets the value of the specified element in the array.
* **Example**:

csharp

int[] numbers = { 1, 2, 3 };

int value = (int)numbers.GetValue(1); // Output: 2

**20. Array.Clone**

* **Definition**: Creates a shallow copy of the array.
* **Example**:

csharp

int[] original = { 1, 2, 3 };

int[] copy = (int[])original.Clone(); // Creates a shallow copy

**note: must declare like this**

// Output: copy = { 1, 2, 3 }

**21. Array.TrueForAll**

* **Definition**: Determines whether all elements in the array match the conditions defined by the specified predicate.
* **Example**:

csharp

int[] numbers = { 2, 4, 6 };

bool allEven = Array.TrueForAll(numbers, n => n % 2 == 0); // Output: true

22. .Length

To get length of an array

**Char functions**

**1. char.IsDigit(char)**

* **Definition**: Determines whether a character is a numeric digit (0-9).
* **Example**:

csharp

char c = '5';

Console.WriteLine(char.IsDigit(c)); // Output: True

**2. char.IsLetter(char)**

* **Definition**: Determines whether a character is an alphabetic letter (A-Z or a-z).
* **Example**:

csharp

char c = 'A';

Console.WriteLine(char.IsLetter(c)); // Output: True

**3. char.IsWhiteSpace(char)**

* **Definition**: Determines whether a character is categorized as a white-space character.
* **Example**:

csharp

char c = ' ';

Console.WriteLine(char.IsWhiteSpace(c)); // Output: True

**4. char.IsLower(char)**

* **Definition**: Determines whether a character is a lowercase letter.
* **Example**:

csharp

char c = 'g';

Console.WriteLine(char.IsLower(c)); // Output: True

**5. char.IsUpper(char)**

* **Definition**: Determines whether a character is an uppercase letter.
* **Example**:

csharp

char c = 'G';

Console.WriteLine(char.IsUpper(c)); // Output: True

**6. char.IsLetterOrDigit(char)**

* **Definition**: Determines whether a character is either a letter or a numeric digit.
* **Example**:

csharp

char c = '9';

Console.WriteLine(char.IsLetterOrDigit(c)); // Output: True

**7. char.IsPunctuation(char)**

* **Definition**: Determines whether a character is a punctuation mark.
* **Example**:

csharp

char c = '.';

Console.WriteLine(char.IsPunctuation(c)); // Output: True

**8. char.IsControl(char)**

* **Definition**: Determines whether a character is a control character, like a newline (\n), backspace, etc.
* **Example**:

csharp

char c = '\n';

Console.WriteLine(char.IsControl(c)); // Output: True

**9. char.IsSymbol(char)**

* **Definition**: Determines whether a character is a symbol character (like +, =, $, etc.).
* **Example**:

csharp

char c = '$';

Console.WriteLine(char.IsSymbol(c)); // Output: True

**10. char.IsSurrogate(char)**

* **Definition**: Determines whether a character is a surrogate code unit, which is used in UTF-16 encoding.
* **Example**:

csharp

char c = '\uD800'; // High surrogate

Console.WriteLine(char.IsSurrogate(c)); // Output: True

**11. char.ToUpper(char)**

* **Definition**: Converts a character to its uppercase equivalent.
* **Example**:

csharp

char c = 'g';

char upper = char.ToUpper(c); // Output: 'G'

Console.WriteLine(upper);

**12. char.ToLower(char)**

* **Definition**: Converts a character to its lowercase equivalent.
* **Example**:

csharp

char c = 'G';

char lower = char.ToLower(c); // Output: 'g'

Console.WriteLine(lower);

**13. char.GetUnicodeCategory(char)**

* **Definition**: Gets the Unicode category of the character, such as UppercaseLetter, LowercaseLetter, DecimalDigitNumber, etc.
* **Example**:

csharp

char c = 'A';

var category = char.GetUnicodeCategory(c); // Output: UppercaseLetter

Console.WriteLine(category);

**14. char.Parse(string)**

* **Definition**: Converts the first character of the string into a char. Throws an exception if the string is null or empty.
* **Example**:

csharp

char c = char.Parse("A"); // Output: 'A'

Console.WriteLine(c);

**15. char.TryParse(string, out char)**

* **Definition**: Tries to convert the first character of a string into a char without throwing an exception. Returns true if successful.
* **Example**:

csharp

if (char.TryParse("B", out char result))

{

Console.WriteLine(result); // Output: 'B'

}

**16. char.ConvertFromUtf32(int)**

* **Definition**: Converts a Unicode code point (UTF-32) into a string.
* **Example**:

csharp

string str = char.ConvertFromUtf32(0x1F600); // Output: 😀 (Grinning Face Emoji)

Console.WriteLine(str);

**17. char.ConvertToUtf32(char, char)**

* **Definition**: Converts a surrogate pair into a UTF-32 code point.
* **Example**:

csharp

int codePoint = char.ConvertToUtf32('\uD800', '\uDC00'); // Output: 65536

Console.WriteLine(codePoint);

**18. char.IsAscii(char) (available in newer versions of C#)**

* **Definition**: Determines whether a character is an ASCII character (from 0 to 127).
* **Example**:

csharp

char c = 'A';

Console.WriteLine(char.IsAscii(c)); // Output: True

**19. Enumerable.Range ?**

Enumerable.Range(start, count) is a **LINQ method** in C# that creates a sequence of integers starting from start and of length count.

✅ Example:

csharp

CopyEdit

var seq = Enumerable.Range(5, 4);

// seq = [5, 6, 7, 8]