Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Мультипарадигменне програмування

ЗВІТ

до лабораторних робіт

Виконав студент ІТ-04 Стрільчук Михайло Васильович
(№ групи, прізвище, ім'я, по батькові)

Прийняв ас. Очеретяний О. К.
(посада, прізвище, ім'я, по батькові)

Київ 2022

1. ЗАВДАННЯ ЛАБОРАТОРНОЇ РОБОТИ

Практична робота складається із трьох завдань, які самі по собі є досить простими. Але, оскільки задача - зрозуміти, як писали код наші славні пращури у 1950-х, ми введемо кілька обмежень:

- Заборонено використовувати функції
- Заборонено використовувати цикли
- Для виконання потрібно взяти мову, що підтримує конструкцію GOTO

Завдання 1:

Обчислювальна задача тут тривіальна: 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. Щоб все було просто, ми не піклуємося про порядок слів з однаковою частотою повторень. Ця обчислювальна задача відома як **term frequency**.

Завдання 2:

Тепер, нам потрібно виконати задачу, що називається словниковим індексуванням. 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.

2. ОПИС ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

Для виконання завдань лабораторної роботи я обрав мову програмування C#, яка підтримує конструкцію GOTO. Серед вбудованих методів були використані лише: `File.ReadAllLines()` та `Console.WriteLine()`, окрім них були використані масиви, умовні конструкції "if" та конструкція "goto".

3. ОПИС ПРОГРАМНОГО КОДУ

Завдання 1:

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

namespace Lab1
{
    class Program
    {
        static void Main(string[] args)
        {
            string txtFile = @"D:\Downloads\input.txt";
```

```
            string[] allLines = File.ReadAllLines(txtFile);
            string[] allWords = new string[1000000];
            int[] amountOfUsedWords = new int[1000000];
            int lineIndex = 0;
            int allWordsIndex = 0;
            int amountOfUsedWordsIndex = 0;

        LoopForLines:
            if (lineIndex != allLines.Length)
            {
                string currLine = allLines[lineIndex];
                currLine += " ";
                int lengthOfLine = currLine.Length;
                string currWord = string.Empty;
                int currSymbolIndexInLine = 0;

            LoopForSymbols:
                if (currSymbolIndexInLine != lengthOfLine)
                {
                    if (currLine[currSymbolIndexInLine] != ' ')
                    {
                        if(currLine[currSymbolIndexInLine] >= 'A' &&
currLine[currSymbolIndexInLine] <= 'Z')
                        {
                            currWord += (char)(currLine[currSymbolIndexInLine] + 32);
                        }
                        else if(currLine[currSymbolIndexInLine] >= 'a' &&
currLine[currSymbolIndexInLine] <= 'z')
                        {
                            currWord += currLine[currSymbolIndexInLine];
                        }
                    }
                    else
                    {
                        int indexOfWord = 0;
                        int indexOfCoincidence = 0;
                        bool flag = false;

                        if (currWord.Length > 3)
                        {

                        LoopForCheckWord:
                            if (!flag && indexOfWord < allWords.Length - 1)
                            {
                                if (currWord == allWords[indexOfWord])
                                {
                                    flag = true;
                                    indexOfCoincidence = indexOfWord;
                                }
                                indexOfWord++;
                                goto LoopForCheckWord;
                            }

                            if (flag)
                            {
                                amountOfUsedWords[indexOfCoincidence]++;
                            }
                            else
                            {
                                allWords[allWordsIndex] = currWord;
                                allWordsIndex++;
                                amountOfUsedWords[amountOfUsedWordsIndex] = 1;
                                amountOfUsedWordsIndex++;
```

```
                        }
                    }
                    currWord = string.Empty;
                }
                currSymbolIndexInLine++;
                goto LoopForSymbols;
            }
            lineIndex++;
            goto LoopForLines;
        }

        // Bubble sort
        int i = allWordsIndex, j = allWordsIndex, temp;
        string tempWord;

    LoopI:
        if (i >= 0)
        {
            j = allWordsIndex - 1;

        LoopJ:
            if (j >= 1)
            {
                if (amountOfUsedWords[j] < amountOfUsedWords[j - 1])
                {
                    temp = amountOfUsedWords[j];
                    amountOfUsedWords[j] = amountOfUsedWords[j - 1];
                    amountOfUsedWords[j - 1] = temp;

                    tempWord = allWords[j];
                    allWords[j] = allWords[j - 1];
                    allWords[j - 1] = tempWord;
                }
                j--;
                goto LoopJ;
            }

            i--;
            goto LoopI;
        }

        // Print words
        i = allWordsIndex - 1;

    LoopPrint:
        if (i >= 0)
        {
            Console.WriteLine(allWords[i] + " - " + amountOfUsedWords[i]);
            i--;
            goto LoopPrint;
        }
    }
}
```

Завдання 2:

