2100030320

M. Bhavana

EPAM - LAB 7

TASK1: To create generic type **CustomArray** – one dimensional array with random index range

CustomArray is a collection – array of random type values with fixed length and with original index that is specified by user.

Example 1: array of 20 elements length, array values – symbols, index starts with 18.
Example 2: array of 10 elements length, array values – objects of class Animals, index starts with -5

Values of random type can be located in array, custom first index and the number of elements in array should be specified while creating. The length and range of indexes cannot be changed after creating. Values of array elements can be set while creating the array and later with the help of indexer.

Initial and finite index, array length, array elements in the form of standard array Array that starts with 0 can be obtained from array.

CustomArray should be able to use operator foreach and other constructions that are oriented to the presence of numerator in class.

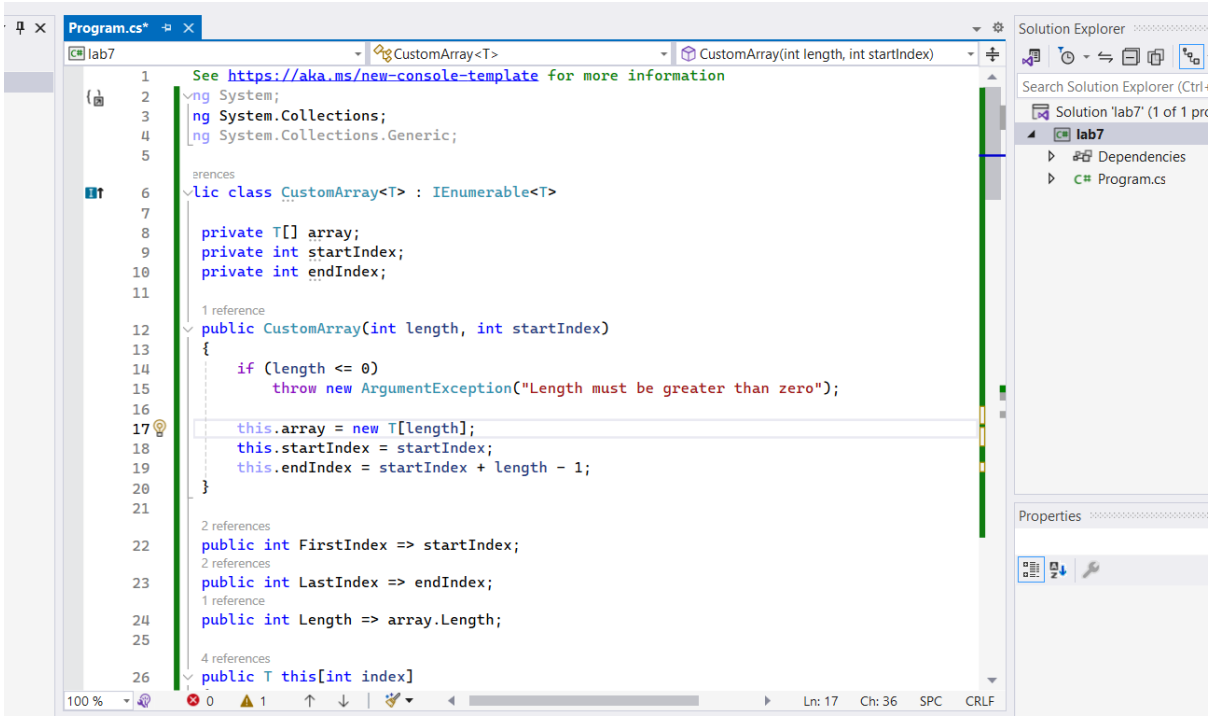
The task has two levels of complexity: Low and Advanced.

