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 лабораторної роботи №1 «Імперативне програмування»

Виконав студент ІТ-04 Зубрей Андрій Євгенович _____
(№ групи, прізвище, ім’я, по батькові)

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

Київ 2022

1. Завдання лабораторної роботи

Умови: 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. Наприклад, якщо взяти книгу Pride and Prejudice, перші кілька записів індексу будуть:

2. Опис використаних технологій та алгоритмів

Для написання алгоритмів використовувалсь мова C# разом із її вбудованою конструкцією GOTO.

Завдання 1:
Передумова: надано файл і правильно задано шлях до нього
З файлу зчитується його зміст. Далі ми приведемо кожну літеру до нижнього регістру. Далі створимо масив, що буде містити "кортежі" слова та його частоти Потім ми наповнюємо наш масив словами, та при повторній зустрічі збільшуємо другий член кортежу. Після цього сортуємо їх. У кінці друкуємо рельтат: виводимо перші 25 частот, та відповідні їм слова.

Завдання 2:

Спершу вигружаєм текст в пам’ять. Далі ми приведемо кожну літеру до нижнього регістру. Далі створюємо двовимірний масив, де кортеж буде складатися з чотирьох елементів: слово, частота, сторінки, остання додана сторінка. Далі ми послідовно наповнюємо масив значеннями та обробляєм їх. Далі сортуємо слова та друкуємо результат слів кількістю менше 100.

3. Опис програмного коду

Завдання 1:

```
using System;
using System.ComponentModel.Design;
using System.Globalization;
using System.IO;
namespace MultiParadigm_Lab1_1
{
   class Task_1
   {
      public static void Main(string[] args)
      {
         int MaxFrequentCount = 25;
         string Path = "Path";
         String Content = File.ReadAllText(Path);
         Content += "$";
         String[] IgWord = { "a", "in", "an", "for", "the" };
         String UCase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
         String LCase = ",.!?-;“”abcdefghijklmnopqrstuvwxyz";
         String TempContent = "";
         int i = 0;
         StrCheck:
         if (Content[i] == ' ' || Content[i] == '\n')
         {
            i++;
            TempContent += ' ';
            goto StrCheck;
         }
         if (Content[i] == '\r')
         {
            i++;
            goto StrCheck;
         }
         if (Content[i] == '$')
         {
            TempContent += ' ';
            TempContent += '$';
            goto StrCheckEnd;
         }
         bool Lower = false;
         int LCaseIndex = 0;
         LCaseCheck:
         if (Content[i] == LCase[LCaseIndex]) Lower = true;
         {
            LCaseIndex++;
         }
         if (LCaseIndex != LCase.Length && Lower == false)
         {
            goto LCaseCheck;
         }
         if (Lower)
         {
            TempContent += Content[i];
         }
         else
         {
            int j = 0;
            getLCase:
            if (Content[i] != UCase[j])
            {
               j++;
               goto getLCase;
            }
            TempContent += LCase[j];
         }
         i++;
```

```
goto StrCheck;
StrCheckEnd:
String Word = "";
Object[,] Result = new Object[20000, 2];
int WordCount = 0;
i = 0;
WordCheck:
if (TempContent[i] == ' ')
{
   int IgWordCount = 0;
   bool IgState = false;
   IgWordcheck:
   if (Word.Equals(IgWord[IgWordCount]))
   {
      IgState = true;
   }
   else
   {
      IgWordCount++;
      if (!(IgWordCount >= IgWord.Length)) goto IgWordcheck;
   }
   if (IgState)
   {
      Word = "";
      i++;
      goto WordCheck;
   }
   int j = 0;
   Insert:
   if (Word.Equals((String)Result[j, 0]))
   {
      Result[j, 1] = (int)Result[j, 1] + 1;
      i++;
      Word = "";
      goto WordCheck;
   }
   if (Result[j, 0] == null)
   {
      if (Word.Equals(""))
      {
         i++;
         Word = "";
         goto WordCheck;
      }
      WordCount = j;
      Result[j, 0] = Word;
      Result[j, 1] = 1;
      i++;
      Word = "";
      goto WordCheck;
   }
   j++;
   goto Insert;
}
if (TempContent[i] == '$')
{
   WordCount++;
   goto PrepFrequencies;
}
Word += TempContent[i];
i++;
goto WordCheck;
PrepFrequencies:
int[] Frequency = new int[WordCount];
i = 0;
AddFrequency:
Frequency[i] = (int)Result[i, 1];
i++;
if (i < WordCount)
{
   goto AddFrequency;
```

```
        }
        int Size = Frequency.Length;
        int Temp;
        int Write = 0;
        Out:
        Write++;
        int Sort = 0;
        In:
        if (Frequency[Sort] < Frequency[Sort + 1])
        {
            Temp = Frequency[Sort + 1];
            Frequency[Sort + 1] = Frequency[Sort];
            Frequency[Sort] = Temp;
        }
        Sort++;
        if (Sort < Size - 1)
        {
            goto In;
        }

        if (Write < Size)
        {
            goto Out;
        }
        i = 0;
        int PresFrequency = Frequency[0];
        int Max = MaxFrequentCount;
        int PrevFrequency;
        FrequencyPrint:
        int k = 0;
        ResultPrint:
        if ((int)Result[k, 1] == PresFrequency)
        {
            Console.WriteLine(Result[k, 0] + " - " + PresFrequency);
        }
        k++;
        if (k != WordCount)
        {
            goto ResultPrint;
        }
        DecFrequency:
        if (PresFrequency == 1)
        {
            PresFrequency = 0;
            goto End;
        }
        i++;
        PrevFrequency = PresFrequency;
        if (i >= Frequency.Length)
        {
            goto Exit;
        }
        PresFrequency = Frequency[i];
        if (PresFrequency == PrevFrequency)
        {
            goto DecFrequency;
        }
        End:
        Max--;
        if (Max != 0 && PresFrequency != 0)
        {
            goto FrequencyPrint;
        }
        Exit:
        i = 0;
    }
}
```

