C# Chapter 7

1. Write a program, which creates an array of **20 elements of type integer** and initializes each of the elements with a value equals to the index of the element multiplied by 5. Print the elements to the console.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_1

{

class Program

{

static void Main(string[] args)

{

int[] arr = new int[20];

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

{

arr[i] = i \* 5;

Console.WriteLine(arr[i]);

}

}

}

}

1. Write a program, which **reads two arrays** from the console and **checks whether they are equal** (two arrays are equal when they are of equal length and all of their elements, which have the same index, are equal).

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_2

{

class Program

{

static void Main(string[] args)

{

bool arraysEqual = true;

Console.Write("Enter lenght of first array: ");

int length = Int32.Parse(Console.ReadLine());

int[] arrA = new int[length];

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

{

Console.Write("Enter element {0}: ", i);

arrA[i] = Int32.Parse(Console.ReadLine());

}

Console.Write("\nEnter lenght of second array: ");

if (length != Int32.Parse(Console.ReadLine())) Console.WriteLine("\nArrays have different lengths.");

else

{

int[] arrB = new int[length];

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

{

Console.Write("Enter element {0}: ", i);

arrB[i] = Int32.Parse(Console.ReadLine());

}

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

{

if (arrA[i] != arrB[i])

{

Console.WriteLine("\nArrays are different.");

arraysEqual = false;

break;

}

}

if (arraysEqual) Console.WriteLine("\nArrays are the same.");

}

}

}

1. Write a program, which **compares two arrays of type char lexicographically** (character by character) and checks, which one is first in the lexicographical order.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_3

{

class Program

{

static void Main(string[] args)

{

bool arrayEqual = true;

char[] arrA = new char[5] { 'a', 'b', 'c', 'd', 'e' };

char[] arrB = new char[5] { 'a', 'b', 'c', 'd', 'e' };

if (arrA.Length > arrB.Length) Console.WriteLine("Second array is lexicographicaly first.");

else if (arrA.Length < arrB.Length) Console.WriteLine("First array is lexicographicaly first.");

else

{

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

{

if (arrA[i] < arrB[i])

{

Console.WriteLine("First array is lexicographicaly first.");

arrayEqual = false;

break;

}

if (arrA[i] > arrB[i])

{

Console.WriteLine("Second array is lexicographicaly first.");

arrayEqual = false;

break;

}

}

if (arrayEqual) Console.WriteLine("Arrays are lexicographicaly equal.");

}

}

}

}

1. Write a program, which finds the **maximal sequence** **of consecutive equal elements** in an array. E.g.: {1, 1, 2, 3, **2, 2, 2**, 1} à {2, 2, 2}.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_4

{

class Program

{

static void Main(string[] args)

{

int count = 1, tempCount = 1, number = 0;

Console.Write ("Enter array length: ");

int length = Int32.Parse (Console.ReadLine ());

int[] arr = new int[length];

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

{

Console.Write ("Enter {0} element: ", i);

arr [i] = Int32.Parse (Console.ReadLine ());

}

for (int i = 0; i < arr.Length - 1; i++)

{

if (arr[i] == arr[i + 1]) tempCount++;

else tempCount = 1;

if (tempCount > count)

{

count = tempCount;

number = arr [i];

}

}

for (int i = 0; i < count; i++) Console.Write ("{0}, ", number);

}

}

}

1. Write a program, which finds the **maximal sequence** of consecutively placed **increasing** integers. Example: {3, **2, 3, 4**, 2, 2, 4} à {2, 3, 4}.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_5

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

int sames = 1, bestSames = 1, bestStart = 0, lastElement = 0;

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

for (int i = 0; i < arr.Length - 1; i++)

{

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

{

sames++;

if (sames > bestSames)

{

bestSames = sames;

lastElement = i + 1;

bestStart = lastElement - bestSames + 1;

}

}

else sames = 1;

}

for (int i = bestStart; i < bestSames + bestStart; i++) Console.Write("{0}, ", arr[i]);

}

}

}

1. Write a program, which finds the **maximal sequence of increasing elements** in an array **arr[n]**. It is not necessary the elements to be consecutively placed. E.g.: {9, 6, **2**, 7, **4**, 7, **6**, 5, **8**, 4} à {2, 4, 6, 8}.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_6

{

class Program

{

static void Main(string[] args)

{

int counter = 0, tempIndex, tempCounter;

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

int[] result = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

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

{

int[] tempResult = new int[length];

tempIndex = tempCounter = 1;

tempResult[0] = arr[i];

for (int j = i + 1; j < length; j++)

{

if (arr[j] > tempResult[tempIndex - 1])

{

tempResult[tempIndex] = arr[j];

tempIndex++;

tempCounter++;