Low level tasks require implementation of the following functionality:

- Creating of empty user array and the one based on standard existing
- Receiving first, last indexes, length, values in form of standard array with 0.
- Access to writing and reading element based on predetermined correct index

Advanced level tasks require implementation of the following functionality:

- All completed tasks of Low level
- Creating of array based on values params
- Generating exceptions, specified in xml-comments to class methods
- Receiving numerator from array for operator foreach




```
See https://aka.ms/new-console-template for more information
ng System;
ng System.Collections;
ng System.Collections.Generic;

erences
lic class CustomArray<T> : IEnumerable<T>

 private T[] array;
 private int startIndex;
 private int endIndex;

 1 reference
 public CustomArray(int length, int startIndex)
 {
     if (length <= 0)
         throw new ArgumentException("Length must be greater than zero");

     this.array = new T[length];
     this.startIndex = startIndex;
     this.endIndex = startIndex + length - 1;
 }

 2 references
 public int FirstIndex => startIndex;
 2 references
 public int LastIndex => endIndex;
 1 reference
 public int Length => array.Length;

 4 references
 public T this[int index]
```

lab7 | CustomArray<T> | CustomArray(int length, int startIndex)

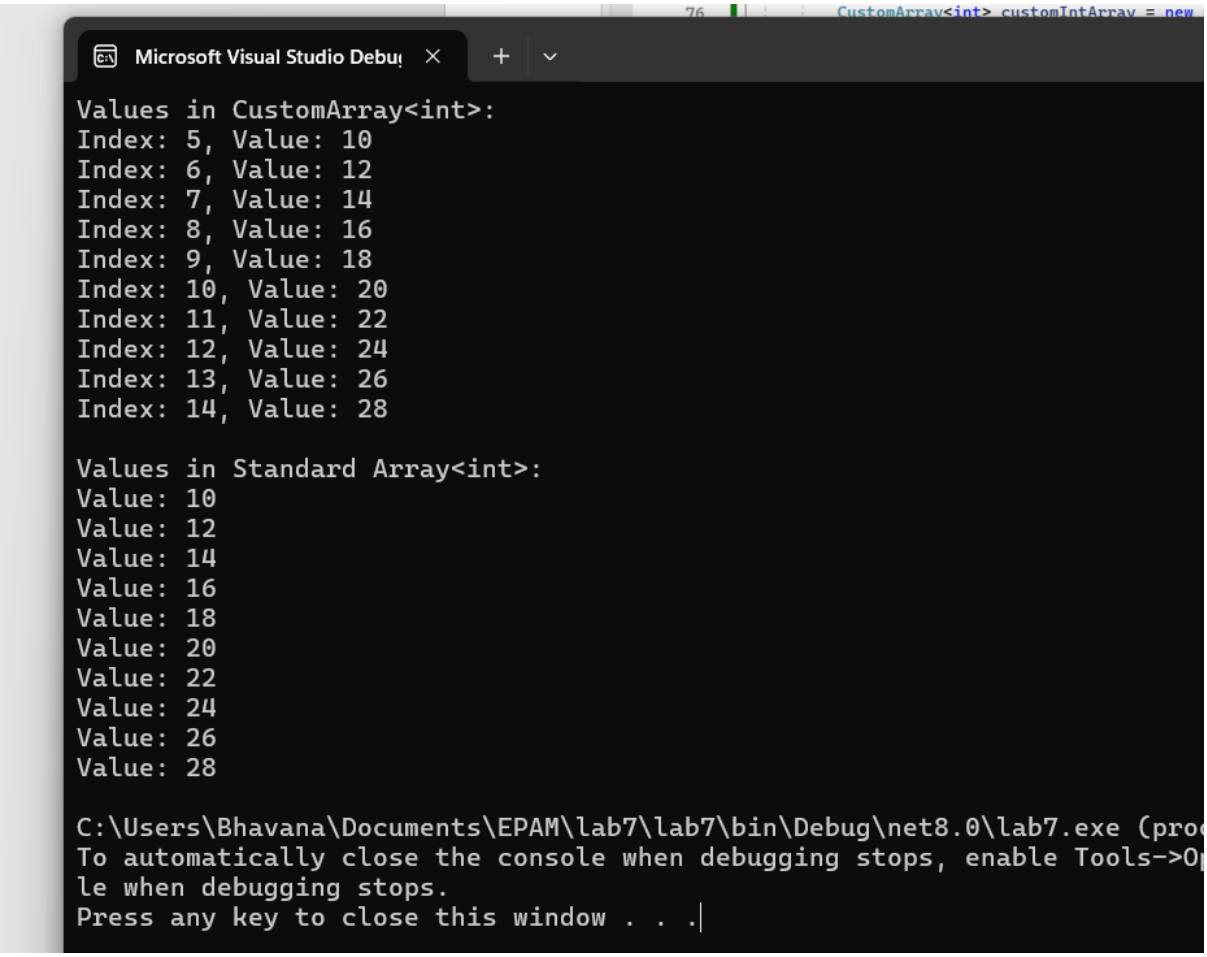
```
public T this[int index]
{
    get
    {
        if (!IsValidIndex(index))
            throw new IndexOutOfRangeException("Index is out of range");
        return array[index - startIndex];
    }
    set
    {
        if (!IsValidIndex(index))
            throw new IndexOutOfRangeException("Index is out of range");
        array[index - startIndex] = value;
    }
}

1 reference
public T[] ToStandardArray()
{
    T[] standardArray = new T[Length];
    for (int i = startIndex; i <= endIndex; i++)
    {
        standardArray[i - startIndex] = this[i];
    }
    return standardArray;
}

2 references
private bool IsValidIndex(int index)
{
    return index >= startIndex && index <= endIndex;
```

