C# Coding Questions For Technical Interviews

Introduction

In this article, we will learn about some of the frequently asked C# programming questions in technical interviews.

Note: We won't be using any inbuilt functions such as Reverse, Substring etc. for string manipulation, also we will avoid using LINQ as these are generally restricted to be used in coding interviews.

Source Code

Download the source code for all the questions from GitHub

Q.1: How to reverse a string?

Ans.: The user will input a string and the method should return the reverse of that string

- input: hello, output: olleh
- input: hello world, output: dlrow olleh

```
internal static void ReverseString(string str)
{

    char[] charArray = str.ToCharArray();
    for (int i = 0, j = str.Length - 1; i < j; i++, j--)
    {
        charArray[i] = str[j];
        charArray[j] = str[i];
    }
    string reversedstring = new string(charArray);
    Console.WriteLine(reversedstring);
}
```

Q.2: How to find if the given string is a palindrome or not?

Ans.: The user will input a string and we need to print "*Palindrome*" or "*Not Palindrome*" based on whether the input string is a palindrome or not.

- input: madam, output: Palindrome
- input: step on no pets, output: Palindrome
- input: book, output: Not Palindrome

if we pass an integer as a string parameter then also this method will give the correct output

- input: 1221, output: Palindrome

```
internal static void chkPalindrome(string str)
 {
     bool flag = false;
     for (int i = 0, j = str.Length - 1; i < str.Length / 2; i++, j--)
     {
         if (str[i] != str[j])
         {
             flag = false;
             break;
         }
         else
             flag = true;
     }
     if (flag)
     {
         Console.WriteLine("Palindrome");
```

```
        }
        else
            Console.WriteLine("Not Palindrome");
    }
```

Q.3: How to reverse the order of words in a given string?

Ans.: The user will input a sentence and we need to reverse the sequence of words in the sentence.

- input: Welcome to Csharp corner, output: corner Csharp to Welcome

```
internal static void ReverseWordOrder(string str)
{
    int i;
    StringBuilder reverseSentence = new StringBuilder();

    int Start = str.Length - 1;
    int End = str.Length - 1;

    while (Start > 0)
    {
        if (str[Start] == ' ')
        {
            i = Start + 1;
            while (i <= End)
            {
                reverseSentence.Append(str[i]);
                i++;
            }
            reverseSentence.Append(' ');
            End = Start - 1;
        }
        Start--;
    }

    for (i = 0; i <= End; i++)
    {
        reverseSentence.Append(str[i]);
    }
    Console.WriteLine(reverseSentence.ToString());
}
```

Q.4: How to reverse each word in a given string?

Ans.: The user will input a sentence and we need to reverse each word individually without changing its position in the sentence.

- input: Welcome to Csharp corner, output: emocleW ot prahsC renroc

```
internal static void ReverseWords(string str)
 {
     StringBuilder output = new StringBuilder();
     List<char> charlist = new List<char>();

     for (int i = 0; i < str.Length; i++)
     {
         if (str[i] == ' ' || i == str.Length - 1)
         {
             if (i == str.Length - 1)
                 charlist.Add(str[i]);
             for (int j = charlist.Count - 1; j >= 0; j--)
                 output.Append(charlist[j]);

             output.Append(' ');
             charlist = new List<char>();
```

```
            }
            else
                charlist.Add(str[i]);
        }
        Console.WriteLine(output.ToString());
    }
```

Q.5: How to count the occurrence of each character in a string?

Ans.: The user will input a string and we need to find the count of each character of the string and display it on console. We won't be counting space character.

- input: hello world;

output:

h – 1

e – 1

l – 3

o – 2

w – 1

r – 1

d – 1

```
internal static void Countcharacter(string str)
{
    Dictionary<char, int> characterCount = new Dictionary<char, int>();

    foreach (var character in str)
    {
        if (character != ' ')
        {
            if (!characterCount.ContainsKey(character))
            {
                characterCount.Add(character, 1);
            }
            else
            {
                characterCount[character]++;
            }
        }

    }
    foreach (var character in characterCount)
    {
        Console.WriteLine("{0} - {1}", character.Key, character.Value);
    }
}
```

Q.6: How to remove duplicate characters from a string?

Ans.: The user will input a string and the method should remove multiple occurrences of characters in the string

- input: csharpcorner, output: csharpone

```
internal static void removeduplicate(string str)
 {
     string result = string.Empty;

     for (int i = 0; i < str.Length; i++)
     {
         if (!result.Contains(str[i]))
```

```
            {
                result += str[i];
            }
        }
        Console.WriteLine(result);
    }
```

Q.7: How to find all possible substring of a given string?

Ans.: This is a very frequent interview question. Here we need to form all the possible substrings from input string, varying from length 1 to the input string length. The output will include the input string also.

- input: abcd , output : a ab abc abcd b bc bcd c cd d

```
internal static void findallsubstring(string str)
  {
      for (int i = 0; i < str.Length; ++i)
      {
          StringBuilder subString = new StringBuilder(str.Length - i);
          for (int j = i; j < str.Length; ++j)
          {
              subString.Append(str[j]);
              Console.Write(subString + " ");
          }
      }
  }
```

Q.8: How to perform Left circular rotation of an array?

Ans.: The user will input an integer array and the method should shift each element of input array to its Left by one position in circular fashion. The logic is to iterate loop from Length-1 to 0 and swap each element with last element.

- input: 1 2 3 4 5, output: 2 3 4 5 1

```
internal static void RotateLeft(int[] array)
  {
      int size = array.Length;
      int temp;
      for (int j = size - 1; j > 0; j--)
      {
          temp = array[size - 1];
          array[array.Length - 1] = array[j - 1];
          array[j - 1] = temp;
      }

      foreach (int num in array)
      {
          Console.Write(num + " ");
      }
  }
```

Q.9: How to perform Right circular rotation of an array?

Ans: The user will input an integer array and the method should shift each element of input array to its Right by one position in circular fashion. The logic is to iterate loop from 0 to Length-1 and swap each element with first element

- input: 1 2 3 4 5, output: 5 1 2 3 4

```
internal static void RotateRight(int[] array)
{
    int size = array.Length;
    int temp;
    for (int j = 0; j < size - 1; j++)
    {
        temp = array[0];
        array[0] = array[j + 1];
```

```
            array[j + 1] = temp;
        }
        foreach (int num in array)
        {
            Console.Write(num + " ");
        }
    }
```

Q.10: How to find if a positive integer is a prime number or not?

Ans.: The user will input a positive integer and the method should output "*Prime*" or "*Not Prime*" based on whether the input integer is a prime number or not.

The logic is to find a positive integer less than or equal to the square root of input integer. If there is a divisor of number that is less than the square root of number, then there will be a divisor of number that is greater than square root of number. Hence, we have to traverse till the square root of number.

The time complexity of this function is O(√N) because we traverse from 1 to √N.

- input: 20, output: Not Prime
- input: 17, output: Prime

```
static void Main(string[] args)
  {
      if (FindPrime(47))
      {
          Console.WriteLine("Prime");
      }
      else
      {
          Console.WriteLine("Not Prime");
      }
      Console.ReadLine();
  }

internal static bool FindPrime(int number)
  {
      if (number == 1) return false;
      if (number == 2) return true;
      if (number % 2 == 0) return false;

      var squareRoot = (int)Math.Floor(Math.Sqrt(number));

      for (int i = 3; i <= squareRoot; i += 2)
      {
          if (number % i == 0) return false;
      }

      return true;
  }
```

Q.11: How to find the sum of digits of a positive integer?

Ans.: The user will input a positive integer and the method should return the sum of all the digits in that integer.

- input: 168, output: 15

```
internal static void SumOfDigits(int num)
{
    int sum = 0;
    while (num > 0)
    {
        sum += num % 10;
        num /= 10;
    }
    Console.WriteLine(sum);
}
```

Q.12: How to find second largest integer in an array using only one loop?

Ans.: The user will input an unsorted integer array and the method should find the second largest integer in the array.

- input: 3 2 1 5 4, output: 4

```
internal static void FindSecondLargeInArray(int[] arr)
{
    int max1 = int.MinValue;
    int max2 = int.MinValue;
    foreach (int i in arr)
    {
        if (i > max1)
        {
            max2 = max1;
            max1 = i;
        }
        else if (i >= max2 && i != max1)
        {
            max2 = i;
        }
    }
    Console.WriteLine(max2); ;
}
```

Q.13: How to find third largest integer in an array using only one loop?

Ans.: The user will input an unsorted integer array and the method should find the third largest integer in the array.

- input: 3 2 1 5 4, output: 3

```
internal static void FindthirdLargeInArray(int[] arr)
{
    int max1 = int.MinValue;
    int max2 = int.MinValue;
    int max3 = int.MinValue;

    foreach (int i in arr)
    {
        if (i > max1)
        {
            max3 = max2;
            max2 = max1;
            max1 = i;
        }
        else if (i > max2 && i != max1)
        {
            max3 = max2;
            max2 = i;
        }
        else if (i > max3 && i != max2 && i != max1)
        {
            max3 = i;
        }
    }
    Console.WriteLine(max3); ;
}
```

Q.14: How to convert a two-dimensional array to a one-dimensional array?

Ans.: The user will input a 2-D array (matrix) and we need to convert it to a 1-D array. We will create a 1-D array column-wise.

- input: { { 1, 2, 3 }, { 4, 5, 6 } }, output: 1 4 2 5 3 6

```
internal static void MultiToSingle(int[,] array)
{
    int index = 0;
    int width = array.GetLength(0);
    int height = array.GetLength(1);
    int[] single = new int[width * height];
    for (int y = 0; y < height; y++)
    {
        for (int x = 0; x < width; x++)
        {
            single[index] = array[x, y];
            Console.Write(single[index] + " ");
            index++;
        }
    }
}
```

This question can also be asked to form a 1-D array row-wise. In this case, just swap the sequence of the for loops as shown below. The output will be 1 2 3 4 5 6 for the input matrix mentioned above.

```
for (int x = 0; x < width; x++ )
 {
     for ( int y = 0; y < height; y++)
     {
         single[index] = array[x, y];
         Console.Write(single[index] + " ");
         index++;
     }
 }
```

Q.15: How to convert a one-dimensional array to a two-dimensional array?

