Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни

«Мультипарадигменне програмування»

«Імперативне програмування»

Виконав(ла) *ІП-01 Капщук М .В.* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Очеретяний О. К.* _____
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 1

Завдання

Практична робота складається із двох 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**.

Алгоритм:

Алгоритм досить простий. Для початку зчитати слова. Для цього я зчитую посимвольно файл. Якщо зчитаний символ це літера, то просто додаємо його до поточного слова. Інакше перевіряємо чи символ не є ‘ (апостроф) чи – (дефіс) (вони можуть являтися частиною слова, тому просто ігноруємо їх) та чи не пусте поточне слово. Якщо ж умови не виконуються (тобто ми закінчили зчитувати слово), то спочатку перевіримо чи являється це слово стоп-словом. Якщо ні, то перевіряємо чи ми вже зустрічали слово. Якщо зустрічали – збільшуємо кількість повторів, інакше додаємо слово в список і встановлюємо кількість повторів на 1. Якщо ж місце в масиві слів закінчилося, то перед тим як додавати нове слово, необхідно буде збільшити розмір масиву (деталь реалізації). Після зчитування я використала сортування вставками. І звичайний прохід по словам та вивід.
Опишемо алгоритм у вигляді псевдокоду:

```
//reading
While !EndOfFile
      symbol = Read()
      word = “”
      If symbol is lowercase letter
            word+=symbol
            if !EndOfFile
                  continue
            endIf
```

```
      endIf
      Else if symbol is uppercase letter
            word+=lowercase(symbol)
            if !EndOfFile
                  continue
            endIf
      endElseIf
      If word isn`t empty and symbol isn`t (‘-‘ or ‘`’)
            If word is stop_word
                  Word = ””
            endIf
            Else if word isn`t new
                  Increase word counts
                  Word = ””
            endElseIf
            Else //word is new
                  Remember word
                  Set word counts to 1
                  Word = ””
            endElse
      endIf
endWhile

//sorting
for j = 2 to count of words
      key = counts[j]
      currentWord = words[j]
      i = j - 1
      while i > 0 and counts[i] > key
            words [i + 1] = words [i]
            counts [i + 1] = counts [i]
            i =- 1
      endWhile
      words [i + 1] = currentWord
      counts [i + 1] = key
endFor

//output
For i = 1 to count of words
      Output words[i], ‘ - ‘, counts[i]
endFor
```

Виконання:

```
using System;
using System.IO;

namespace Task1
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] stopWords = new string[] { "the","in","a","an","for","to","on",
                "onto","at","with","about","before", "of"};
```

```
            string inPath = "input.txt";
            string outPath = "output.txt";
            string[] words = new string[0];
            int[] counts = new int[0];
            int length = 0, i;
            int maxWordsCount = 25;
            string word = "";

            //reading
            StreamReader reader = new StreamReader(inPath);
            For1:
            {
                if (reader.EndOfStream)
                    goto endReading;

                char symbol = (char)reader.Read();

                if ('Z' >= symbol && symbol >= 'A')
                {
                    word += ((char)(symbol + 32)).ToString();
                    if (!reader.EndOfStream)
                        goto For1;
                }
                else if ('z' >= symbol && symbol >= 'a')
                {
                    word += symbol;
                    if (!reader.EndOfStream)
                        goto For1;
                }

                if (word != "" && symbol!='-' && symbol!='\'')
                {
                    i = 0;
                    checkStopWords: //check if it`s a stop word
                    {
                        if (word == stopWords[i])
                        {
                            word = "";
                            if (reader.EndOfStream)
                                goto endReading;
                            goto For1;
                        }
                        i++;
                        if (i < stopWords.Length)
                            goto checkStopWords;
                    }

                    i = 0;
                    checkWords: //check if it`s a new word
                    {
                        if (i == length)
                            goto newWord;
                        if (word == words[i])
                        {
                            counts[i]++;
                            word = "";
                            if (reader.EndOfStream)
                                goto endReading;
                            goto For1;
                        }
```

```
                    i++;
                    goto checkWords;
                }

                newWord: //it`s new word
                if (length == words.Length)
                {

                    string[] newWords = new string[(length + 1) * 2];
                    int[] newCounts = new int[(length + 1) * 2];

                    i = 0;
                    forCopy:
                    {
                        if (i == length)
                        {
                            words = newWords;
                            counts = newCounts;
                            goto endCopy;
                        }
                        newWords[i] = words[i];
                        newCounts[i] = counts[i];
                        i++;
                        goto forCopy;
                    }

                }

                endCopy:
                words[length] = word;
                counts[length] = 1;
                word = "";
                length++;
            }

            if(!reader.EndOfStream)
                goto For1;
        }

        endReading:
        reader.Close();

        //sorting (insertion sort)
        int  curr, k;
        i = 1;
        sort:
        {
            curr = counts[i];
            word = words[i];
            k = i - 1;
            whileSort:
            {
                if (k >= 0 && counts[k] < curr)
                {
                    counts[k + 1] = counts[k];
                    words[k + 1] = words[k];
                    k--;
                    goto whileSort;
                }
            }
```

```
                counts[k + 1] = curr;
                words[k + 1] = word;
                i++;
                if (i < length)
                    goto sort;
            }

            //output
            StreamWriter writer = new StreamWriter(outPath);
            i = 0;
            write:
            {
                writer.WriteLine(words[i] + " - " + counts[i]);
                i++;
                if (i < maxWordsCount && i < length)
                    goto write;
            }

            writer.Close();
        }
    }
}
```

