НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО”
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Кафедра інформатики та програмної інженерії

Звіт до лабораторної роботи №1

з курсу

«Мультипарадигменне програмування»

студента 2 курсу
групи ІТ-01
Нікітченко Назара Олеговича

Викладач:
ас. Очеретяний О. К.

Київ – 2022

Завдання

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**.

```
using System;

using System.IO;


namespace Task1 {

   class task1 {

       static void Main(string[] args) {

           string[] stopWords = new string[] {

           "i", "me", "my", "we", "our", "you",
"your", "he", "him", "his",

           "she", "her", "it", "they", "them",
"their", "what", "which", "who",

           "whom", "this", "that", "these", "those",
"am", "is", "are", "was",

           "were", "be", "have", "has", "had", "do",
"does", "did", "a", "an",

           "the", "and", "but", "if", "or",
"because", "as", "until", "while",

           "of", "at", "by", "for", "to", "then",
"here", "there", "when", "where",

           "why", "how", "each", "no", "nor", "not",
"so", "than", "too", "can",

           "will", "now" }; // Стоп слова обробка
яких вказавна в завданні. Список може бути більшим

           string inPath = "input.txt"; // Файл з
вхідними даними, текстом
```

```
            string outPath = "output.txt"; // Файл з вихідними даними, результатом роботи програми
            string[] words = new string[0]; // Створюємо масив рядків для слів
            int[] counts = new int[0]; // Створюємо масив для підрахунку
            int length = 0, i;
            int maxWordsCount = 25; // Кількість виведених слів 25
            string word = ""; // Порожній рядок для зчитаного слова
            StreamReader reader = new StreamReader(inPath); // Читаємо вхідні дані
            Basic_Label: { // Основна мітка
                if (reader.EndOfStream) // Якщо кінець, то завершуємо читання
                    goto End_Reading_Label;
                char symbol = (char)reader.Read(); // Читаємо по символу
                if ('Z' >= symbol && symbol >= 'A') { // По символу обробляємо слово, і зберігаємо його
                    word += ((char)(symbol + 32)).ToString();
                    if (!reader.EndOfStream)
                        goto Basic_Label;
                }
                else if ('z' >= symbol && symbol >= 'a') {
                    word += symbol;
                    if (!reader.EndOfStream)
                        goto Basic_Label;
```

```
                }
                if (word != "" && symbol != '-' &&
symbol != '\'') {
                    i = 0;
                    Check_Stop_Words_Label: { //
Мітка для перевірки чи являється слово стоп словом
                        if (word == stopWords[i]) {
                            word = "";
                            if (reader.EndOfStream)
                                goto
End_Reading_Label;
                            goto Basic_Label;
                        }
                        i++;
                        if (i < stopWords.Length)
                            goto
Check_Stop_Words_Label;
                    }
                    i = 0;
                    Check_Words_Label: { // Мітка для
перевірки чи це нове слово
                        if (i == length)
                            goto New_Word_Label;
                        if (word == words[i]) {
                            counts[i]++;
                            word = "";
                            if (reader.EndOfStream)
                                goto
End_Reading_Label;
```

```
                    goto Basic_Label;
                }
                i++;
                goto Check_Words_Label;
            }
        New_Word_Label: // Мітка для обробки появи нового слова
        if (length == words.Length) {
            string[] New_Word_Labels = new string[(length + 1) * 2];
            int[] newCounts = new int[(length + 1) * 2];
            i = 0;
            For_Copy_Label: { // Мітка для запису даних про нове слово
                if (i == length) {
                    words = New_Word_Labels;
                    counts = newCounts;
                    goto End_Copy_Label;
                }
                New_Word_Labels[i] = words[i];
                newCounts[i] = counts[i];
                i++;
                goto For_Copy_Label;
            }
        }
    End_Copy_Label: // Мітка для завершення запису даних про нове слово
```

```
                    words[length] = word;
                    counts[length] = 1;
                    word = "";
                    length++;
                }
                if(!reader.EndOfStream) // Якщо не кінець файлу, то продовжуємо все з початку основної мітки
                    goto Basic_Label;
            }
            End_Reading_Label: // Мітка для завершення читання
            reader.Close();
            int  current, c;
            i = 1;
            Sort_Label: { // Мітка початку сортування вставленням
                current = counts[i];
                word = words[i];
                c = i - 1;
                While_Sort_Label: { // Мітка основної частини сортування
                    if (c >= 0 && counts[c] < current)
{
                        counts[c + 1] = counts[c];
                        words[c + 1] = words[c];
                        c--;
                        goto While_Sort_Label;
                    }
```

```
                }
                counts[c + 1] = current;
                words[c + 1] = word;
                i++;
                if (i < length)
                    goto Sort_Label;
            }
            StreamWriter writer = new
StreamWriter(outPath); //  Виводимо вихідні дані в
файл
            i = 0;
            Write_Label: { // Мітка для виведення
даних в файл
                writer.WriteLine(words[i] + " - " +
counts[i]);
                i++;
                if (i < maxWordsCount && i < length)
// Обмеження кількості результатів 25
                    goto Write_Label;
            }
            writer.Close(); // Завершення запису
        }
    }
}
```