}

else if (tempIndex > 1 && arr[j] > tempResult[tempIndex - 2] && arr[j] < tempResult[tempIndex - 1]) tempResult[tempIndex - 1] = arr[j];

}

if (counter < tempCounter)

{

counter = tempCounter;

result = tempResult;

}

}

for (int i = 0; i < counter; i++) Console.Write("{0},", result[i]);

}

}

}

1. Write a program, which reads from the console two integer numbers **N** and **K** (K<N) and array of N integers. Find those **K consecutive elements**in the array, which have**maximal sum**.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_7

{

class Program

{

static void Main(string[] args)

{

int sum = 0;

Console.Write("Enter N: ");

int n = Int32.Parse(Console.ReadLine());

Console.Write("Enter K (K < N): ");

int k = Int32.Parse(Console.ReadLine());

int[] arr = new int[n];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

Array.Sort(arr, (a, b) => b.CompareTo(a));

for (int i = 0; i < k; i ++) sum += arr[i];

Console.WriteLine("\nBiggest sum is {0}", sum);

}

}

}

1. **Sorting an array** means to arrange its elements in an increasing (or decreasing) order. Write a program, which sorts an array using the algorithm "**selection sort**".

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_8

{

class Program

{

static void Main(string[] args)

{

int i, j, iMin, temp;

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

for (i = 0; i < length; i++)

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

for (j = 0; j < length - 1; j++)

{

iMin = j;

for (i = j + 1; i < length; i++) if (arr[i] < arr[iMin]) iMin = i;

if (iMin != j)

{

temp = arr[j];

arr[j] = arr[iMin];

arr[iMin] = temp;

}

}

for (i = 0; i < length; i++) Console.Write("{0} ", arr[i]);

}

}

}

1. Write a program, which finds a **subsequence of numbers with maximal sum**. E.g.: {2, 3, -6, -1, **2, -1, 6, 4**, -8, 8} à **11**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_9

{

class Program

{

static void Main(string[] args)

{

int sum = 0, tempSum;

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

for (int i = 0; i < length - 1; i++)

{

tempSum = arr[i];

for (int j = i + 1; j < length; j++)

{

tempSum += arr[j];

if (tempSum > sum) sum = tempSum;

}

}

Console.WriteLine("Result is {0}. ", sum);

}

}

}

1. Write a program, which finds the **most frequently occurring** element in an array. Example: {**4**, 1, 1, **4**, 2, 3, **4**, **4**, 1, 2, **4**, 9, 3} à 4 (5 times).

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_10

{

class Program

{

static void Main(string[] args)

{

int counter = 0, tempCounter = 1, foundNumber = 0;

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

Array.Sort(arr);

for (int i = 0; i < length - 1; i++)

{

if (arr[i] == arr[i + 1]) tempCounter++;

else tempCounter = 1;

if (tempCounter > counter)

{

counter = tempCounter;

foundNumber = arr[i];

}

}

Console.WriteLine("{0} was found {1} times.", foundNumber, counter);

}

}

}

1. Write a program to find a sequence of neighbor numbers in an array, which has a **sum of certain number S**. Example: {4, 3, 1, **4, 2, 5**, 8}, S=**11** à {4, 2, 5}.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_11

{

class Program

{

static void Main(string[] args)

{

int sum = 0, start = 0, end = 0;

bool sumFound = false;

Console.Write("Enter S: ");

int s = Int32.Parse(Console.ReadLine());

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

for (int i = 0; i < length - 1; i++)

{

sum = arr[i];

for (int j = i + 1; j < length; j++)

{

sum += arr[j];

if (sum == s)

{

start = i;

end = j;

sumFound = true;

break;

}

}

if (sumFound) break;

}

if (sumFound) for (int i = start; i <= end; i++) Console.Write("{0},", arr[i]);

else Console.WriteLine("No sum found.");

}

}

}

1. Write a program, which creates **square matrices** like those in the **figures below** and prints them formatted to the console. The size of the matrices will be read from the console. E.g. matrices with size of 4 x 4:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_12a

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter height: ");

int y = Int32.Parse(Console.ReadLine());

Console.Write("Enter width: ");

int x = Int32.Parse(Console.ReadLine());

int a = 0;

for (int i = 1; i <= y; i++)

{

Console.Write("{0} ", i);

a += i;

for (int j = 1; j < x; j++)

{

a += y;

Console.Write("{0} ", a);

}

a = 0;

Console.WriteLine();

}

}

}

}

1. Write a program, which creates a rectangular array with size of **n** by **m**elements. The dimensions and the elements should be read from the console. Find a **platform with size of (3, 3) with a maximal sum**.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_13

{

class Program

{

static void Main(string[] args)

{

int row = 0, col = 0, sum = -1000;

Console.Write("Enter N: ");

int n = Int32.Parse(Console.ReadLine());

