1. When to use String vs. StringBuilder in C# ?

* **String**: Use **string** when you have a fixed sequence of characters. Strings are immutable in C#, meaning once created, they cannot be modified. Use strings when you do not need to frequently modify the content.
* **StringBuilder**: Use **StringBuilder** when you need to manipulate (append, insert, remove) strings frequently. **StringBuilder** is mutable and provides better performance in scenarios where there are multiple modifications to the string content.

1. What is the base class for all arrays in C#?

The base class for all arrays in C# is **System.Array**

1. How do you sort an array in C#?

**Array.Sort()** method

1. What property of an array object can be used to get the total number of elements in an array?

**Length**

1. Can you store multiple data types in System.Array?

Yes, you can store multiple data types in **System.Array** using the **object** type. Since **object** is the base type for all other types in C#, you can store any type of object in an array of type **object[]**

1. What’s the difference between the System.Array.CopyTo() and System.Array.Clone()?

* **Array.CopyTo(Array array, int index)**: Copies the entire contents of the array to another array starting at the specified index. The destination array must be of compatible type and have enough space to accommodate the elements being copied.
* **Array.Clone()**: Creates a shallow copy of the array. It returns a new array object with the same length and elements as the original array. Changes to the elements of the cloned array do not affect the original array, but changes to the objects referenced by the elements do affect both arrays (since they reference the same objects).

Copy array:

using System;

class Program

{

static void Main()

{

// Create an initial array with 10 items (example with integers)

int[] originalArray = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

// Create a second array with the same length as the original array

int[] copiedArray = new int[originalArray.Length];

// Copy elements from the original array to the copied array using a loop

for (int i = 0; i < originalArray.Length; i++)

{

copiedArray[i] = originalArray[i];

}

// Print the contents of both arrays

Console.WriteLine("Original Array:");

PrintArray(originalArray);

Console.WriteLine("\nCopied Array:");

PrintArray(copiedArray);

}

// Helper method to print the contents of an array

static void PrintArray(int[] arr)

{

foreach (var item in arr)

{

Console.Write(item + " ");

}

Console.WriteLine();

}

}

Manage a list of elements:

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

List<string> itemList = new List<string>(); // Initialize an empty list to store items

while (true) // Infinite loop to manage user commands

{

Console.WriteLine("Enter command (+ item, - item, or -- to clear):");

string input = Console.ReadLine().Trim(); // Read user input and trim whitespace

if (input.StartsWith("+")) // Add item

{

string itemToAdd = input.Substring(1).Trim(); // Extract item name after "+"

itemList.Add(itemToAdd);

}

else if (input.StartsWith("-")) // Remove item

{

string itemToRemove = input.Substring(1).Trim(); // Extract item name after "-"

itemList.Remove(itemToRemove);

}

else if (input.Equals("--")) // Clear list

{

itemList.Clear();

}

else // Invalid command

{

Console.WriteLine("Invalid command. Please use '+ item', '- item', or '--'.");

continue; // Restart the loop to prompt again

}

// Print current list contents

Console.WriteLine("\nCurrent List:");

foreach (var item in itemList)

{

Console.WriteLine("- " + item);

}

Console.WriteLine(); // Add a blank line for readability

}

}

}

Prime numbers:

using System;

using System.Collections.Generic;

class Program

{

static void Main()

{

int startNum = 1;

int endNum = 50;

int[] primes = FindPrimesInRange(startNum, endNum);

Console.WriteLine($"Prime numbers between {startNum} and {endNum}:");

foreach (var prime in primes)

{

Console.Write(prime + " ");

}

Console.WriteLine();

}

static int[] FindPrimesInRange(int startNum, int endNum)

{

List<int> primeList = new List<int>();

for (int num = startNum; num <= endNum; num++)

{

if (IsPrime(num))

{

primeList.Add(num);

}

}

return primeList.ToArray();

}

static bool IsPrime(int number)

{

if (number <= 1)

{

return false;

}

if (number == 2)

{

return true;

}

if (number % 2 == 0)

{

return false;

}

int boundary = (int)Math.Floor(Math.Sqrt(number));

for (int i = 3; i <= boundary; i += 2)

{

if (number % i == 0)

{

return false;

}

}

return true;

}

}

Rotation numbers:

using System;

class Program

{

static void Main()

