

Detyra te shtepise

Hyrje ne Struktura e te te Dhenave

Studenti/ja: Besiana Sinani

Kampusi Prishtine

Viti I-Grupi II

Ligjeruesi: Laberion Zebica

2.    Write a program that reads a string, **reverse** it and prints it to the console. For example: "**introduction**"  "**noitcudortni**".

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.    Write a program that **checks whether the parentheses are placed correctly** in an arithmetic expression. Example of expression with correctly placed brackets: **((a+b)/5-d)**. Example of an incorrect expression: **)(a+b))**.

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);

}

}

4.    How many backslashes you must specify as an argument to the method **Split(…)** in order to **split the text by a backslash**?

Example: **one\two\three**.

Note: In C# backslash is an escaping character.

5.    Write a program that detects how many times a substring is contained in the text. For example, let’s look for the substring "**in**" in the text:

|  |
| --- |
| We are liv**in**g **in** a yellow submar**in**e. We don't have anyth**in**g else. **In**side the submar**in**e is very tight. So we are dr**in**k**in**g all the day. We will move out of it **in** 5 days. |

The result is 9 occurrences.

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.    A text is given. Write a program that **modifies the casing** of letters to uppercase at all places in the text surrounded by **<upcase>** and **</upcase>** tags. Tags cannot be nested.

Example:

|  |
| --- |
| We are living in a <upcase>yellow submarine</upcase>. We don't have <upcase>anything</upcase> else. |

Result:

|  |
| --- |
| We are living in a YELLOW SUBMARINE. We don't have ANYTHING else. |

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.    Write a program that reads a string from the console (20 characters maximum) and if shorter complements it right with "**\***" to 20 characters.

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.    Write a program that converts a given string into the form of array of Unicode escape sequences in the format used in the C# language. Sample input: "**Test**". Result: "**\u0054\u0065\u0073\u0074**".

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.    Write a program that **encrypts a text** by applying XOR (excluding or) operation between the given source characters and given cipher code. The encryption should be done by applying XOR between the first letter of the text and the first letter of the code, the second letter of the text and the second letter of the code, etc. until the last letter of the code, then goes back to the first letter of the code and the next letter of the text. Print the result as a series of Unicode escape characters **\xxxx**.

Sample source text: "**Test**". Sample cipher code: "**ab**". The result should be the following: "**\u0035\u0007\u0012\u0016**".

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.   Write a program that **extracts from a text all sentences that contain a particular word**. We accept that the sentences are separated from each other by the character "**.**" and the words are separated from one another by a character which is not a letter. Sample text:

|  |
| --- |
| We are living **in** a yellow submarine. We don't have anything else. Inside the submarine is very tight. So we are drinking all the day. We will move out of it **in** 5 days. |

Sample result:

|  |
| --- |
| We are living in a yellow submarine.  We will move out of it in 5 days. |

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.   A string is given, composed of several **"forbidden" words** separated by commas. Also a text is given, containing those words. Write a program that **replaces the forbidden words with asterisks**. Sample text:

|  |
| --- |
| Microsoft announced its next generation C# compiler today. It uses advanced parser and special optimizer for the Microsoft CLR. |

Sample string containing the forbidden words: "**C#,CLR,Microsoft**".

Sample result:

|  |
| --- |
| \*\*\*\*\*\*\*\*\* announced its next generation \*\* compiler today. It uses advanced parser and special optimizer for the \*\*\*\*\*\*\*\*\* \*\*\*. |

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.   Write a program that reads a number from console and prints it in **15-character field, aligned right** in several ways: as a decimal number, hexadecimal number, percentage, currency and exponential (scientific) notation.

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.   Write a program that **parses an URL** in following format:

|  |
| --- |
| [protocol]://[server]/[resource] |

It should **extract** from the URL the protocol, server and resource parts. For example, when [**http://www.cnn.com/video**](http://www.cnn.com/video/) is passed, the result is:

|  |
| --- |
| [protocol]="http"  [server]="www.cnn.com"  [resource]="/video" |

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.   Write a program that **reverses the words in a given sentence** without changing punctuation and spaces. For example: "**C# is not C++ and PHP is not Delphi**"  "**Delphi not is PHP and C++ not is C#**".