Console.Write("Enter M: ");

int m = Int32.Parse(Console.ReadLine());

int[,] arr = new int[n, m];

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

for (int j = 0; j < m; j++)

{

Console.Write("Arr [{0}][{1}] = ", i, j);

arr[i, j] = Int32.Parse(Console.ReadLine());

}

for (int tempRow = 0; tempRow < arr.Length - 2; tempRow++)

for (int tempCol = 0; tempCol < arr.GetLength(0) - 2; tempCol++)

{

int tempSum = arr[row, col] + arr[row, col + 1] + arr[row, col + 2] +

arr[row + 1, col] + arr[row + 1, col + 1] + arr[row + 1, col + 2] +

arr[row + 2, col] + arr[row + 2, col + 1] + arr[row + 2, col + 2];

if (tempSum > sum)

{

row = tempRow;

col = tempCol;

sum = tempSum;

}

}

Console.WriteLine("Result");

Console.WriteLine("{0} {1} {2}", arr[row, col], arr[row, col + 1], arr[row, col + 2]);

Console.WriteLine("{0} {1} {2}", arr[row + 1, col], arr[row + 1, col + 1], arr[row + 1, col + 2]);

Console.WriteLine("{0} {1} {2}", arr[row + 2, col], arr[row + 2, col + 2], arr[row + 2, col + 2]);

Console.WriteLine("The maximum sum is {0}.", sum);

}

}

}

1. Write a program, which finds the **longest sequence of equal** **string** elements in a matrix. A sequence in a matrix we define as a set of neighbor elements **on the same row, column or diagonal**.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_14

{

class Program

{

static void Main(string[] args)

{

int tempSeq = 1, seq = 1;

string element = "e";

Console.Write("Enter N: ");

int n = Int32.Parse(Console.ReadLine());

Console.Write("Enter M: ");

int m = Int32.Parse(Console.ReadLine());

string[,] arr = new string[n, m];

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

for (int j = 0; j < m; j++)

{

Console.Write("Arr [{0}][{1}] = ", i, j);

arr[i, j] = Console.ReadLine();

}

for (int rows = 0; rows < arr.GetLength(0); rows++)

{

for (int cols = 0; cols < arr.GetLength(1) - 1; cols++)

{

if (arr[rows, cols] == arr[rows, cols + 1]) tempSeq++;

else tempSeq = 1;

if (seq < tempSeq)

{

seq = tempSeq;

element = arr[rows, cols];

}

}

tempSeq = 1;

}

for (int cols = 0; cols < arr.GetLength(1); cols++)

{

for (int rows = 0; rows < arr.GetLength(0) - 1; rows++)

{

if (arr[rows, cols] == arr[rows + 1, cols]) tempSeq++;

else tempSeq = 1;

if (seq < tempSeq)

{

seq = tempSeq;

element = arr[rows, cols];

}

}

tempSeq = 1;

}

for (int i = 0; i < arr.GetLength(0) - 1; i++)

for (int j = 0; j < arr.GetLength(1) - 1; j++)

{

for (int rows = i, cols = j; rows < arr.GetLength(0) - 1 && cols < arr.GetLength(1) - 1; rows++, cols++)

{

if (arr[rows, cols] == arr[rows + 1, cols + 1]) tempSeq++;

else tempSeq = 1;

if (seq < tempSeq)

{

seq = tempSeq;

element = arr[rows, cols];

}

}

tempSeq = 1;

}

for (int i = 0; i < arr.GetLength(0) - 1; i++)

for (int j = 1; j < arr.GetLength(1); j++)

{

for (int rows = i, cols = j; rows < arr.GetLength(0) - 1 && cols > 0; rows++, cols--)

{

if (arr[rows, cols] == arr[rows + 1, cols - 1]) tempSeq++;

else tempSeq = 1;

if (seq < tempSeq)

{

seq = tempSeq;

element = arr[rows, cols];

}

}

tempSeq = 1;

}

for (int i = 0; i < seq; i++) Console.Write("{0}, ", element);

}

}

}

1. Write a program, which creates an array containing **all Latin letters**. The user inputs **a word** from the console and as result the program prints to the console the **indices of the letters from the word**.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_15

{

class Program

{

static void Main(string[] args)

{

char[] alphabet = { 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z' };

Console.Write("Enter a word: ");

char[] word = (Console.ReadLine()).ToCharArray();

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

for (int j = 0; j < alphabet.Length; j++)

if (word[i] == alphabet[j]) Console.Write("{0} ", j);

}

}

}

1. Write a program, which uses a **binary** **search** in a **sorted**array of integer numbers to find a certain element.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_16

{

class Program

{

static void Main(string[] args)

{

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

Console.Write("Enter searched number: ");

int number = Int32.Parse(Console.ReadLine());

int index = Array.BinarySearch(arr, number);

if (index >= 0) Console.Write("Number is on {0} index.", index);

else Console.Write("Number wasn't found.");