{

// Read input array and k from the user

Console.WriteLine("Enter array of integers (space separated):");

int[] array = Array.ConvertAll(Console.ReadLine().Split(' '), int.Parse);

Console.WriteLine("Enter k (number of rotations):");

int k = int.Parse(Console.ReadLine());

// Initialize arrays to store rotated arrays and sum

int[][] rotatedArrays = new int[k][];

int[] sum = new int[array.Length];

// Perform rotations and calculate sum after each rotation

for (int r = 0; r < k; r++)

{

rotatedArrays[r] = RotateRight(array, r + 1); // Rotate array to the right r+1 times

for (int i = 0; i < array.Length; i++)

{

sum[i] += rotatedArrays[r][i]; // Accumulate sum of the rotated array

}

}

// Output results

for (int r = 0; r < k; r++)

{

Console.WriteLine($"rotated{r + 1}[] = {string.Join(" ", rotatedArrays[r])}");

}

Console.WriteLine($"sum[] = {string.Join(" ", sum)}");

}

// Method to rotate the array to the right r times

static int[] RotateRight(int[] array, int r)

{

int n = array.Length;

int[] rotated = new int[n];

for (int i = 0; i < n; i++)

{

int newIndex = (i + r) % n;

rotated[newIndex] = array[i];

}

return rotated;

}

}

Longest sequence of equal elements:

using System;

class Program

{

static void Main()

{

// Read input array

Console.WriteLine("Enter array of integers (space separated):");

int[] array = Array.ConvertAll(Console.ReadLine().Split(' '), int.Parse);

// Initialize variables to track longest sequence

int maxLength = 1; // Length of the longest sequence found so far

int maxStartIndex = 0; // Starting index of the longest sequence found

int currentLength = 1; // Length of the current sequence

int startIndex = 0; // Starting index of the current sequence

// Traverse through the array to find the longest sequence

for (int i = 1; i < array.Length; i++)

{

if (array[i] == array[i - 1])

{

currentLength++;

}

else

{

// Check if current sequence is longer than the previously recorded longest sequence

if (currentLength > maxLength)

{

maxLength = currentLength;

maxStartIndex = startIndex;

}

// Reset for the new sequence

startIndex = i;

currentLength = 1;

}

}

// Check the last sequence in case it's the longest

if (currentLength > maxLength)

{

maxLength = currentLength;

maxStartIndex = startIndex;

}

// Output the longest sequence found

Console.WriteLine("Longest sequence of equal elements:");

for (int i = maxStartIndex; i < maxStartIndex + maxLength; i++)

{

Console.Write(array[i] + " ");

}

Console.WriteLine();

}

}

Most frequent number:

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main()

{

// Read input sequence of numbers

Console.WriteLine("Enter sequence of numbers (space separated):");

int[] numbers = Array.ConvertAll(Console.ReadLine().Split(' '), int.Parse);

// Dictionary to count frequency of each number

Dictionary<int, int> frequencyMap = new Dictionary<int, int>();

// Populate frequency map

foreach (int num in numbers)

{

if (frequencyMap.ContainsKey(num))

{

frequencyMap[num]++;

}

else

{

frequencyMap[num] = 1;

}

}

// Variables to track the most frequent number

int mostFrequentNumber = 0;

int maxFrequency = 0;

// Find the most frequent number and handle ties

foreach (var kvp in frequencyMap)

{

if (kvp.Value > maxFrequency || (kvp.Value == maxFrequency && kvp.Key < mostFrequentNumber))

{

mostFrequentNumber = kvp.Key;

maxFrequency = kvp.Value;

}

}

// Output the result

Console.Write($"The number {mostFrequentNumber} is the most frequent (occurs {maxFrequency} times)");

}

}

Reverse string:

* 1. Using char

using System;

class Program

{

static void Main()

{

// Read input string from console

Console.WriteLine("Enter a string:");

string input = Console.ReadLine();

// Convert string to char array

char[] charArray = input.ToCharArray();

// Reverse the char array

Array.Reverse(charArray);

// Convert char array back to string

string reversedString = new string(charArray);

// Output the reversed string

Console.WriteLine($"Reversed string: {reversedString}");

}

}

* 1. Using for loop

using System;

class Program

{

static void Main()

{

// Read input string from console

Console.WriteLine("Enter a string:");

string input = Console.ReadLine();