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.

```
using System;
using System.IO;

namespace Task2 {
   class task2 {
       static void Main(string[] args) {
           string inPath = "input.txt"; // Файл з
вхідними даними, текстом
           string outPath = "output.txt"; // Файл з
вихідними даними, результатом роботи програми
           string[] words = new string[0]; //
Створюємо масив рядків для слів
           int[] counts = new int[0]; // Створюємо
масив для підрахунку
           int[][] pages = new int[0][]; // Створюємо
масив для сторінок
           int currentPage = 1;
           int length = 0, i, strCount = 0;
           int pageLinesCount = 45; // Кількість
рядків на сторінці файлу, для розрізнення сторінок
           string word = ""; // Порожній рядок для
зчитаного слова
           StreamReader reader = new
StreamReader(inPath); // Читаємо вхідні дані
           Read_File_Label: { // Основна мітка
читання і обробки файлу
               if (reader.EndOfStream) // Якщо
кінець, то завершуємо читання
                   goto End_Reading_Label;
               string str = reader.ReadLine(); //
Читаємо по рядку
               if (strCount == pageLinesCount) {
                   currentPage++; // Зміна кількості
сторінок
                   strCount = 0;
               }
               strCount++;
```

```
                int j = 0;
                Basic_Label: { // Мітка обробки файлу
                    if (j == str.Length)
                        goto End_Basic_Label;
                    char symbol = str[j];
                    if ('Z' >= symbol && symbol >=
'A') { // По символу обробляємо слово, і зберігаємо
його
                        word += ((char)(symbol + 32));
                        if (j + 1 < str.Length)
                            goto End_Basic_Label;
                    }
                    else if ('z' >= symbol && symbol
>= 'a') {
                        word += symbol;
                        if (j + 1 < str.Length)
                            goto End_Basic_Label;
                    }
                    if (word != "" && symbol != '-'
&& symbol != '\'') {
                        i = 0;
                        Check_Words_Label: { // Мітка
для перевірки чи це нове слово
                            if (i == length)
                                goto New_Word_Label;
                            if (word == words[i]) {
// Якщо це слово не нове
                                word = "";
                                if (counts[i] > 100)
{ // Якщо це слово зустрічається менше 100 разів, то
воно ігнорується
                                    goto
End_Basic_Label;
                                }
                                counts[i]++;
                                if (counts[i] <=
pages[i].Length) {

pages[i][counts[i] - 1] = currentPage;
                                }
                                else {
                                    int[] pagesTmp =
new int[counts[i] * 2];
                                    int p = 0;
```

```
                                        Copy_Pages_Label:
{ // Мітка копіювання сторінок
                                            pagesTmp[p] =
pages[i][p];
                                            p++;
                                            if (p <
counts[i] - 1)
                                                goto
Copy_Pages_Label;
                                        }
                                        pages[i] =
pagesTmp;

pages[i][counts[i] - 1] = currentPage;
                                    }
                                    goto End_Basic_Label;
                                }
                                i++;
                                goto Check_Words_Label;
                            }
                            New_Word_Label: // Мітка для
обробки появи нового слова
                            if (length == words.Length) {
                                string[] New_Word_Labels
= new string[(length + 1) * 2];
                                int[] newCounts = new
int[(length + 1) * 2];