Результати:

Input:
White tigers live mostly in India
Wild lions, live mostly In Africa

Output:
live - 2
mostly - 2
white - 1
tigers - 1
india - 1
wild - 1
lions - 1
africa – 1

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.

Алгоритм:

Алгоритм схожий до алгоритму в попередньому завданні. Тільки тепер ми зчитуємо файл пострічно, а кожні 45 стрічок змінюємо номер поточної сторінки. Зчитуємо посимвольно зі стрічки слова. Алгоритм зчитування слова аналогічний. Проте тепер ми не перевіряємо чи слово є стоп-словом і окрім кількості запам'ятовуємо сторінку, на якій зустріли слово (навіть якщо ця сторінка вже в нас записана). І при цьому, коли ми зустрічаємо не нове слово, ми спочатку робимо перевірку чи потраплялося воно нам більше ніж 100 разів (кількість повторів більше 100), тоді поточну сторінку не запам'ятовуємо і кількість не збільшуємо. Після зчитування сортуємо все тим ж сортуванням вставками, але з відмінністю, що слова ми посимвольно порівнюємо і сортуємо за алфавітом. При виводі ми робимо звичайний прохід по всім словам, перевіряємо чи потраплялися вони нам не більше 100 разів і лиш тоді виводимо. При виводі сторінок ми слідкуємо чи була вже виведена сторінка, і якщо ні, то виводимо її.
Опишемо алгоритм у вигляді псевдокоду:

```
//reading
While !EndOfFile
    string = ReadLine()
    if stringsCount == 45
          currentPage+=1
          stringsCount=0
    endif
    stringCount+=1
    For symbol in string
          word = “”
          If symbol is lowercase letter
                word+=symbol
                if !EndOfString
                      continue
                endIf
          endIf
          Else if symbol is uppercase letter
                word+=lowercase(symbol)
                if !EndOfFile
                      continue

                endIf
          endElseIf
          If word isn`t empty and symbol isn`t (‘-‘ or ‘`’)
                if word isn`t new
                     word = ””
                    If wordCount isn`t more than 100
                        Increase word counts
                        Remember page
                    endIf
                endIf
```

```
                Else //word is new
                      Remember word
                      Set word counts to 1
                      Remember page
                      word = ””
                endElse
            endIf
      endFor
endWhile

//sorting
for j = 2 to count of words
      key = words[j]
      currentCounts = counts[j]
      currentPages = pages[j]
      i = j - 1
      while i > 0 and words[i] > key
            words [i + 1] = words [i]
            pages [i + 1] = pages [i]
            counts [i + 1] = counts [i]
            i =- 1
      endWhile
      words [i + 1] = key
      counts [i + 1] = currentCounts
      pages [i + 1] = currentPages
endFor

//output
For i = 1 to count of words
      If word count <= 100
            Output words[i], “ – “, pages[i][0]
            For j = 1 to word count
                  If pages[i][j] != pages[i][j-1]
                        Output “, “, pages[i][j]
            endFor
            Output endline
      endif
endFor
```