// Output the string in reverse using a for loop

Console.Write("Reversed string: ");

for (int i = input.Length - 1; i >= 0; i--)

{

Console.Write(input[i]);

}

Console.WriteLine(); // to move to the next line after printing the reversed string

}

}

Reverse sentence, delete special characters:

using System;

using System.Text.RegularExpressions;

class Program

{

static void Main()

{

// Example sentences

string[] sentences = {

"C# is not C++, and PHP is not Delphi!",

"Delphi not is PHP, and C++ not is C#!",

"The quick brown fox jumps over the lazy dog /Yes! Really!!!/"

};

foreach (string sentence in sentences)

{

string reversedSentence = ReverseWordsInSentence(sentence);

Console.WriteLine(reversedSentence);

}

}

static string ReverseWordsInSentence(string sentence)

{

// Define separators that indicate word boundaries

string separatorsPattern = @"[.,:;=()&\[\]""'\\\/!? ]+"; // Add any additional separators as needed

// Split the sentence into words and separators

string[] wordsAndSeparators = Regex.Split(sentence, separatorsPattern);

// Reverse the array of words and separators

Array.Reverse(wordsAndSeparators);

// Join the reversed array back into a single string

string reversedSentence = string.Join("", wordsAndSeparators);

return reversedSentence;

}

}

Palindromes:

using System;

using System.Collections.Generic;

using System.Linq;

class Program

{

static void Main()

{

// Example input text

string inputText = "Hi,exe? ABBA! Hog fully a string: ExE. Bob";

// Find and print all unique palindromes from the input text

string palindromes = FindAndPrintPalindromes(inputText);

Console.WriteLine(palindromes);

}

static string FindAndPrintPalindromes(string text)

{

// Define characters to ignore (non-alphanumeric)

char[] ignoreChars = { ',', '.', '!', '?', ':', ' ', ';', '-' };

// Split text into words and clean each word from ignored characters

string[] words = text.Split(ignoreChars, StringSplitOptions.RemoveEmptyEntries);

// HashSet to store unique palindromes

HashSet<string> palindromeSet = new HashSet<string>();

// Check each word if it is a palindrome and add it to the HashSet

foreach (string word in words)

{

if (IsPalindrome(word))

{

palindromeSet.Add(word);

}

}

// Convert HashSet to sorted array

string[] sortedPalindromes = palindromeSet.OrderBy(p => p).ToArray();

// Join sorted palindromes with ", " separator

string result = string.Join(", ", sortedPalindromes);

return result;

}

static bool IsPalindrome(string word)

{

// Convert word to lowercase for case insensitivity

string lowerWord = word.ToLower();

// Check if the word is a palindrome

for (int i = 0; i < lowerWord.Length / 2; i++)

{

if (lowerWord[i] != lowerWord[lowerWord.Length - 1 - i])

{

return false;

}

}

return true;

}

}

Parses URL:

using System;

class Program

{

static void Main()

{

// Example URLs

string[] urls = {

"https://www.apple.com/iphone",

"ftp://www.example.com/employee",

"https://google.com",

"www.apple.com"

};

foreach (string url in urls)

{

ParseUrl(url);

}

}

static void ParseUrl(string url)

{

// Initialize variables to store protocol, server, and resource

string protocol = "";

string server = "";

string resource = "";

// Check if the URL contains "://"

int protocolSeparatorIndex = url.IndexOf("://");

if (protocolSeparatorIndex != -1)

{

// Extract protocol

protocol = url.Substring(0, protocolSeparatorIndex);

// Remove protocol part from URL

url = url.Substring(protocolSeparatorIndex + 3);

}

// Find the index of the next "/"

int resourceSeparatorIndex = url.IndexOf('/');

// If there is a "/", extract server and resource parts

if (resourceSeparatorIndex != -1)

{

server = url.Substring(0, resourceSeparatorIndex);

resource = url.Substring(resourceSeparatorIndex + 1);

}

else

{

// If no "/", the entire remaining string is the server

server = url;

}

// Output the parsed parts

Console.WriteLine($"[{protocol}] = \"{protocol}\"");

Console.WriteLine($"[{server}] = \"{server}\"");

Console.WriteLine($"[{resource}] = \"{resource}\"");

Console.WriteLine();

}

}