# Hyrje ne baza dhe struktura te dhenave

## Chapter-13

### Gerti Gonxh

2.

using System;

namespace ReverseString

{

class ReverseString

{

static void Main(string[] args)

{

string word = Console.ReadLine();

for (int i = word.Length - 1; i >= 0; i--)

{

Console.Write(word[i]);

}

Console.WriteLine();

}

}

}

3.

using System;

namespace CorrectBrackets

{

class CorrectBrackets

{

static void Main(string[] args)

{

string expression = Console.ReadLine();

int leftBrackets = 0, rightBrackets = 0;

bool wright = true;

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

{

if (expression[i] == ')')

{

rightBrackets++;

}

else if (expression[i] == '(')

{

leftBrackets++;

}

if (rightBrackets > leftBrackets)

{

wright = false;

}

}

if (rightBrackets != leftBrackets)

{

wright = false;

}

Console.WriteLine(wright);

}

}

}

5.

using System;

namespace SubstringCounter

{

class SubstringCounter

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string word = string.Empty;

string substring = Console.ReadLine();

int substringCounter = 0;

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

{

word += text[i];

if (word.Length == substring.Length)

{

if (word.ToLower() == substring.ToLower())

{

substringCounter++;

}

word = string.Empty;

word += text[i];

}

}

Console.WriteLine(“Резултатът е ‘{0}’ срещания.”, substringCounter);

}

}

}

6.

using System;

namespace UpperCase

{

class UpperCase

{

static void Main(string[] args)

{

int firstIndex = 0, secondIndex = 0;

string str = Console.ReadLine();

while (firstIndex < str.LastIndexOf("<upcase>"))

{

firstIndex = str.IndexOf("<upcase>", firstIndex + 1);

secondIndex = str.IndexOf("</upcase>", secondIndex + 1);

string upCase = str.Substring(firstIndex + 8, secondIndex - firstIndex - 8).ToUpper();

string toReplace = str.Substring(firstIndex, secondIndex - firstIndex + 9);

str = str.Replace(toReplace, upCase);

}

Console.WriteLine(str);

}

}

}

7.

using System;

namespace StringOf20Characters

{

class StringOf20Characters

{

static void Main(string[] args)

{

string line;

line = Console.ReadLine();

if (line.Length>20)

{

for (int i = 0; i < 20; i++)

{

Console.Write(line[i]);

}

Console.WriteLine();

}

else

{

string newLine = line.PadRight(20, '\*');

Console.WriteLine(newLine);

}

}

}

}

8.

using System;

namespace CsharpUnicodeLiterals

{

class CsharpUnicodeLiterals

{

static void Main(string[] args)

{

string word = Console.ReadLine();

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

{

Console.Write("\\u{0:x4}", (ushort)word[i]);

}

Console.WriteLine();

}

}

}

9.

using System;

using System.Text;

namespace XORCoding

{

class XORCoding

{

static void Main(string[] args)

{

string chiper = Console.ReadLine();

string text = Console.ReadLine();

StringBuilder sb = new StringBuilder();

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

{

if (i == 0)

{

sb.AppendFormat("\\u{0:x4}", (ushort)text[i] ^ (ushort)chiper[i]);

}

else

sb.AppendFormat("\\u{0:x4}", (ushort)text[i] ^ (ushort)chiper[i % chiper.Length]);

}

sb.ToString();

Console.WriteLine(sb);

}

}

}

10.

using System;

namespace ExtractSentencesContainingSpecifiedWord

{

class SentencesContainingSpecifiedWordExtractor

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string word = Console.ReadLine();

int dotIndex = 0, previousDotIndex = 0;

char[] separators = { ' ', ',', '.' };

bool exist = false;

while (dotIndex >= 0 && previousDotIndex < text.Length)

{

dotIndex = text.IndexOf('.', dotIndex + 1);

string sentence = text.Substring(previousDotIndex, dotIndex - previousDotIndex + 1);

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

{

int exists = 0;

for (int k = 0; k < separators.Length; k++)

{

exist = false;

if ((exists = sentence.IndexOf(" " + word + separators[k])) >= 0)

{

Console.WriteLine(sentence);

exist = true;

break;

}

}

if (exist)

{

break;

}

}

previousDotIndex = dotIndex + 2;

}

}

}

}

11.

using System;

namespace ForbidenWords

{

class ForbidenWords

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string[] forbidenWords = Console.ReadLine().Split(',');

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

{

int existForbidenWord = 0;

if ((existForbidenWord = text.IndexOf(forbidenWords[i])) >= 0)