Виконання:

```
using System;
using System.IO;

namespace Task2
{
    class Program
    {
        static void Main(string[] args)
        {
            string inPath = "input.txt";
            string outPath = "output.txt";
            string[] words = new string[0];
            int[] counts = new int[0];
            int[][] pages = new int[0][];
```

```
            int currentPage = 1;
            int length = 0, i, strCount = 0;
            int pageLinesCount = 45;
            string word = "";
            StreamReader reader = new StreamReader(inPath);
            readFile:
            {
                if (reader.EndOfStream)
                    goto endReading;

                string str = reader.ReadLine();
                if (strCount == pageLinesCount)
                {
                    currentPage++;
                    strCount = 0;
                }
                strCount++;

                int j = 0;
                For1:
                {
                    if (j == str.Length)
                        goto endFor;
                    char symbol = str[j];
                    if ('Z' >= symbol && symbol >= 'A')
                    {
                        word += ((char)(symbol + 32));
                        if (j + 1 < str.Length)
                            goto endFor;
                    }
                    else if ('z' >= symbol && symbol >= 'a')
                    {
                        word += symbol;
                        if (j + 1 < str.Length)
                            goto endFor;
                    }

                    if (word != "" && symbol != '-' && symbol != '\'')
                    {
                        i = 0;
                        checkWords: //check if it's new word
                        {
                            if (i == length)
                                goto newWord;
                            if (word == words[i]) //it isn't new word
                            {
                                word = "";
                                if (counts[i] > 100) //word is ignored
                                {
                                    goto endFor;
                                }
                                counts[i]++;
                                if (counts[i] <= pages[i].Length) //is array enough for new pa
ge

                                {
                                    pages[i][counts[i] - 1] = currentPage;
                                }
                                else
                                {
                                    //increase pages[i] array
```

```
                            int[] pagesTmp = new int[counts[i] * 2];
                            int p = 0;
                            copyPages:
                            {
                                pagesTmp[p] = pages[i][p];
                                p++;
                                if (p < counts[i] - 1)
                                    goto copyPages;
                            }
                            pages[i] = pagesTmp;
                            pages[i][counts[i] - 1] = currentPage;
                        }
                        goto endFor;
                    }
                    i++;
                    goto checkWords;
                }

                newWord: //it`s new word
                if (length == words.Length)
                {
                    string[] newWords = new string[(length + 1) * 2];
                    int[] newCounts = new int[(length + 1) * 2];
                    int[][] newPages = new int[(length + 1) * 2][];

                    i = 0;
                    forCopy:
                    {
                        if (i == length)
                        {
                            words = newWords;
                            counts = newCounts;
                            pages = newPages;
                            goto endCopy;
                        }
                        newWords[i] = words[i];
                        newCounts[i] = counts[i];
                        newPages[i] = pages[i];
                        i++;
                        goto forCopy;
                    }
                }

                endCopy:
                words[length] = word;
                counts[length] = 1;
                pages[length] = new int[] { currentPage };
                length++;
                word = "";
            }

            endFor:
            j++;
            if(j < str.Length)
                goto For1;
        }

        if (!reader.EndOfStream)
            goto readFile;
    }
```

```
endReading:
reader.Close();

//sorting (insertion sort)
int curr, k;
int[] currPages;
i = 1;
sort:
{
    curr = counts[i];
    word = words[i];
    currPages = pages[i];
    k = i - 1;
    whileSort:
    {
        if (k >= 0)
        {
            int symb = 0;
            compWords: //compare words
            {
                if (symb == words[k].Length || words[k][symb] < word[symb])
                    goto endWhile;

                if (symb + 1 < word.Length && words[k][symb] == word[symb])
                {
                    symb++;
                    goto compWords;
                }
            }

            counts[k + 1] = counts[k];
            words[k + 1] = words[k];
            pages[k + 1] = pages[k];
            k--;
            goto whileSort;
        }
    }
    endWhile:
    counts[k + 1] = curr;
    words[k + 1] = word;
    pages[k + 1] = currPages;
    i++;
    if (i < length)
        goto sort;
}

//output
StreamWriter writer = new StreamWriter(outPath);
i = 0;
write:
{
    if (counts[i] <= 100) { //if word isn`t igrnored
        writer.Write(words[i] + " - " + pages[i][0]);
        int j = 1;
        outPages:
        {
            if (j == counts[i])
                goto endOutPages;
            if(pages[i][j] != pages[i][j - 1]) // it`s new page
```

```
                        writer.Write(", "+pages[i][j]);
                    j++;
                    goto outPages;
                }
                endOutPages:
                writer.WriteLine();
            }
            i++;
            if (i < length)
                goto write;
        }

        writer.Close();
    }
}
```

Результати:

Input:
Книга «Pride and Prejudice»

Output:
Перших 6 рядків з файлу виводу
abatement - 99
abhorrence - 111, 160, 167, 263, 299, 306
abhorrent - 276
abide - 174, 318
abiding - 177
abilities - 72, 74, 107, 155, 171, 194

Висновок:

В межах даної роботи необхідно було необхідно вивчити імперативне програмування та виконати два завдання. Для виконання завдань необхідно було дотримуватися імперативної парадигми програмування та не порушувати накладених на роботу обмежень, які були описані вище в розділі завдання. Для виконання я обрала мову C#, яка підтримує використання goto. Також під час виконання роботи я використала деякі класи з простіру імен System.IO. А саме класи StreamReader і StreamWriter, які були використані для зчитування інформації з файлу та запис в нього, а саме методи цих класів Read() (зчитує символ з потоку), Readline() (зчитує стрічку з потоку), Write() (записує дані в потік), Writeline() (записує стрічку в потік), Close() (закриває файл і потік) та властивість EndOfStream (булеве значення чи знаходиться позиція поточного потоку в кінці потоку).