Ans.: The user will input a 1-D array along with the number of rows and columns. The method should convert this 1-D array to a 2-D array(matrix) of a given row and column. We will create a matrix row-wise.

- input: {1, 2, 3, 4, 5, 6} ,2 ,3
- output:

1 2 3

4 5 6

```
internal static void SingleToMulti(int[] array, int row, int column)
 {
     int index = 0;
     int[,] multi = new int[row, column];

     for (int y = 0; y < row; y++)
     {
         for (int x = 0; x < column; x++)
         {
             multi[y, x] = array[index];
             index++;
             Console.Write(multi[y, x] + " ");
         }
         Console.WriteLine();
     }
 }
```

Q.16: How to find the angle between hour and minute hands of a clock at any given time?

Ans.: The user will input the hour and minute of the time and the method should give the angle between the hour hand and minute hand at that given time.

- input: 9 30, output: The angle between hour hand and minute hand is 105 degrees
- input: 13 30, output: The angle between hour hand and minute hand is 135 degrees

The logic is to find the difference in the angle of an hour and minute hand from the position of 12 O Clock when the angle between them is zero. Each hour on the clock represents an angle of 30 degrees (360 divided by 12). Similarly, each minute on the clock will represent an angle of 6 degrees (360 divided by 60) and the angle for an hour will increase as the minutes for that hour increases.

So, our code will be as follows:

```
internal static void FindAngleinTime(int hours, int mins)
 {
     double hourDegrees = (hours * 30) + (mins * 30.0 / 60);
     double minuteDegrees = mins * 6;

     double diff = Math.Abs(hourDegrees - minuteDegrees);

     if (diff > 180)
     {
         diff = 360 - diff;
     }

     Console.WriteLine("The angle between hour hand and minute hand is {0} degrees", diff);
 }
```

Q.17: Explain Bubble Sort Algorithm In C#

This algorithm follows the concept of iterating through the array from the first index to the last index and comparing adjacent elements and then swapping them if they appear in the wrong order. i.e. If the next element is smaller than the current element, they are swapped.

The Time Complexity of bubble sort is **O(n^2)**.

Read Bubble Sort Algorithm In C# to learn more.

Q.18: Explain Quick Sort Algorithm In C#

This Algorithm selects an element as a pivot element from the given array and partitions the array around it such that, Left side of the pivot contains all the elements that are less than the pivot element. The right side contains all elements that are greater than the pivot element. Let P be the index of pivot after partitioning the array. Then, the left subarray(start to P-1) and right subarray(P+1 to end) are sorted recursively to get the final sorted array as output.

The worst-case time complexity of this algorithm is **O(N^2)**. The best-case and average-case time complexity of this algorithm is **O(NLogN)**.

Read Quick Sort Algorithm In C# to learn more.

Q.19: Explain Merge Sort Algorithm In C#

This algorithm works as follows.

- Divide the unsorted array of size N into N subarrays having single element each.
- Take two adjacent subarrays and merge them to form a sorted subarray having 2 elements.Now we have N/2 subarrays of size 2.
- Repeat the process until a single sorted array is obtained.

In all three cases (worst, average, best), the time complexity of Merge sort is **O(NLogN)**

Read Merge Sort Algorithm In C# to learn more.

Q. 20: Explain Insertion Sort Algorithm In C#

Insertion sort compares the current element with the largest value in the sorted array. If the current element is smaller then the algorithm finds its correct position in a sorted array and moves the element to that position otherwise if the current element is greater then it leaves the element in its place and moves on to the next element.

To place the element in its correct position in a sorted array, all the elements larger than the current element are shifted one place ahead. Thus the sorted array will grow at each iteration.

Every element is compared to every other element of a sorted array. Hence the complexity of Insertion sort is **$O(n^2)$**.

Read Insertion Sort Algorithm In C# to learn more.

Q. 21: Explain Selection Sort Algorithm In C#

This algorithm follows the concept of dividing the given array into two subarrays.

1. The subarray of sorted elements
2. The subarray of unsorted elements

The algorithm will find the minimum element in the unsorted subarray and put it into its correct position in the sorted subarray.

Read Selection Sort Algorithm In C# to learn more.

The time complexity of Selection Sort as **$O(n^2)$.**

Q. 22: Explain Binary Search In C#

Binary search is an efficient and commonly used searching algorithm. This algorithm works only on sorted sets of elements. So if the given array is not sorted then we need to sort it before applying Binary search.

This algorithm searches a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty. If found return the index of matched element, else return -1.

Read Searching Algorithms In C# to learn more.

The array to be searched is reduced by half in every iteration. Hence time complexity of the Binary search is **O(LogN)**.