02 Arrays and Strings

Test your Knowledge

1. When to use String vs. StringBuilder in C# ?
Use String → When the value is immutable (doesn't change often).
Use StringBuilder → When performing frequent modifications (appending, inserting, removing) to improve performance.

2. What is the base class for all arrays in C#?
System.Array is the base class for all arrays in C#.

3. How do you sort an array in C#?
Use Array.Sort() to sort an array in ascending order.

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

5. Can you store multiple data types in System.Array?
No, System.Array is strongly typed, meaning all elements must be of the same type. Use object[] if multiple data types are needed.

6. What's the difference between the System.Array.CopyTo() and System.Array.Clone()?
Array.CopyTo() copies elements into an existing array, requiring a destination array with enough space, while Array.Clone() creates a new independent copy of the array. CopyTo() performs a deep copy for value types but only copies references for reference types, whereas Clone() always performs a shallow copy, meaning it duplicates the structure but not the objects inside for reference types.

Practice Arrays

1. Copying an Array
Write code to create a copy of an array. First, start by creating an initial array. (You can use whatever type of data you want.) Let's start with 10 items. Declare an array variable and assign it a new array with 10 items in it. Use the things we've discussed to put some values in the array. Now create a second array variable. Give it a new array with the same length as the first. Instead of using a number for this length, use the Lengthproperty to get the size of the original array. Use a loop to read values from the original array and place them in the new array. Also print out the contents of both arrays, to be sure everything copied correctly.

```
using System;

class Program
{
   static void Main()
   {
      // Step 1: Create and initialize the original array
      int[] originalArray = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};

      // Step 2: Create a second array of the same length
      int[] copiedArray = new int[originalArray.Length];

      // Step 3: Copy values using a loop
      for (int i = 0; i < originalArray.Length; i++)
      {
         copiedArray[i] = originalArray[i];
      }

      // Step 4: Print both arrays to verify the copy
      Console.WriteLine("Original Array: " + string.Join(", ", originalArray));
      Console.WriteLine("Copied Array: " + string.Join(", ", copiedArray));
   }
}
```

2. Write a simple program that lets the user manage a list of elements. It can be a grocery list, "to do" list, etc. Refer to Looping Based on a Logical Expression if necessary to see how to implement an infinite loop. Each time through the loop, ask the user to perform an operation, and then show the current contents of their list. The operations available should be Add, Remove, and Clear. The syntax should be as follows:
+ some item
- some item
--
Your program should read in the user's input and determine if it begins with a “+” or “-“ or if it is simply “—“ . In the first two cases, your program should add or remove the string given ("some item" in the example). If the user enters just “—“ then the program should clear the current list. Your program can start each iteration through its loop with the following instruction:
Console.WriteLine("Enter command (+ item, - item, or -- to clear)):");

```
using System;
using System.Collections.Generic;

class Program
{
   static void Main()
   {
      List<string> itemList = new List<string>(); // List to store items

      while (true)
      {
         Console.WriteLine("Enter command (+ item, - item, or -- to clear, 'exit' to quit):");
         string input = Console.ReadLine().Trim();

         if (input.ToLower() == "exit") // Exit condition
            break;

         if (input == "--")
         {
            itemList.Clear();
            Console.WriteLine("List cleared.");
         }
         else if (input.StartsWith("+ "))
         {
```

```
            string item = input.Substring(2).Trim();
            if (!string.IsNullOrEmpty(item))
            {
                itemList.Add(item);
                Console.WriteLine($"Added: {item}");
            }
        }
        else if (input.StartsWith("- "))
        {
            string item = input.Substring(2).Trim();
            if (itemList.Remove(item))
                Console.WriteLine($"Removed: {item}");
            else
                Console.WriteLine($"Item '{item}' not found.");
        }
        else
        {
            Console.WriteLine("Invalid command. Use + item, - item, or --.");
        }

        // Display current list
        Console.WriteLine("Current List: " + (itemList.Count > 0 ? string.Join(", ",
itemList) : "Empty"));
    }
  }
}
```

3. Write a method that calculates all prime numbers in given range and returns them as array of integers

```
static int[] FindPrimesInRange(startNum, endNum)
{
}

class Program
{
    static void Main()
    {
        int startNum = 10, endNum = 50; // Example range
        int[] primes = FindPrimesInRange(startNum, endNum);

        Console.WriteLine("Prime Numbers: " + string.Join(", ", primes));
    }

    static int[] FindPrimesInRange(int startNum, int endNum)
    {
        List<int> primes = new List<int>();

        for (int num = Math.Max(startNum, 2); num <= endNum; num++)
        {
            if (IsPrime(num))
                primes.Add(num);
        }

        return primes.ToArray();
    }

    static bool IsPrime(int num)
    {
        if (num < 2) return false;
        for (int i = 2; i * i <= num; i++)
        {
            if (num % i == 0)
                return false;
        }
        return true;
    }
}
```

4. Write a program to read an array of n integers (space separated on a single line) and an integer k, rotate the array right k times and sum the obtained arrays after each rotation as shown below.
After r rotations the element at position I goes to position (I + r) % n.
The sum[] array can be calculated by two nested loops: for r = 1 ... k; for I = 0 ... n-1.
Input Output Comments
3 2 4 -1 3 2 5 6 rotated1[] = -1 3 2 4
2 rotated2[] = 4 -1 3 2
sum[] = 3 2 5 6
1 2 3 4 5 12 10 8 6 9 rotated1[] = 5 1 2 3 4
3 rotated2[] = 4 5 1 2 3
rotated3[] = 3 4 5 1 2
sum[] = 12 10 8 6 9

```
class Program
{
   static void Main()
   {
      // Read input array
      Console.Write("Enter space-separated integers: ");
      int[] arr = Console.ReadLine().Split().Select(int.Parse).ToArray();

      // Read number of rotations
      Console.Write("Enter number of rotations: ");
      int k = int.Parse(Console.ReadLine());

      int n = arr.Length;
      int[] sumArray = new int[n];

      // Perform k rotations
      for (int r = 1; r <= k; r++)
      {
         int[] rotatedArray = new int[n];

         // Rotate right by 1
         for (int i = 0; i < n; i++)
         {
            rotatedArray[(i + 1) % n] = arr[i];
         }
```

```
            // Update sum array
            for (int i = 0; i < n; i++)
            {
                sumArray[i] += rotatedArray[i];
            }

            // Set arr to rotated array for next rotation
            arr = rotatedArray;

            // Print rotated array
            Console.WriteLine($"Rotated {r}: " + string.Join(" ", rotatedArray));
        }

        // Print sum array
        Console.WriteLine("Sum Array: " + string.Join(" ", sumArray));
    }
}
```

5. Write a program that finds the longest sequence of equal elements in an array of integers. If several longest sequences exist, print the leftmost one.
Input Output
2 1 1 2 3 3 2 2 2 1 2 2 2
1 1 1 2 3 1 3 3 1 1 1
4 4 4 4 4 4 4 4
0 1 1 5 2 2 6 3 3 1 1

```
class Program
{
    static void Main()
    {
        // Read input array
        Console.Write("Enter space-separated integers: ");
        int[] arr = Console.ReadLine().Split().Select(int.Parse).ToArray();

        int maxLength = 1, currentLength = 1, startIndex = 0, bestStart = 0;

        // Iterate through the array to find the longest sequence
        for (int i = 1; i < arr.Length; i++)
        {
            if (arr[i] == arr[i - 1])
                currentLength++;
            else
                currentLength = 1;

            if (currentLength > maxLength)
            {
                maxLength = currentLength;
                bestStart = i - maxLength + 1;
            }
        }

        // Extract and print the longest sequence
        int[] longestSequence = arr.Skip(bestStart).Take(maxLength).ToArray();
        Console.WriteLine("Longest Sequence: " + string.Join(" ", longestSequence));
    }
}
```

7. Write a program that finds the most frequent number in a given sequence of numbers. In case of multiple numbers with the same maximal frequency, print the leftmost of them

Input Output
4 1 1 4 2 3 4 4 1 2 4 9 3 The number 4 is the most frequent (occurs 5 times)
7 7 7 0 2 2 2 0 10 10 10 The numbers 2, 7 and 10 have the same maximal frequence (each occurs 3 times). The leftmost of them is 7.

```
class Program
{
    static void Main()
    {
        // Read input and convert to integer array
        Console.Write("Enter space-separated numbers: ");
        int[] numbers = Console.ReadLine().Split().Select(int.Parse).ToArray();

        Dictionary<int, int> frequency = new Dictionary<int, int>();
        int maxCount = 0, mostFrequent = numbers[0];

        // Count frequency and track the leftmost most frequent number
        foreach (int num in numbers)
        {
            if (!frequency.ContainsKey(num))
                frequency[num] = 0;
            frequency[num]++;

            if (frequency[num] > maxCount)
            {
                maxCount = frequency[num];
                mostFrequent = num; // Update most frequent number
            }
        }

        // Output result
        Console.WriteLine($"The number {mostFrequent} is the most frequent (occurs
{maxCount} times).");
    }
}
```

Practice Strings

1. Write a program that reads a string from the console, reverses its letters and prints the result back at the console.

Write in two ways Convert the string to char array, reverse it, then convert it to string again Print the letters of the string in back direction (from the last to the first) in a for-loop

Input Output
sample elpmas
24tvcoi92 29iocvt42

```
class Program
{
    static void Main()
    {
        // Read input string
        Console.Write("Enter a string: ");
        string input = Console.ReadLine();

        // Method 1: Using char array
        char[] charArray = input.ToCharArray();
        Array.Reverse(charArray);
        string reversed1 = new string(charArray);
        Console.WriteLine("Reversed (Method 1): " + reversed1);

        // Method 2: Using for-loop
        string reversed2 = "";
        for (int i = input.Length - 1; i >= 0; i--)
        {
            reversed2 += input[i];
        }
        Console.WriteLine("Reversed (Method 2): " + reversed2);
    }
}
```

2. Write a program that reverses the words in a given sentence without changing the punctuation and spaces
Use the following separators between the words: . , : ; = ( ) & [ ] " ' \ / ! ? (space).
All other characters are considered part of words, e.g. C++, a+b, and a77 are considered valid words.
The sentences always start by word and end by separator.

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

```
class Program
{
   static void Main()
   {
      // Read input sentence
      Console.Write("Enter a sentence: ");
      string input = Console.ReadLine();

      // Define separators (punctuation & spaces)
      string separators = @"[.,:;=\(\)&\[\]""'\\\/!? ]+";

      // Extract words and separators
      string[] words = Regex.Split(input, separators);
      MatchCollection punctuationMatches = Regex.Matches(input, separators);

      // Reverse only words, keep punctuation order
      List<string> reversedWords = new List<string>();
      foreach (string word in words)
      {
         if (!string.IsNullOrEmpty(word))
            reversedWords.Insert(0, word);
      }

      // Reconstruct sentence
      StringBuilder result = new StringBuilder();
      int wordIndex = 0, punctuationIndex = 0;
```

```
        foreach (Match match in punctuationMatches)
        {
            if (wordIndex < reversedWords.Count)
                result.Append(reversedWords[wordIndex++] + match.Value);
            else
                result.Append(match.Value);
        }

        // Print result
        Console.WriteLine("Reversed Sentence: " + result.ToString().Trim());
    }
}
```

3. Write a program that extracts from a given text all palindromes, e.g. “ABBA”, “lamal”, “exe”and prints them on the console on a single line, separated by comma and space.Print all unique palindromes (no duplicates), sorted

Hi,exe? ABBA! Hog fully a string: ExE. Bob
a, ABBA, exe, ExE

```
class Program
{
   static void Main()
   {
      // Input text
      Console.Write("Enter text: ");
      string input = Console.ReadLine();

      // Extract words using Regex (ignores punctuation but keeps original casing)
      string[] words = Regex.Split(input, @"\W+").Where(w =>
!string.IsNullOrEmpty(w)).ToArray();

      // Find unique palindromes (case-sensitive)
      SortedSet<string> palindromes = new SortedSet<string>();

      foreach (string word in words)
      {
          if (IsPalindrome(word))
              palindromes.Add(word); // SortedSet ensures uniqueness & sorting
      }

      // Print palindromes
      Console.WriteLine("Palindromes: " + string.Join(", ", palindromes));
   }

   // Method to check if a word is a palindrome (includes single-character words)
   static bool IsPalindrome(string word)
   {
      return word.SequenceEqual(word.Reverse());
   }
}
```

4. Write a program that parses an URL given in the following format:
[protocol]://[server]/[resource]
The parsing extracts its parts: protocol, server and resource.
The [server] part is mandatory.
The [protocol] and [resource] parts are optional.
https://www.apple.com/iphone
[protocol] = "https"
[server] = "www.apple.com"
[resource] = "iphone"
ftp://www.example.com/employee
[protocol] = "ftp"
[server] = "www.example.com"
[resource] = "employee"
https://google.com
[protocol] = "https"
[server] = "google.com"
[resource] = ""
www.apple.com
[protocol] = ""
[server] = "www.apple.com"
[resource] = ""

```
class Program
{
   static void Main()
   {
      // Read input URL
      Console.Write("Enter URL: ");
      string url = Console.ReadLine();

      // Initialize parts
      string protocol = "", server = "", resource = "";

      // Check if protocol exists (contains "://")
      int protocolEndIndex = url.IndexOf("://");
      if (protocolEndIndex != -1)
      {
         protocol = url.Substring(0, protocolEndIndex);
         url = url.Substring(protocolEndIndex + 3); // Remove protocol part
      }
```

```
        // Extract server and resource
        int resourceStartIndex = url.IndexOf("/");
        if (resourceStartIndex != -1)
        {
            server = url.Substring(0, resourceStartIndex);
            resource = url.Substring(resourceStartIndex + 1); // Extract resource
        }
        else
        {
            server = url; // No resource, entire remaining string is server
        }

        // Output parsed values
        Console.WriteLine($"[protocol] = \"{protocol}\"");
        Console.WriteLine($"[server] = \"{server}\"");
        Console.WriteLine($"[resource] = \"{resource}\"");
    }
}
```