                                int[][] newPages = new
int[(length + 1) * 2][];
                                i = 0;
                                For_Copy_Label: { //
Мітка для запису даних про нове слово
                                    if (i == length) {
                                        words =
New_Word_Labels;
                                        counts =
newCounts;
                                        pages = newPages;
                                        goto
End_Copy_Label;
                                    }
                                    New_Word_Labels[i] =
words[i];
```

```
                                newCounts[i] =
counts[i];
                                newPages[i] =
pages[i];
                                i++;
                                goto For_Copy_Label;
                            }
                        }
                        End_Copy_Label: // Мітка для
завершення запису даних про нове слово
                       words[length] = word;
                        counts[length] = 1;
                        pages[length] = new int[] {
currentPage };
                        length++;
                        word = "";
                    }
                    End_Basic_Label: // Мітка для
перезапуску
                    j++;
                    if(j < str.Length)
                        goto Basic_Label;
                }
                if (!reader.EndOfStream) // Якщо не
кінець файлу, то продовжуємо все з початку основної
мітки
                    goto Read_File_Label;
            }
            End_Reading_Label: // Мітка для завершення
читання
            reader.Close();
            int current, c;
            int[] currentPages;
            i = 1;
            Sort_Label: { // Мітка початку сортування
вставленням
                current = counts[i];
                word = words[i];
                currentPages = pages[i];
                c = i - 1;
                While_Sort_Label: { // Мітка основної
частини сортування
                    if (c >= 0) {
                        int symbol = 0;
```

```
                        compWords: { // Мітка для
порівняння слів
                            if (symbol ==
words[c].Length || words[c][symbol] < word[symbol])
                                goto
End_While_Sort_Label;
                            if (symbol + 1 <
word.Length && words[c][symbol] == word[symbol]) {
                                symbol++;
                                goto compWords;
                            }
                        }
                        counts[c + 1] = counts[c];
                        words[c + 1] = words[c];
                        pages[c + 1] = pages[c];
                        c--;
                        goto While_Sort_Label;
                    }
                }
                End_While_Sort_Label: // Мітка
перезапуску сортування
                counts[c + 1] = current;
                words[c + 1] = word;
                pages[c + 1] = currentPages;
                i++;
                if (i < length)
                    goto Sort_Label;
            }
            StreamWriter writer = new
StreamWriter(outPath); //  Виводимо вихідні дані в
файл
            i = 0;
            Write_Label: { // Мітка для виведення
даних в файл
                if (counts[i] <= 100) { // Якщо слово
не було проігноровано
                    writer.Write(words[i] + " - " +
pages[i][0]);
                    int j = 1;
                    outPages: { // Мітка для виведення
номеру сторінки
                        if (j == counts[i])
                            goto endOutPages;
```

```
                        if(pages[i][j] != pages[i][j
- 1])
                            writer.Write(",
"+pages[i][j]);
                        j++;
                        goto outPages;
                    }
                    endOutPages:
                    writer.WriteLine();
                }
                i++;
                if (i < length)
                    goto Write_Label;
            }
            writer.Close(); // Завершення запису
        }
    }
}
```