Program.cs*

lab7 | CustomArray<T> | CustomArray(int length, int startIndex)

```
54          return index >= startIndex && index <= endIndex;
55      }
56
        1 reference
57      public IEnumerator<T> GetEnumerator()
58      {
59          for (int i = startIndex; i <= endIndex; i++)
60          {
61              yield return this[i];
62          }
63      }
64
        0 references
65      IEnumerator IEnumerable.GetEnumerator()
66      {
67          return GetEnumerator();
68      }
69
        erences
70      lic class main {
71
        0 references
72      static void Main(string[] args)
73      {
74          try
75          {
76              CustomArray<int> customIntArray = new CustomArray<int>(10, 5);
77
78              for (int i = customIntArray.FirstIndex; i <= customIntArray.LastIndex; i++)
79              {
80                  customIntArray[i] = i * 2;
81              }
82
83              Console.WriteLine("Values in CustomArray<int>:");
84              for (int i = customIntArray.FirstIndex; i <= customIntArray.LastIndex; i++)
85              {
86                  Console.WriteLine($"Index: {i}, Value: {customIntArray[i]}");
87              }
88
89              int[] standardIntArray = customIntArray.ToStandardArray();
90              Console.WriteLine("\nValues in Standard Array<int>:");
91              foreach (var value in standardIntArray)
92              {
93                  Console.WriteLine($"Value: {value}");
94              }
95          }
96          catch (Exception ex)
97          {
98              Console.WriteLine($"An error occurred: {ex.Message}");
99          }
100     }
101
102
103
```

Output:


Microsoft Visual Studio Debu
Values in CustomArray<int>:
Index: 5, Value: 10
Index: 6, Value: 12
Index: 7, Value: 14
Index: 8, Value: 16
Index: 9, Value: 18
Index: 10, Value: 20
Index: 11, Value: 22
Index: 12, Value: 24
Index: 13, Value: 26
Index: 14, Value: 28
Values in Standard Array<int>:
Value: 10
Value: 12
Value: 14
Value: 16
Value: 18
Value: 20
Value: 22
Value: 24
Value: 26
Value: 28
C:\Users\Bhavana\Documents\EPAM\lab7\lab7\bin\Debug\net8.0\lab7.exe (proc
To automatically close the console when debugging stops, enable Tools->Op
le when debugging stops.
Press any key to close this window . . .