Answer 1:

String:

|  |
| --- |
|  |

|  |
| --- |
| Use when the text is **not frequently modified**. Strings are **immutable**, so every change (like concatenation) creates a **new object** in memory. |

StringBuffer:

Use when performing **many string manipulations** (e.g., loops, concatenations, appends). StringBuilder is **mutable**, meaning it modifies the existing object without creating new ones — improving performance.

Answer 2:

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

Answer 3:

You can sort an array using the **Array.Sort()** method from the System namespace.

Answer 4:

Use the **Length** property.

Answer 5:

**Yes**, but only if the array type is declared as object[].

Answer 6:

| **Feature** | **CopyTo()** | **Clone()** |
| --- | --- | --- |
| **Purpose** | Copies **elements** into an existing array | Creates a **new array instance** |
| **Syntax** | array1.CopyTo(array2, index); | array2 = (int[])array1.Clone(); |
| **Destination** | Requires an already initialized target array | Returns a brand new array |
| **Depth** | Shallow copy (copies references for reference types) | Shallow copy (same behavior) |

Arrays:

using System;

class Program1Arrays

{

static void Main()

{

int[] originalArray = new int[10];

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

{

originalArray[i] = (i + 1) \* 10;

}

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

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

{

copiedArray[i] = originalArray[i];

}

Console.WriteLine("Original Array: ");

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

{

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

}

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

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

{

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

}

Console.WriteLine();

}

}

Answer 2:

using System;

using System.Collections.Generic;

class Program1Arrays2

{

static void Main()

{

List<string> l = new List<string>();

while (true)

{

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

string s = Console.ReadLine();

if (string.IsNullOrWhiteSpace(s))

continue;

s = s.Trim();

if (s == "--")

{

l.Clear();

Console.WriteLine("List cleared.");

}

else if (s.StartsWith("+"))

{

string i = s.Substring(1).Trim();

if (!string.IsNullOrEmpty(i))

{

l.Add(i);

Console.WriteLine($"Added: {i}");

}

}

else if (s.StartsWith("-"))

{

string i = s.Substring(1).Trim();

if (l.Remove(i))

Console.WriteLine($"Removed: {i}");

else

Console.WriteLine($"Item '{i}' not found.");

}

else

{

Console.WriteLine("Invalid command. Use + item, - item, or --.");

}

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

if (l.Count == 0)

Console.WriteLine("(empty)");

else

foreach (string i in l)

Console.WriteLine("- " + i);

Console.WriteLine();

}

}

}

Answer3:

using System;

using System.Collections.Generic;

class Program1Arrays3

{

static void Main()

{

int s = 10;

int e = 50;

int[] p = F(s, e);

Console.WriteLine($"Prime numbers between {s} and {e}:");

Console.WriteLine(string.Join(", ", p));

}

static int[] F(int a, int b)

{

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

for (int n = a; n <= b; n++)

if (P(n)) l.Add(n);

return l.ToArray();

}

static bool P(int n)

{

if (n < 2) return false;

for (int i = 2; i <= Math.Sqrt(n); i++)

if (n % i == 0) return false;

return true;

}

}

Answer 4:

using System;

using System.Linq;

class Program1Arrays4

{

static void Main()

{

Console.Write("Enter array elements (space separated): ");

int[] arr = Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

Console.Write("Enter number of rotations: ");

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

int n = arr.Length;

int[] sum = new int[n];

int[] temp = new int[n];

Array.Copy(arr, temp, n);

for (int r = 1; r <= k; r++)

{

int last = temp[n - 1];

for (int i = n - 1; i > 0; i--)

{

temp[i] = temp[i - 1];

}

temp[0] = last;

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

{

sum[i] += temp[i];

}

}

Console.WriteLine("Sum after rotations:");

Console.WriteLine(string.Join(" ", sum));

}

}

Answer 5:

using System;

using System.Linq;

class Program1Arrays5

{

static void Main()

