Brandon Girard

02 Arrays and Strings

**Test your Knowledge**

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

Use string when dealing with changing string values, use string builder when manipulating strings.

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

Array

**3.How do you sort an array in C#?**

Array.Sort()

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

Length

**5.Can you store multiple data types in System.Array?**

No

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

Clone() returns a new array object with all the elements of the copied array, CopyTo() copies elements from one array to an existing array.

**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 Length property 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.**

int[] array1 = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

int[] array2 = new int[array1.Length];

for(int i = 0; i < array1.Length; i++)

{

array2[i] = array1[i];

Console.WriteLine($"array1:{array1[i]} array2:{array2[i]}");

}

**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)):")

;

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

while (true)

{

//instructions

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

//get input

string input = Console.ReadLine();

if (input is null || input.Length == 0 || input.Equals(" "))

{

Console.WriteLine("Empty Input");

continue;

}

if (input.Equals("QQ")) break;

//parse input

string op = input.Substring(0, 2);

string item = input.Substring(2);

switch (op)

{

case "+ ":

if(item.Length > 0)

todoList.Add(item);

break;

case "- ":

if (item.Length > 0)

todoList.Remove(item);

break;

case "--":

todoList.Clear();

break;

default:

Console.WriteLine("Invalid Input");

continue;

}

//show list

Console.WriteLine("\nTODO----------------------------------");

foreach(string i in todoList)

Console.WriteLine(i);

Console.WriteLine("--------------------------------------\n");

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

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

{

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

if(startNum < 1 || endNum < 1 || endNum < startNum) return new int[0];

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

{

int mid = startNum / 2;

bool isPrime = true;

for(int i = 2; i <= mid; i++)

{

if (startNum % i == 0)

{

isPrime = false;

break;

}

}

if(isPrime) primes.Add(startNum);

}

return primes.ToArray();

}

**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 l 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

Console.WriteLine("Enter numbers");

string numInput = Console.ReadLine();

string[] splitArray = numInput.Split(" ");

Console.WriteLine("Enter rotations");

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

int[] rotIter = new int[splitArray.Length];

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

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

{

rotIter[i] = int.Parse(splitArray[i]);

sum[i] = 0;

}

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

{

int[] newRot = new int[rotIter.Length];

for(int l = 0; l < rotIter.Length; l++)

newRot[(l + r) % rotIter.Length] = rotIter[l];

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

sum[i] += newRot[i];

Console.WriteLine("rotated" + r + " " + getArrString(newRot));

}

Console.WriteLine("sum " + getArrString(sum));

string getArrString(int[] a)

{

string ret = "";

foreach (int i in a)

ret += i + " ";

return ret;

}

**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

int longestNum, sequenceLength = 0;

int currentNum, currentSequenceLength = 0;

Console.WriteLine("Enter numbers");

string numInput = Console.ReadLine();

string[] splitArray = numInput.Split(" ");

int[] numArray = new int[splitArray.Length];

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

numArray[i] = int.Parse(splitArray[i]);

longestNum = currentNum = numArray[0];

sequenceLength = currentSequenceLength = 1;

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

{

if(currentNum != numArray[i])

{

currentNum = numArray[i];

currentSequenceLength = 1;

}

else currentSequenceLength++;

if(currentSequenceLength > sequenceLength)

{

longestNum = currentNum;

sequenceLength = currentSequenceLength;

}

}

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

Console.Write(longestNum + " ");

Console.WriteLine();

**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**

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

Console.WriteLine("Enter numbers");

string numInput = Console.ReadLine();

string[] splitArray = numInput.Split(" ");

int mostFrequentNum = int.Parse(splitArray[0]);

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

{

int n = int.Parse(splitArray[i]);

if (frequencies.ContainsKey(n)) frequencies[n]++;

else frequencies.Add(n, 1);

}

foreach (int n in frequencies.Keys)

{

if (frequencies[mostFrequentNum] < frequencies[n])

mostFrequentNum = n;

}

Console.WriteLine(mostFrequentNum + ": " + frequencies[mostFrequentNum]);