}

}

}

1. Write a program, which sorts an array of integer elements using a **"merge** **sort"** algorithm.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_17

{

class Program

{

static public void DoMerge(int[] numbers, int left, int mid, int right)

{

int[] temp = new int[25];

int i, left\_end, num\_elements, tmp\_pos;

left\_end = (mid - 1);

tmp\_pos = left;

num\_elements = (right - left + 1);

while ((left <= left\_end) && (mid <= right))

{

if (numbers[left] <= numbers[mid]) temp[tmp\_pos++] = numbers[left++];

else temp[tmp\_pos++] = numbers[mid++];

}

while (left <= left\_end) temp[tmp\_pos++] = numbers[left++];

while (mid <= right) temp[tmp\_pos++] = numbers[mid++];

for (i = 0; i < num\_elements; i++)

{

numbers[right] = temp[right];

right--;

}

}

static public void MergeSort\_Recursive(int[] numbers, int left, int right)

{

int mid;

if (right > left)

{

mid = (right + left) / 2;

MergeSort\_Recursive(numbers, left, mid);

MergeSort\_Recursive(numbers, (mid + 1), right);

DoMerge(numbers, left, (mid + 1), right);

}

}

static void Main(string[] args)

{

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

MergeSort\_Recursive(arr, 0, arr.Length - 1);

for (int i = 0; i < arr.Length; i++ ) Console.WriteLine(arr[i]);

}

}

}

1. Write a program, which sorts an array of integer elements using a "**quick** **sort"**algorithm.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_18

{

class Program

{

public static void Quicksort(int[] elements, int left, int right)

{

int i = left, j = right;

IComparable pivot = elements[(left + right) / 2];

while (i <= j)

{

while (elements[i].CompareTo(pivot) < 0) i++;

while (elements[j].CompareTo(pivot) > 0) j--;

if (i <= j)

{

int tmp = elements[i];

elements[i] = elements[j];

elements[j] = tmp;

i++;

j--;

}

}

if (left < j) Quicksort(elements, left, j);

if (i < right) Quicksort(elements, i, right);

}

static void Main(string[] args)

{

Console.Write("Enter array length: ");

int length = Int32.Parse(Console.ReadLine());

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = Int32.Parse(Console.ReadLine());

}

Quicksort(arr, 0, arr.Length - 1);

for (int i = 0; i < arr.Length; i++) Console.WriteLine(arr[i]);

}

}

}

1. Write a program, which finds **all prime numbers** in the range [1…10,000,000].

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_19

{

class Program

{

static void PrintList(List<int> nums, bool printZero = true)

{

foreach (int n in nums)

{

if (!printZero && n == 0) continue;

Console.Write("{0}\t", n);

}

Console.WriteLine();

}

static void Main(string[] args)

{

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

// 1 is not a prime number

for (int i = 2; i <= 10000000; ++i) nums.Add(i);

int p = 2;

int indexAt = 0;

bool limitReached = false;

while (!limitReached)

{

for (int i = 0; i < nums.Count; ++i)

{

int v = nums[i];

if (v % p == 0 && v != p) nums[i] = 0;

}

do

{

p = nums[++indexAt];

if (indexAt >= nums.Count - 1)

{

limitReached = true;

break;

}

} while (p == 0);

}

PrintList(nums, false);

}

}

1. \* Write a program, which checks whether there is a **subset**of given array of **N** elements, which has a **sum S**. The numbers **N**, **S** and the array values are read from the console. Same number can be used many times.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Chapter\_7\_Solution\_20

{

class Program

{

static int wantedSum;

static bool solution = false;

static void GenerateSubset(int[] arr, int[] subset, int index, int current, int elementsInSubset)

{

if (index == elementsInSubset)

{

CheckSubsets(subset, elementsInSubset);

return;

}

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

{

subset[index] = arr[i];

GenerateSubset(arr, subset, index + 1, i + 1, elementsInSubset);

}

}

static void CheckSubsets(int[] subset, int elementsInSubset)

{

int sum = 0;

for (int i = 0; i < elementsInSubset; i++) sum += subset[i];

if (sum == wantedSum)

{

for (int i = 0; i < elementsInSubset; i++) Console.Write("{0} ", subset[i]);

Console.WriteLine();

solution = true;

}

}

static void Main()

{

Console.Write("Enter array length: ");

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

int[] arr = new int[length];

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

{

Console.Write("Enter {0} element: ", i);

arr[i] = int.Parse(Console.ReadLine());

}

Console.Write("Enter S: ");

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

int[] subset = new int[length];

for (int i = 1; i <= length; i++) GenerateSubset(arr, subset, 0, 0, i);

if (!solution) Console.WriteLine("No subset with sum {0} found.", wantedSum);

}

}

}