Lekcja 3 – Listy, tablice, Linq, pliki

# **Zadanie z lekcji**

Napisać program Notes, który będzie pozwalał na zapisywanie kolejnych linii wprowadzonego tekstu do listy. W programie będą dostępne komendy:

1. Wypisanie wszystkich wprowadzonych linii tekstu - #list-all
2. Wyszukanie wprowadzonej frazy i podanie numeru wiersza(y) lub że nie istnieje - komenda #search “<tekst>”
3. Wyczyszczenie wszystkich tekstów - komenda #clear
4. Wyjście z programu: #exit

Jeśli wprowadzony tekst nie będzie poprzedzony znakiem #, wtedy dodajemy go do listy, w przeciwnym wypadku program uruchamia odpowiednią komendę, lub informuje użytkownika że takowa nie istnieje. Wpisanie na wejście ## oznacza zapisanie jednego znaku # do listy.

Tego nie przerobiliśmy (kod źródłowy niżej):

\*Dodać możliwość zapisu wszystkich słów do pliku: #save “<nazwa\_pliku>”

\*\*Dodać do programu możliwość wypisania statystyk dla wszystkich tekstów:

1. Ilość linii: #count-lines
2. Ilość słów: #count-words
3. Ilość znaków: #count-chars

\*\*\*Dodać możliwość odczytu z pliku:

1. Nadpisuje bieżący bufor: #load “<nazwa\_pliku>”
2. Dodaje do bieżącego bufora danych: #load-append “<nazwa\_pliku>”

Zadanie domowe

**1. Notes 2.0**

Dodać 3 nowe dowolne komendy do programu Notes: bez argumentową, z 1 argumentem i z 2 argumentami. Np. usuń z wprowadzonych tekstów samogłoski, zamień teksty na hash-e itd.

**2. Tablice 2D**

Stworzyć program który będzie odczytywał z pliku 2 wymiarową mapę do tablicy 2-wymiarowej a następnie określał odległość w linii prostej między dowolnym najniższym oraz dowolnym najwyższym punktem tej mapy oraz wypisywał różnicę wysokości między tymi dwoma punktami.

Plik mapy ma następujący format:

|  |
| --- |
| x y  x liczb o wartościach 0-9  x liczb o wartościach 0-9  ...  ...  …  y razy  …  x liczb o wartościach 0-9 |

**Przykład:**

|  |
| --- |
| 5 4  3 5 1 2 **9**  1 **0** 1 2 6  4 5 7 9 8  1 2 1 0 7 |

Wynik dla powyższego przykładu to odległość: ~3.16 (sqrt(3\*3+1\*1)), różnica wysokości: 9

## **Dodatek (program z lekcji)**

using System;

using System.Collections.Generic;

using System.Linq;

using System.IO;

namespace P14ListsAndFiles

{

class Program

{

static List<string> \_buffer = new List<string>();

private static readonly Dictionary<string, Func<string[], string>>

\_functions = new Dictionary<string, Func<string[], string>>

{

{"list-all", ListAll},

{"search", Find },

{"clear", Clear },

{"save", Save }

};

static void Main(string[] args)

{

ShowInfo();

while (true)

{

var line = Console.ReadLine();

if (line == "#exit") break;

if (CheckInputIsAFunction(line))

{

try

{

var tokens = line.Substring(1).Split(' ');

Console.WriteLine(ExecuteCommand(tokens[0], tokens.Skip(1).ToArray()));

}

catch

{

Console.WriteLine("Invalid command input.");

}

}

else

{

\_buffer.Add(line);

}

}

}

private static bool CheckInputIsAFunction(string line)

{

return line.Length > 0 && line[0] == '#' && (line.Length == 1 || line[1] != '#');

}

private static void ShowInfo()

{

Console.WriteLine("Notepad, available commands:");

Console.WriteLine(string.Join(", ", \_functions.Select(x => "#" + x.Key)));

}

private static string ExecuteCommand(string commandName, params string[] args)

{

if (!\_functions.ContainsKey(commandName))

{

return "Command not found.";

}

return \_functions[commandName](args);

}

private static string ListAll(params string[] args)

{

return string.Join("\r\n", \_buffer);

}

private static string Find(params string[] args)

{

var toFind = args[0].ToLower();

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

//wersja for

/\*for (int i = 0; i < \_buffer.Count; i++)

{

if (\_buffer[i].ToLower().Contains(toFind))

{

listOut.Add($"Lina {i} : {\_buffer[i]}");

}

}\*/

var output = \_buffer

.Select((x, i) => new { item = x, index = i })

.Where(x => x.item.ToLower().Contains(toFind))

.Select(x => $"Line {x.index}:{x.item}").ToArray();

if (output.Length == 0)

{

return "Not found.";

}

return string.Join("\r\n", output);

}

private static string Clear(params string[] args)

{

\_buffer.Clear();

return "Buffer cleared";

}

private static string Save(params string[] args)

{

File.WriteAllLines(args[0], \_buffer);

return $"File {args[0]} saved.";

}

}

}