{

string replaced = new string('\*', forbidenWords[i].Length);

text = text.Replace(forbidenWords[i], replaced);

}

}

Console.WriteLine(text);

}

}

}

12.

using System;

namespace NumberFormating

{

class NumberFormating

{

static void Main(string[] args)

{

double number = double.Parse(Console.ReadLine());

Console.Write(

"{0,15:D}\n" +

"{1,15:E}\n" +

"{1,15:P3}\n" +

"{0,15:X}\n",

(int)number, number);

}

}

}

13.

using System;

namespace ParseURL

{

class URLParser

{

static void Main(string[] args)

{

string url = Console.ReadLine();

int protokolIndex = url.IndexOf(':');

string protocol = url.Substring(0, protokolIndex);

int serverIndex = url.IndexOf('/', protokolIndex + 3);

string server = url.Substring(protokolIndex + 3, serverIndex - protokolIndex - 3);

string resource = url.Substring(serverIndex + 1);

Console.WriteLine("[protocol]= \"{0}\"", protocol);

Console.WriteLine("[server]= \"{0}\"", server);

Console.WriteLine("[resource]= \"{0}\"", resource);

}

}

}

14.

using System;

namespace ReverseWordsInText

{

class StringReverser

{

static void Main(string[] args)

{

string sentence = Console.ReadLine();

string[] splitted = sentence.Split(' ');

for (int i = splitted.Length - 1; i >= 0; i--)

{

Console.Write(splitted[i] + " ");

}

Console.WriteLine();

}

}

}

15.

using System;

namespace WordsDictionary

{

class WordsDictionary

{

static void Main(string[] args)

{

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

string[] dictionary = new string[n];

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

{

dictionary[i] = Console.ReadLine();

}

string word = Console.ReadLine();

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

{

int firstSpace = dictionary[i].IndexOf(' ');

string foundWord = dictionary[i].Substring(0, firstSpace);

if (foundWord == word)

{

Console.WriteLine("{0} means {1}", word, dictionary[i].Substring(word.Length + 3));

}

}

}

}

}

16.

using System;

using System.Text.RegularExpressions;

namespace ReplaceHTMLTags

{

class HTMLTagsReplacer

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string newtext = Regex.Replace(text, @"<a href=""", "[URL=");

newtext = Regex.Replace(newtext, @""">", "] ");

newtext = Regex.Replace(newtext, "</a>", "[/URL]");

Console.WriteLine();

Console.WriteLine(newtext);

}

}

}

17.

using System;

using System.Globalization;

namespace DaysBetweenTwoDays

{

class DaysBetweenTwoDays

{

static void Main(string[] args)

{

string format = "dd.MM.yyyy";

DateTime FirstDate = DateTime.ParseExact(Console.ReadLine(), format, CultureInfo.InvariantCulture.DateTimeFormat);

DateTime SecondDate = DateTime.ParseExact(Console.ReadLine(), format, CultureInfo.InvariantCulture.DateTimeFormat);

Console.WriteLine("{0}", Math.Abs((SecondDate - FirstDate).Days));

}

}

}

18.

using System;

using System.Globalization;

namespace DateAndTimeAfter6Hours

{

class DateAndTimeAfter6Hours

{

static void Main(string[] args)

{

string format = "dd.MM.yyyy HH:mm:ss";

DateTime now = DateTime.ParseExact(Console.ReadLine(), format, CultureInfo.InvariantCulture.DateTimeFormat);

DateTime after = now.AddHours(6.5);

Console.WriteLine("{0:dd.MM.yyyy HH:mm:ss}", after);

}

}

}

19.

using System;

using System.Text.RegularExpressions;

namespace ExtractEmails

{

class EmailsExtractor

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string[] words = text.Split(' ');

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

{

if (TestEmailRegex(words[i]))

{

Console.WriteLine(words[i]);

}

}

}

static bool TestEmailRegex(string emailAddress)

{

string patternStrict = @"^(([\w-]+\.)+[\w-]+|([a-zA-Z]{1}|[\w-]{2,}))@"

+ @"((([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\.([0-1]?

[0-9]{1,2}|25[0-5]|2[0-4][0-9])\."

+ @"([0-1]?[0-9]{1,2}|25[0-5]|2[0-4][0-9])\.([0-1]?