{

Console.Write("Enter array elements (space separated): ");

int[] arr = Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

int bestNum = arr[0];

int bestLen = 1;

int currentLen = 1;

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

{

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

{

currentLen++;

}

else

{

currentLen = 1;

}

if (currentLen > bestLen)

{

bestLen = currentLen;

bestNum = arr[i];

}

}

Console.WriteLine(string.Join(" ", Enumerable.Repeat(bestNum, bestLen)));

}

}

Answer 6:

using System;

using System.Collections.Generic;

using System.Linq;

class ProgramProgram1Arrays7

{

static void Main()

{

Console.Write("Enter numbers (space separated): ");

int[] arr = Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

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

foreach (int num in arr)

{

if (freq.ContainsKey(num))

freq[num]++;

else

freq[num] = 1;

}

int maxFreq = freq.Values.Max();

var mostFrequent = freq.Where(x => x.Value == maxFreq)

.Select(x => x.Key)

.ToList();

int leftmost = arr.First(x => mostFrequent.Contains(x));

if (mostFrequent.Count == 1)

{

Console.WriteLine($"The number {leftmost} is the most frequent (occurs {maxFreq} times).");

}

else

{

Console.WriteLine($"The numbers {string.Join(", ", mostFrequent)} have the same maximal frequency (each occurs {maxFreq} times).");

Console.WriteLine($"The leftmost of them is {leftmost}.");

}

}

}

Strings:

Answer 1:

using System;

class Program1String1

{

static void Main()

{

Console.Write("Enter text: ");

string s = Console.ReadLine();

char[] a = s.ToCharArray();

Array.Reverse(a);

string r1 = new string(a);

Console.WriteLine(r1);

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

Console.Write(s[i]);

}

}

Answer 2:

using System;

using System.Text.RegularExpressions;

class Program1String2

{

static void Main()

{

Console.Write("Enter sentence: ");

string s = Console.ReadLine();

string pattern = @"[.,:;=\(\)&\[\]""'\\/!?\s]";

string[] words = Regex.Split(s, pattern);

string[] seps = Regex.Split(s, @"[A-Za-z0-9**\+\-**]+");

Array.Reverse(words);

int wi = 0;

string res = "";

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

{

if (i < words.Length && words[i] != "")

res += words[wi++] + seps[i];

else

res += seps[i];

}

Console.WriteLine(res);

}

}

Answer 3:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text.RegularExpressions;

class Program1String3

{

static void Main()

{

Console.Write("Enter text: ");

string s = Console.ReadLine();

string[] w = Regex.Split(s, @"[^A-Za-z0-9]+");

HashSet<string> set = new HashSet<string>(StringComparer.OrdinalIgnoreCase);

foreach (string t in w)

{

if (t.Length > 1 && IsPal(t))

set.Add(t.ToLower());

}

var res = set.OrderBy(x => x);

Console.WriteLine(string.Join(", ", res));

}

static bool IsPal(string s)

{

int l = 0, r = s.Length - 1;

while (l < r)

{

if (char.ToLower(s[l]) != char.ToLower(s[r]))

return false;

l++;

r--;

}

return true;

}

}

Answer 4:

using System;

class Program1String4

{

static void Main()

{

Console.Write("Enter URL: ");

string url = Console.ReadLine();

string protocol = "";

string server = "";

string resource = "";

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

if (iProtocol != -1)

{

protocol = url.Substring(0, iProtocol);

url = url.Substring(iProtocol + 3);

}

int iSlash = url.IndexOf("/");

if (iSlash != -1)

{

server = url.Substring(0, iSlash);

resource = url.Substring(iSlash + 1);

}

else

{

server = url;

}

Console.WriteLine("[protocol] = " + (protocol == "" ? "(none)" : protocol));

Console.WriteLine("[server] = " + server);

Console.WriteLine("[resource] = " + (resource == "" ? "(none)" : resource));

}

}