Тестований текст:

Plato's Atlantis

The lost city of Atlantis is a fictional island mentioned in Plato's works *Timaeus* and *Critias*, where it represents the antagonist naval power that besieges "Ancient Athens". In the story, Athens was able to repel the Atlantean attack, unlike any other nation of the known world, supposedly giving testament to the superiority of Plato's concept of a state as described in his work the Republic. At the end of the story, Atlantis eventually falls out of favor with the gods and famously submerges into the Atlantic Ocean.

The lost city of Atlantis

Despite its minor importance in Plato's work, the Atlantis story has had a considerable impact on literature. The allegorical aspect of Atlantis was taken up in utopian works of several Renaissance writers, such as Bacon's *New Atlantis* and More's *Utopia.*

Plato's vague indications of the time of the events—more than 9,000 years before his day—and the alleged location of Atlantis—"beyond the Pillars of Hercules", an area at the entrance to the Strait of Gibraltar.—has led to much pseudoscientific speculation. As a consequence, Atlantis has become a byword for any and all supposed advanced prehistoric lost civilizations and continues to inspire contemporary fiction, from comic books to films.

But is Atlantis real?

Many teams of scientists have tried to identify the location of the lost city of Atlantis in vain. Many of the proposed sites share some of the characteristics of the Atlantis story (water, catastrophic end, relevant period), but none has been demonstrated to be a true historical Atlantis.

Результат 1:

atlantis - 14

in - 6

platos - 5

lost - 4

story - 4

city - 3

works - 2

athens - 2

any - 2

work - 2

end - 2

location - 2

many - 2

athanasius - 1

kirchers - 1

fictional - 1

island - 1

mentioned - 1

timaeus - 1

critias - 1

represents - 1

antagonist - 1

naval - 1

power - 1

besieges - 1

Результат 2:

a - 1

able - 1

advanced - 1

all - 1

alleged - 1

allegorical - 1

an - 1

ancient - 1

and - 1

antagonist - 1

any - 1

area - 1

as - 1

aspect - 1

at - 1

athanasius - 1

athens - 1

atlantean - 1

atlantic - 1

atlantis - 1

attack - 1

bacons - 1

be - 1

become - 1

been - 1

before - 1

besieges - 1

beyond - 1

books - 1

but - 1

byword - 1

catastrophic - 1

characteristics - 1

city - 1

civilizations - 1

comic - 1

concept - 1

consequence - 1

considerable - 1

contemporary - 1

continues - 1

critias - 1

day - 1

demonstrated - 1

described - 1

despite - 1

end - 1

entrance - 1

events - 1

eventually - 1

falls - 1

famously - 1

favor - 1

fiction - 1

fictional - 1

films - 1

for - 1

from - 1

gibraltar - 1

giving - 1

gods - 1

had - 1

has - 1

have - 1

hercules - 1

his - 1

historical - 1

identify - 1

impact - 1

importance - 1

in - 1

indications - 1

inspire - 1

into - 1

is - 1

island - 1

it - 1

its - 1

kirchers - 1

known - 1

led - 1

literature - 1

location - 1

lost - 1

many - 1

mentioned - 1

minor - 1

more - 1

mores - 1

much - 1

nation - 1

naval - 1

new - 1

none - 1

ocean - 1

of - 1

on - 1

other - 1

out - 1

period - 1

pillars - 1

platos - 1

power - 1

prehistoric - 1

proposed - 1

pseudoscientific - 1

real - 1

relevant - 1

renaissance - 1

repel - 1

represents - 1

republic - 1

scientists - 1

several - 1

share - 1

sites - 1

some - 1

speculation - 1

state - 1

story - 1

strait - 1

submerges - 1

such - 1

superiority - 1

supposed - 1

supposedly - 1

taken - 1

teams - 1

testament - 1

than - 1

that - 1

the - 1

timaeus - 1

time - 1

to - 1

tried - 1

true - 1

unlike - 1

up - 1

utopia - 1

utopian - 1

vague - 1

vain - 1

was - 1

water - 1

where - 1

with - 1

work - 1

works - 1

world - 1

writers - 1

years - 1

Висновок: на даній лабораторній роботі було розглянуто методи програмування без функцій і процедур з метою розуміння складності програмування для програмістів минулого. Застосовувалися мітки і з допомогою GOTO здійснювався перехід до них. Як мову програмування було обрано C# за її простоту і зрозумілість.