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.   A dictionary is given, which consists of several lines of text. Each line consists of a **word and its explanation**, separated by a hyphen:

|  |
| --- |
| .NET – platform for applications from Microsoft CLR – managed execution environment for .NET namespace – hierarchical organization of classes |

Write a program that **parses the dictionary** and then reads words from the console in a loop, **gives an explanation** for it or writes a message on the console that the word is not into the dictionary.

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.   Write a program that **replaces all hyperlinks** in a HTML document consisting of **<a href="…">…</a>** and hyperlinks in "forum" style, which look like **[URL=…]…[/URL]**.

Sample text:

|  |
| --- |
| <p>Please visit <a href="https://softuni.bg">our site</a> to choose a training course. Also visit <a href="www.devbg.org">our forum</a> to discuss the courses.</p> |

Sample result:

|  |
| --- |
| <p>Please visit [URL=https://softuni.bg]our site[/URL] to choose a training course. Also visit [URL=www.devbg.org]our forum[/URL] to discuss the courses.</p> |

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.   Write a program that **reads two dates** entered in the format "**day.month.year**" and calculates the **number of days between them**.

|  |
| --- |
| Enter the first date: 27.02.2006  Enter the second date: 3.03.2006  Distance: 4 days |

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.   Write a program that reads the date and time entered in the format "**day.month.year hour:minutes:seconds**" and prints the date and time after 6 hours and 30 minutes in the same format.

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.   Write a program that **extracts all e-mail addresses** from a text. These are all substrings that are limited on both sides by text end or separator between words and match the shape **<sender>@<host>…<domain>**. Sample text:

|  |
| --- |
| Please contact us by phone (+001 222 222 222) or by email at example@gmail.com or at [test.user@yahoo.co.uk](mailto:test.user@yahoo.co.uk). This is not email: test@test. This also: @gmail.com. Neither this: a@a.b. |

Extracted e-mail addresses from the sample text:

|  |
| --- |
| example@gmail.com  test.user@yahoo.co.uk |

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.   Write a program that **extracts from a text all dates** written in format **DD.MM.YYYY**and prints them on the console in the standard format for Canada. Sample text:

|  |
| --- |
| I was born at 14.06.1980. My sister was born at 3.7.1984. In 5/1999 I graduated my high school. The law says (see section 7.3.12) that we are allowed to do this (section 7.4.2.9). |

Extracted dates from the sample text:

|  |
| --- |
| 14.06.1980  3.7.1984 |

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.   Write a program that extracts from a text all words which are **palindromes**, such as **ABBA**", "**lamal**", "**exe**".

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.   Write a program that reads a string from the console and prints in alphabetical order **all letters from the input string and how many times each one of them occurs** in the string.

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.   Write a program that reads a string from the console and prints in alphabetical order **all words from the input string** and **how many times each one of them occurs** in the string.

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.   Write a program that reads a string from the console and replaces every sequence of identical letters in it with a single letter (the **repeating** letter). Example: "**aaaaabbbbbcdddeeeedssaa**"  "**abcdedsa**".

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.   Write a program that reads a list of words separated by commas from the console and prints them in alphabetical order (after **sorting**).

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.   Write a program that **extracts all the text without any tags and attribute values** from an HTML document.

Sample text:

|  |
| --- |
| <html>      <head><title>News</title></head>      <body><p><a href="https://softuni.bg">Telerik          Academy</a>aims to provide free real-world practical          training for young people who want to turn into          skillful software engineers.</p></body>  </html> |

Sample result:

|  |
| --- |
| News  Telerik Academy aims to provide free real-world practical training for young people who want to turn into skillful software engineers. |

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);

}

}