}

Завдання 2:

```
using System;
using System.IO;
namespace MultiParadigm_Lab1_2
{
    public class Task_2
    {
        public static void Main(string[] args)
        {
            string Path = "Path";
            String[] Content = File.ReadAllLines(Path);
            //int Lines = 45;
            String UCase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
            String LCase = "abcdefghijklmnopqrstuvwxyz ";
            int LineCount = 1;
            string[] LLine = new string[Content.Length];
            string TempContent;
            ViewLine:
            TempContent = "";
            string Line = Content[LineCount - 1];
            if (Line.Length == 0)
            {
                goto NextLine;
            }
            int i = 0;
            StrCheck:
            int LCaseIndex = 0;
            bool Lower = false;
            LCaseCheck:
            if (Content[LineCount - 1][i] == LCase[LCaseIndex])
            {
                Lower = true;
            }
            LCaseIndex++;
            if (LCaseIndex != LCase.Length && Lower == false)
            {
                goto LCaseCheck;
            }
            if (Lower)
            {
                TempContent += Content[LineCount - 1][i];
            }
            else
            {
                int j = 0;
                getLCase:
                if (Content[LineCount - 1][i] != UCase[j])
                {
                    j++;
                    if (j != UCase.Length)
                    {
                        goto getLCase;
                    }
                }
                if (j != LCase.Length) TempContent += LCase[j];
                else if (Content[LineCount - 1][i] != '.'
                && Content[LineCount - 1][i] != ','
                && Content[LineCount - 1][i] != '?'
                && Content[LineCount - 1][i] != '!'
                && Content[LineCount - 1][i] != "“"
                && Content[LineCount - 1][i] != "”"
                && Content[LineCount - 1][i] != ';')
                {
                    TempContent += Content[LineCount - 1][i];
                }
            }
            i++;
            if (i != Line.Length)
            {
                goto StrCheck;
            }
            NextLine:
```