```
using System;
using System.IO;

namespace Lab1_Task2
{
```

```
    class Program
    {
        static void Main(string[] args)
        {
            const string txtFile = @"D:\Downloads\input.txt";
            var chars = new[] { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9" };
            string[] allLines = File.ReadAllLines(txtFile);
            string[] allWords = new string[1000000];
            string[] listOfPagesForEveryWord = new string[1000000];
            int[] numberOfPagesInListOfPages = new int[1000000];
            int lineIndex = 0;
            int allWordsIndex = 0;

        LoopForLines:
            if (lineIndex != allLines.Length)
            {
                string currLine = allLines[lineIndex];
                currLine += " ";
                int lengthOfLine = currLine.Length;
                string currWord = string.Empty;
                int currSymbolIndexInLine = 0;
                if (currLine != " ")
                {
                LoopForSymbols:
                    if (currSymbolIndexInLine != lengthOfLine)
                    {
                        if (currLine[currSymbolIndexInLine] != ' ')
                        {
                            if (currLine[currSymbolIndexInLine] >= 'A' &&
currLine[currSymbolIndexInLine] <= 'Z')
                            {
                                currWord += (char)(currLine[currSymbolIndexInLine] + 32);
                            }
                            else if (currLine[currSymbolIndexInLine] >= 'a' &&
currLine[currSymbolIndexInLine] <= 'z')
                            {
                                currWord += currLine[currSymbolIndexInLine];
                            }
                        }
                        else
                        {
                            int indexOfWord = 0;
                            int indexOfCoincidence = 0;
                            bool flag = false;

                            if (currWord.Length > 3)
                            {

                            LoopForCheckWord:
                                if (!flag && indexOfWord < allWords.Length - 1)
                                {
                                    if (currWord == allWords[indexOfWord])
                                    {
                                        flag = true;
                                        indexOfCoincidence = indexOfWord;
                                    }
                                    indexOfWord++;
                                    goto LoopForCheckWord;
                                }

                                if (numberOfPagesInListOfPages[indexOfCoincidence] + 1 <= 100)
                                {
                                    if (flag)
```

```
                                    {
                                        int pageNumber = lineIndex / 45;
                                        string pageNumberToString = "";

                                    ConvertIntToString:
                                        if(pageNumber > 0)
                                        {
                                            pageNumberToString = chars[pageNumber % 10] +
pageNumberToString;
                                            pageNumber /= 10;
                                            goto ConvertIntToString;
                                        }

                                        // check if pageNumber already exists in
listOfPagesForEveryWord
                                        int indexInPageNumberToString = 0;
                                        bool exist = true;

                                    CheckExistance:
                                        if(indexInPageNumberToString <
pageNumberToString.Length)
                                        {
                                            int startSymbolToCheck =
listOfPagesForEveryWord[indexOfCoincidence].Length - pageNumberToString.Length;
                                            if
(listOfPagesForEveryWord[indexOfCoincidence][startSymbolToCheck] !=
pageNumberToString[indexInPageNumberToString])
                                            {
                                                exist = false;
                                                indexInPageNumberToString =
pageNumberToString.Length;
                                            }
                                            indexInPageNumberToString++;
                                            goto CheckExistance;
                                        }
                                        if (!exist)
                                        {
                                            listOfPagesForEveryWord[indexOfCoincidence] += ", "
+ pageNumberToString;
                                        }
                                    }
                                    else
                                    {
                                        allWords[allWordsIndex] = currWord;
                                        listOfPagesForEveryWord[allWordsIndex] += " - " +
(int)(lineIndex / 45);
                                        allWordsIndex++;
                                    }
                                    numberOfPagesInListOfPages[allWordsIndex]++;
                                }
                                else
                                {
                                    // remove word and its list
                                    allWords[allWordsIndex] = null;
                                    listOfPagesForEveryWord[indexOfCoincidence] = null;
                                }
                            }
                            currWord = "";
                        }
                        currSymbolIndexInLine++;
                        goto LoopForSymbols;
                    }
                }
```

```
                lineIndex++;
                goto LoopForLines;
            }

            // Bubble sort
            int i = 0, j = 0, index = 0, indexOfCommonSymbols = 0;
            string tempWord, temp;
            bool firstWordIsLarger = false;
        LoopI:
            if (i < allWordsIndex)
            {
                j = 0;
            LoopJ:
                if (j < allWordsIndex - 1)
                {
                    index = 0;
                    indexOfCommonSymbols = 0;
                    int lengthOfShortestWord = allWords[j].Length <= allWords[j + 1].Length ?
allWords[j].Length : allWords[j + 1].Length;
                LoopForSymbolsInCurrWord:
                    if (index < lengthOfShortestWord)
                    {
                        if (allWords[j][index] > allWords[j + 1][index])
                        {
                            firstWordIsLarger = true;
                            goto EndComparingWords;
                        }
                        else if (allWords[j][index] < allWords[j + 1][index])
                        {
                            firstWordIsLarger = false;
                            goto EndComparingWords;
                        }
                        else
                        {
                            indexOfCommonSymbols++;
                            if(indexOfCommonSymbols == lengthOfShortestWord)
                            {
                                firstWordIsLarger = true;
                                goto EndComparingWords;
                            }
                        }
                        index++;
                        goto LoopForSymbolsInCurrWord;
                    }
                EndComparingWords:
                    if (firstWordIsLarger)
                    {
                        temp = listOfPagesForEveryWord[j];
                        listOfPagesForEveryWord[j] = listOfPagesForEveryWord[j + 1];
                        listOfPagesForEveryWord[j + 1] = temp;

                        tempWord = allWords[j];
                        allWords[j] = allWords[j + 1];
                        allWords[j + 1] = tempWord;
                    }
                    j++;
                    goto LoopJ;
                }

                i++;
                goto LoopI;
            }
```

```
            i = 0;

        LoopPrint:
            if (i < allWordsIndex)
            {
                Console.WriteLine(allWords[i] + listOfPagesForEveryWord[i]);
                i++;
                goto LoopPrint;
            }
        }
    }
}
```

Описи роботи алгоритмів:

Завдання 1:

1. Оголошуємо та ініціалізовуємо змінні.
2. Записуємо весь текст з файлу у масив рядків.
3. За допомогою конструкції goto організовуємо два цикли. Один – зовнішній, який проходиться по масиву рядків та інший – внутрішній, який проходиться по символах поточного рядка.
4. Перевіряємо символи. Якщо це не пробіл, тоді перевіряємо чи це буква, якщо маємо велику літеру, то перетворюємо на малу. Впродовж цієї перевірки, записуємо символи до змінної, поки не зустрінемо пропуск.
5. Якщо ми натрапляємо на пропуск, то перевіряємо чи слово, до якого ми записували символи має довжину більше трьох. Далі перевіряємо, чи ми вже не натрапляли на це слово, якщо ні, то записуємо на нашого масиву, та за його індексом записуємо у інший масив одиницю (помітка, що маємо одне таке слово), якщо так, то знаходимо індекс цього слова, та у смасиві, де зберігається кількість повторень, збільшуємо значення за даним індексом на одиницю.
6. Виконуємо сортування бульбашкою, порівнюючи частоту повторень слів.
7. Виводимо наші масиви слів та їх повторень у зворотному порядку.

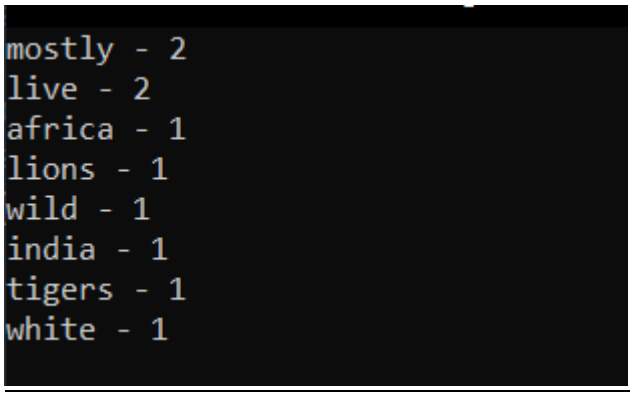
Завдання 2:

1. Оголошуємо та ініціалізовуємо змінні
2. Записуємо весь текст з файлу у масив рядків
3. За допомогою конструкції goto організовуємо два цикли. Один – зовнішній, який проходиться по масиву рядків та інший – внутрішній, який проходиться по символах поточного рядка
4. Перевіряємо символи. Якщо це не пробіл, тоді перевіряємо чи це буква, якщо маємо велику літеру, то перетворюємо на малу. Впродовж цієї перевірки, записуємо символи до змінної, поки не зустрінемо пропуск.
5. Якщо ми натрапляємо на пропуск, то перевіряємо чи слово, до якого ми записували символи має довжину більше трьох. Далі перевіряємо, чи ми вже не натрапляли на це слово, якщо натрапляли 100 і більше разів, то перетворюємо це слово на null у нашому масиві слів. В інакшому випадку дізнаємось номер сторінки, поділивши індекс рядка на 45 та перетворюємо його на текст у конструкції goto.

6. Перевіряємо чи записане наше слово у масиві знайдених слів, якщо так, то дописуємо у інший масив (за індексом цього слова) кому та номер сторінки. Якщо ні, то додаємо до нашого масиву слово та до сусіднього масиву тире та номер сторінки.
7. Виконуємо сортування бульбашкою, порівнюючи слова між собою посимвольно за допомогою конструкцій goto.
8. Виводимо на екран наш масив зі словами та масив із записами сторінок.

4. СКРІНШОТИ РОБОТИ ПРОГРАМНОГО ЗАСТОСУНКУ

Завдання 1:



Завдання 2: