Мультипарадигменне Програмування

Лабораторна Робота №1

Тема: Імперативне програмування

Виконав: Черпак Андрій, ІП-01

Звіт

Для виконання лабораторної роботи я обрав мову програмування C#, тому що вона передбачає можливість використання конструкції goto, може використовуватися без оголошення функцій (в останніх версіях навіть Main() не обов'язковий), але, при тому, надає вбудовані інструменти для комфортної роботи з рядками тексту.

В даній роботі я дотримався усіх як функціональних (специфічні для кожної задачі), так і нефункціональних вимог (без функцій, без циклів, з використанням оператора goto).

Завдання 1

Для текстового файлу ми хочемо відобразити N (наприклад, 25) найчастіших слів і відповідну частоту їх повторення, упорядковано за зменшенням. Слід обов'язково нормалізувати використання великих літер і ігнорувати стоп-слова, як «the», «for» тощо.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

InpFilename:
    Console.Write("Введіть назву файлу: ");
    string path = Console.ReadLine();
    if (!string.IsNullOrEmpty(path) && File.Exists(path) && path.Length>4 &&
path.Substring(path.Length-4)==".txt") goto FileIsSelected;
    Console.WriteLine("Некоректний ввід!");
    goto InpFilename;

FileIsSelected:
    string text = " "+new StreamReader(path, Encoding.Default).ReadToEnd();

List<string> useless = new() { "\r\n", "[", "]", ",", ".", " - ", ":", ";", "?", "!", "--", "
the ", " a ", " an ", " for ", " on ", " in ", " at ", " to ", " and ", " or ",  " as " };
text = text.ToLower();
int ctr = 0;

RemoveUseless:
    text = text.Replace(symbolsToIgnore[ctr], " ");
    ctr++;
    if (ctr<symbolsToIgnore.Count) goto RemoveUseless;

var words = text.Split(" ", StringSplitOptions.RemoveEmptyEntries);
Dictionary<string, int> dictionary = new();
ctr = 0;

ForEachWord:
    if (dictionary.ContainsKey(words[ctr])) dictionary[words[ctr]] += 1;
    else dictionary.Add(words[ctr], 1);
    ctr++;
```

```
    if (ctr<words.Length) goto ForEachWord;

List<KeyValuePair<string, int>> wordCounts = new(dictionary);
wordCounts.Sort((p1, p2)=>p2.Value.CompareTo(p1.Value));
ctr = 0;
InpRequiredWordsNumber:
    Console.Write("Введіть бажану кількість слів: ");
    if (!Int32.TryParse(Console.ReadLine(), out var requiredWordsNumber) ||
requiredWordsNumber < 1)
    {
        Console.WriteLine("Будь ласка, введіть натуральне число!");
        goto InpRequiredWordsNumber;
    }

OutputWords:
    Console.WriteLine(wordCounts[ctr].Key+" - "+wordCounts[ctr].Value);
    ctr++;
    if (ctr<wordCounts.Count && ctr<requiredWordsNumber) goto OutputWords;
```

Алгоритм виконання задачі опишемо псевдокодом:

Початок

Поки не отримано валідну назву **виконати**:

Виведення "Введіть назву файлу: "

Введення назви в path

Якщо path валідна **то**:

Перервати цикл

Все якщо

Виведення "Некоректний ввід!"

Все поки

Зчитування вмісту файлу з path **у** text

Ініціалізація списку ігнорованих слів symbolsToIgnore

Нормалізація тексту

Для кожного слова **з** symbolsToIgnore **виконати:**

Замінити слово **в** text **на “ ”**

Все виконати

words = text **розділений за “ “**

ініціалізація словника dictionary

для кожного слова **у** words **виконати:**

якщо такий ключ існує у dictionary **то:**

dictionary[слово] = dictionary[слово] + 1

інакше:

додати до dictionary (слово, 1)

все якщо

все виконати

ініціалізувати список wordCounts пар ключ-значення з dictionary

сортувати wordCounts **за значеннями**

поки не введено натуральне число **виконати:**

виведення "Введіть бажану кількість слів: "

якщо введене число не натуральне **то:**

виведення "Будь ласка, введіть натуральне число!"

все якщо

все поки

для кожної пари з wordCounts, але не більше, ніж введене число, **виконати:**

виведення «Ключ - значення**»**

все виконати

кінець

Завдання 2

Для текстового файлу виведіть усі слова в алфавітному порядку разом із номерами сторінок, на яких ці слова знаходяться. Ігноруйте всі слова, які зустрічаються більше 100 разів. Припустимо, що сторінка являє собою послідовність із 45 рядків.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;

InpFilename:
    Console.Write("Введіть назву файлу: ");
    var path = Console.ReadLine();
    if (!string.IsNullOrEmpty(path) && File.Exists(path) && path.Length>4 &&
path.Substring(path.Length-4)==".txt") goto FileIsSelected;
    Console.WriteLine("Некоректний ввід!");
    goto InpFilename;

FileIsSelected:
int lineCtr = 0;
StreamReader sr = new StreamReader(path, Encoding.Default);
```

```
Dictionary<string, List<int>> dictionary = new();
List<string> symbolsToIgnore = new() { "[", "]", ",", ".", " - ", ":", ";", "?", "!", "--", "
the ", " a ", " an ", " for ", " on ", " in ", " at ", " to ", " and ", " or ",  " as " };

NextLine:
    if (sr.EndOfStream) goto EndOfFile;
    lineCtr++;
    string line = sr.ReadLine();
    if (string.IsNullOrEmpty(line)) goto NextLine;
    line = " "+line.ToLower();
    int ctr = 0;

    RemoveUseless:
        line = line.Replace(symbolsToIgnore[ctr], " ");
        ctr++;
        if (ctr<symbolsToIgnore.Count) goto RemoveUseless;

    var words = line.Split(" ", StringSplitOptions.RemoveEmptyEntries);
    ctr = 0;

    ForEachWord:
        if (!dictionary.ContainsKey(words[ctr])) dictionary.Add(words[ctr], new
List<int>(){(int)Math.Ceiling(lineCtr/45.0)});
        else if (!dictionary[words[ctr]].Contains((int)Math.Ceiling(lineCtr/45.0)) &&
                 dictionary.Count<100)
dictionary[words[ctr]].Add((int)Math.Ceiling(lineCtr/45.0));
        ctr++;
        if (ctr<words.Length) goto ForEachWord;

    goto NextLine;

EndOfFile:
    if (dictionary.Count == 0)
    {
        Console.WriteLine("Файл порожній!");
        Environment.Exit(0);
    }
    List<KeyValuePair<string, List<int>>> vocabularyIndexingList = new(dictionary);
    vocabularyIndexingList.Sort((p1, p2)=>String.Compare(p1.Key, p2.Key,
StringComparison.Ordinal));
    ctr = 0;

int pageCtr;

OutputWords:
    Console.Write(vocabularyIndexingList[ctr].Key+" - ");
    pageCtr = 0;

    PrintPage:
        Console.Write(vocabularyIndexingList[ctr].Value[pageCtr]);
        pageCtr++;
        if (pageCtr < vocabularyIndexingList[ctr].Value.Count)
        {
            Console.Write(", ");
            goto PrintPage;
        }

    Console.WriteLine();
    ctr++;
    if (ctr<vocabularyIndexingList.Count) goto OutputWords;
```

Алгоритм виконання задачі опишемо псевдокодом:

Початок

Поки не отримано валідну назву **виконати**:

Виведення "Введіть назву файлу: "

Введення назви в path

Якщо path валідна **то**:

Перервати цикл

Все якщо

Виведення "Некоректний ввід!"

Все поки

ініціалізація словника dictionary

Ініціалізація списку ігнорованих слів symbolsToIgnore

Для кожного рядка файлу path **виконати:**

Зчитування рядка **з** path **у** line

Якщо рядок порожній **то:**

Перейти на наступну ітерацію

Все якщо

Нормалізація тексту

Для кожного слова **з** symbolsToIgnore **виконати:**

Замінити слово **в** line **на “ ”**

Все виконати

words = line **розділений за “ “**

для кожного слова **у** words **виконати:**

якщо такий ключ існує у dictionary **то:**

додати до dictionary[слово] номер сторінки

інакше:

додати до dictionary (слово, [неомер сторінки])

все якщо

все виконати

все виконати

якщо файл був порожнім **то:**

виведення "Файл порожній!"

кінець

все якщо

ініціалізувати список vocabularyIndexingList пар ключ-значення з dictionary

сортувати vocabularyIndexingList **за ключами**

для кожної пари з vocabularyIndexingList **виконати:**

виведення «Ключ – значення через кому**»**

все виконати

кінець

Під час виконання лабораторної роботи ми використовували вбудовані функції введення та виведення в консоль(Console.ReadLine та Console.WriteLine), читання з файлу (StreamReader.ReadLine), розділення рядка (Split), заміщення (Replace), вирізання (Substring), приведення до нижнього регістру (ToLower), перевірки на нульовий рядок (string.IsNullOrEmpty), існування файлу (File.Exists), округлення в більшу сторону (Math.Ceiling), додавання до колекції (Add), існування в колекції (Contains), компаратори (String.Compare та CompareTo), а також функцію переривання програми (Environment.Exit).

Висновок: під час виконання лабораторної роботи ми ще раз ознайомилися з імперативним програмуванням, уявили, як писався код у 1950х, а також написали дві прості програмки без використання циклів та функцій, лише з допомогою оператора goto. Результати виконання лабораторної можна знайти тут.