[0-9]{1,2}|25[0-5]|2[0-4][0-9])){1}|"

+ @"([a-zA-Z]+[\w-]+\.)+[a-zA-Z]{2,4})$";

Regex reStrict = new Regex(patternStrict);

bool isStrictMatch = reStrict.IsMatch(emailAddress);

return isStrictMatch;

}

}

}

20.

using System;

using System.Text.RegularExpressions;

using System.Globalization;

namespace ExtractDates

{

class DatesExtractor

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string pattern =

"\\b(?<day>[\\d]{1,2})\\.(?<month>[\\d]{2})\\.(?<year>[\\d]{4})\\.";

Regex dates = new Regex(pattern);

MatchCollection matches = Regex.Matches(text, pattern);

foreach (Match date in matches)

{

DateTime theDate = new DateTime(int.Parse(date.Groups["year"].ToString()),

int.Parse(date.Groups["month"].ToString()),

int.Parse(date.Groups["day"].ToString()));

CultureInfo culture = new CultureInfo("en-CA");

Console.WriteLine(theDate.ToString("d", culture));

}

}

}

}

21.

using System;

namespace Palindroms

{

class Palindroms

{

static char[] delimiters = new char[] { ' ', '.', ',', '!', '?', '\"' };

static void Main(string[] args)

{

string text = Console.ReadLine();

string[] words = text.Split(delimiters,StringSplitOptions.RemoveEmptyEntries);

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

{

if (IsPalindrome(words[i]))

{

Console.WriteLine(words[i]);

}

}

}

static bool IsPalindrome(string word)

{

for (int i = 0; i < word.Length/2; i++)

{

if (word[i].ToLower()==word[word.Length - 1 - i].ToLower())

{

continue;

}

else

{

return false;

}

}

return true;

}

}

}

22.

using System;

namespace CountLetters

{

class LettersCounter

{

static void Main(string[] args)

{

string text = Console.ReadLine();

int[] letters = new int[256];

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

{

for (int j = 0; j < 256; j++)

{

if ((int)text[i] == j)

{

letters[j]++;

}

}

}

for (int i = 0; i < 256; i++)

{

if (letters[i] != 0)

{

Console.WriteLine("{0} - {1}", (char)i, letters[i]);

}

}

}

}

}

23.

using System;

namespace CountWords

{

class WordsCounter

{

static void Main(string[] args)

{

string text = Console.ReadLine();

string[] words = text.Split(' ');

string[] wordCounter = new string[words.Length];

int[] counter = new int[words.Length];

int numberOfWords = 0;

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

{

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

{

if (wordCounter[j] == string.Empty || wordCounter[j] != words[i])

{

wordCounter[numberOfWords] = words[i];

numberOfWords++;

break;

}

}

}

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

{

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

{

if (wordCounter[j] == words[i])

{

counter[j]++;

break;

}

}

}

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

{

if (counter[i] == 0)

{

continue;

}

else

Console.WriteLine("{0} - {1}", wordCounter[i], counter[i]);

}

}

}

}

24.

using System;

using System.Text;

namespace ReplaceConsequentiveLetters

{

class ConsequentiveLettersReplacer

{

static void Main(string[] args)

{

string text = Console.ReadLine();

StringBuilder newtext = new StringBuilder();

char letter = text[0];

newtext.Append(text[0]);

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

{

if (letter == text[i])

{

continue;

}

else

{

newtext.Append(text[i]);

letter = text[i];

}

}

newtext.ToString();

Console.WriteLine(newtext);

}

}

}

25.

using System;

using System.Collections.Generic;

namespace SortedWords

{

class WordsSorter

{

static void Main(string[] args)

{

char[] delimiters = { ',', ' ' };

string text = Console.ReadLine();

string[] words = text.Split(delimiters, StringSplitOptions.RemoveEmptyEntries);

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

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

{

if (!wordsToDisplay.Contains(words[i]))

{

wordsToDisplay.Add(words[i]);

}

}

wordsToDisplay.Sort();

foreach (var word in wordsToDisplay)

{

Console.WriteLine(word);

}

}

}

}

26.

using System;

using System.Text;

namespace RemoveHTMLTags

{

class HTMLTagsRemover

{

static void Main(string[] args)

{

StringBuilder output = new StringBuilder();

string line = null;

string lineToLower = null;

while (lineToLower != "</html>")

{

line=Console.ReadLine();

lineToLower = line.ToLower();

bool inTag = false;

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

{

if (line[i] == '>')

{

inTag = false;

continue;

}

if (inTag)

{

continue;

}

if (line[i] == '<')

{

inTag = true;

continue;

}

output.Append(line[i]);

}

output.Append('\n');

}

Console.WriteLine(output);

}

}