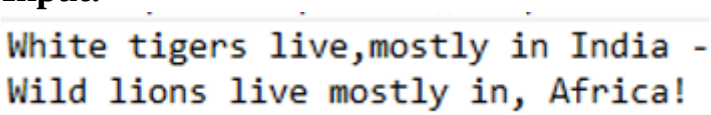
```
if (TempContent != null) LLine[LineCount - 1] = TempContent;
LineCount++;
if (LineCount != Content.Length)
{
   goto ViewLine;
}
object[,] Result = new object[20000, 4];
int Page = 1;
int LineNum = 1;
ViewPage:
//int PrevPage = 0;
Line = LLine[LineNum - 1];
if (LLine.Length == 0)
{
   goto GoToNextLine;
}
i = 0;
String Word = "";
ViewAll:
if (Line[i] == ' ')
{
   int ResultCount = 0;
   ViewSameWord:
   if (Word.Equals((String)Result[ResultCount, 0]))
   {
      Result[ResultCount, 1] = (int)Result[ResultCount, 1] + 1;
      if ((int)Result[ResultCount, 3] != Page)
      {
         Result[ResultCount, 2] = (string)Result[ResultCount, 2] + "," + Page;
         Result[ResultCount, 3] = Page;
      }
      Word = "";
      goto NextSym;
   }
   if (Result[ResultCount, 0] != null)
   {
      ResultCount++;
      goto ViewSameWord;
   }
   Result[ResultCount, 0] = Word;
   Result[ResultCount, 1] = 1;
   Result[ResultCount, 2] = "" + Page;
   Result[ResultCount, 3] = Page;
   Word = "";
   goto NextSym;
}
Word += Line[i];
NextSym:
i++;
if (i != Line.Length)
{
   goto ViewAll;
}
GoToNextLine:
LineNum++;
if (LineNum % 45 == 0)
{
   Page++;
}
if (LineNum < Content.Length)
{
   goto ViewPage;
}
int k = 0;
string[] Words = new string[20000];
GetWords:
if (Result[k, 0] != null)
{
   Words[k] = (string)Result[k, 0];
   k++;
   goto GetWords;
```

```
            }
            string[] TempWords = Words;
            Words = new string[k];
            int WordCount = 0;
            Rewrite:
            Words[WordCount] = TempWords[WordCount];
            WordCount++;
            if (WordCount != k)
            {
               goto Rewrite;
            }
            string Temp;
            int Write = 0;
            Out:
            Write++;
            int Sort = 0;
            In:
            if (Words[Sort].CompareTo(Words[Sort + 1]) > 0)
            {
               Temp = Words[Sort + 1];
               Words[Sort + 1] = Words[Sort];
               Words[Sort] = Temp;
            }
            Sort++;
            if (Sort < Words.Length - 1)
            {
               goto In;
            }
            if (Write < Words.Length)
            {
               goto Out;
            }
            int Amount = Words.Length;
            int f = 0;
            Print:
            if (f == Amount)
            {
               goto Exit;
            }
            int l = 0;
            PrintView:
            if (f != Amount && Words[f].Equals((String)Result[l, 0]))
            {
               Console.WriteLine(Result[l, 0] + " - " + Result[l, 2]);
               f++;
               goto Print;
            }
            l++;
            goto PrintView;
            Exit:
            i = 0;
         }
      }
   }
```

4. Скріншоти роботи програмного застосунку

Завдання 1:

Input:



Output:

```
live - 2
mostly - 2
white - 1
tigers - 1
india - 1
wild - 1
lions - 1
africa - 1
```

Завдання 2:

Input:

```
Download free eBooks of classic literature, books and
novels at Planet eBook. Subscribe to our free eBooks blog
and email newsletter.
Pride and Prejudice
By Jane Austen
 Pride and Prejudice
Chapter 1
I
t is a truth universally acknowledged, that a single man in
possession of a good fortune, must be in want of a wife.
However little known the feelings or views of such a man
may be on his first entering a neighbourhood, this truth is
so well fixed in the minds of the surrounding families, that
he is considered the rightful property of some one or other
of their daughters.
'My dear Mr. Bennet,' said his lady to him one day, 'have
you heard that Netherfield Park is let at last?'
Mr. Bennet replied that he had not.
'But it is,' returned she; 'for Mrs. Long has just been here,
and she told me all about it.'
Mr. Bennet made no answer.
'Do you not want to know who has taken it?' cried his
wife impatiently.
'YOU want to tell me, and I have no objection to hearing it.'
This was invitation enough.
'Why, my dear, you must know, Mrs. Long says that
Netherfield is taken by a young man of large fortune from
the north of England; that he came down on Monday in a
chaise and four to see the place, and was so much delighted
with it, that he agreed with Mr. Morris immediately; that
he is to take possession before Michaelmas, and some of his
Free eBooks at Planet eBook.com
servants are to be in the house by the end of next week.'
'What is his name?'
'Bingley.'
'Is he married or single?'
'Oh! Single, my dear, to be sure! A single man of large
```

Output: