НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. Ігоря Сікорського»
Кафедра інформатики та програмної інженерії

Звіт
з лабораторної роботи №1 з дісципліни
“МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ”
Імперативне програмування

Виконав: Студент II курсу гр. ІТ - 02
Луговець Тетяна Ігорівна

Перевірив: Очеретяний Олександр Костянтинович

2022 р.

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.

Лістинг програми:

```
using
System;
          using System.IO;


          namespace MP_1
          {


              /**
           * The {@code TermFrequency} display N
           * of the most common words and the corresponding frequency of their repetition,
          in descending order.
           * Ignored stop words and normalized capitalization
           */
              class TermFrequency
              {
                  /**
              * @param inPath path to the file from which the data will be read
              * @param outPath path to the file where the result will be written
              * @param word words in a file
              * @param amount word counter
              * @param length word length
              * @param wordAmount the number of words we want to display, entered through
          the console,
              *                   in this case it is ignored and displays all the words
              * @param words sentence line
              */
                  static void Main(string[] args)
                  {
                      string inPath = "input.txt";
                      string outPath = "output.txt";
                      string[] word = new string[0];
                      int[] amount = new int[0];
                      int length = 0, i;
                      int wordAmount;
```

```
            wordAmount = int.Parse(Console.ReadLine());
            string words = "";


            //reading a file from a path
            StreamReader reader = new StreamReader(inPath);
        term_frequency:
            {
                if (reader.EndOfStream)
                    goto endReading;
                //normalizing the use of capital letters
                char symbol = (char)reader.Read();


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


                //ignore stop words
                if (words != "" && symbol != '-' && symbol != '\'' && words !=
"the" && words != "for" && words != "of" && words != "an" && words != "on")
                {
                    i = 0;


                newWords: //check if it`s a new word
                    {
                        if (i == length)
                            goto uniqueWords;
                        if (words == word[i])
                        {
                            amount[i]++;
                            words = "";
                            if (reader.EndOfStream)
                                goto endReading;
                            goto term_frequency;
                        }
                        i++;
```

```
                goto newWords;
            }


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


                string[] newWord = new string[(length + 1) * 2];
                int[] newAmount = new int[(length + 1) * 2];


                i = 0;
            copy:
                {
                    if (i == length)
                    {
                        word = newWord;
                        amount = newAmount;
                        goto end;
                    }
                    newWord[i] = word[i];
                    newAmount[i] = amount[i];
                    i++;
                    goto copy;
                }


            }


        end:
            word[length] = words;
            amount[length] = 1;
            words = "";
            length++;
        }


        if (!reader.EndOfStream)
            goto term_frequency;
    }


endReading:
    reader.Close();
    int curr, j;
```

```
            i = 1;


        //sorting
        sort:
            {
                curr = amount[i];
                words = word[i];
                j = i - 1;
            sortLoop:
                {
                    if (j >= 0 && amount[j] < curr)
                    {
                        amount[j + 1] = amount[j];
                        word[j + 1] = word[j];
                        j--;
                        goto sortLoop;
                    }
                }


                amount[j + 1] = curr;
                word[j + 1] = words;
                i++;
                if (i < length)
                    goto sort;
            }


            //writing the result to a file
            StreamWriter writer = new StreamWriter(outPath);
            i = 0;
        write:
            {
                writer.WriteLine(word[i] + " - " + amount[i]);
                i++;
                if (i < length)
                    goto write;
            }


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

Опис алгоритму:

1. Зчитуємо посимвольно текстовий файл input.txt за допомогою класу StreamReader бібліотеки System.IO
2. Зміна регістру здійснюється при перевірці на символ у додаванні його в змінну поточного слова
3. Після зчитування перевіряємо, чи є це стоп-словом
4. Перевіряємо, чи зустрічалося вже це слово. Перевірка відбувається на кожній ітерації з використанням безумовного переходу на ділянку з «циклом» і поверненням на мітку після
5. Сортуємо масив тільки за кількістю входжень слова
6. Записування результату у файл output.txt

Функції

Для цього завдання було використано такі функції, як зчитування тексту з файлу, зчитування значення з консолі та парсінг строки.

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.

Лістинг програми:

```
using
System.IO;


            namespace MP_2
            {
                class Program
                {
                    /**
                * @param inPath path to the file from which the data will be read
                * @param outPath path to the file where the result will be written
                * @param lines the number of lines that will separate pages
                * @param words words in a file
                * @param pages array of pages separated by 45 @lines
                * @param amount word counter
                * @param length word length
                * @param this_page current page
                * @param word sentence line
                */
                     static void Main(string[] args)
                     {
                         string inPath = "input.txt";
```

```
    string outPath = "output.txt";
    int lines = 45;
    int length = 0, i, string_ = 0;
    int this_page = 1;
    string word = "";
    string[] words = new string[0];
    int[] amount = new int[0];
    int[][] pages = new int[0][];


    //reading a file from a path
    StreamReader reader = new StreamReader(inPath);
read:
    {
        if (reader.EndOfStream)
            goto end;


        //split pages
        string str = reader.ReadLine();
        if (string_ == lines)
        {
            this_page++;
            string_ = 0;
        }
        string_++;


        int j = 0;
    loop:
        {
            if (j == str.Length)
                goto endLoop;
            //normalizing the use of capital letters
            char symbol = str[j];
            if (symbol >= 'A' &&  'Z' >= symbol )
            {
                word += ((char)(symbol + 32));
                if (j + 1 < str.Length)
                    goto endLoop;
            }
            else if (symbol >= 'a' && 'z' >= symbol)
            {
                word += symbol;
                if (j + 1 < str.Length)
                    goto endLoop;
            }
```

```
                    if (word != "" && symbol != '-' && symbol != '\'' && word
!= "the" && word != "for" && word != "of" && word != "an" && word != "on")
                    {
                        i = 0;
                    newWords: //check if it's new word
                        {
                            if (i == length)
                                goto uniqueWords;
                            if (word == words[i]) //the word occurs not for
the first time
                            {
                                word = "";
                                if (amount[i] > 100) //the word is ignored if
it occurs more than 100 times in the text
                                {
                                    goto endLoop;
                                }
                                amount[i]++;
                                if (amount[i] <= pages[i].Length) //move to
another page?
                                {
                                    pages[i][amount[i] - 1] = this_page;
                                }
                                else
                                {
                                    //increase the number of pages
                                    int[] pagesTmp = new int[amount[i] * 2];
                                    int p = 0;
                                copy: //save past pages
                                    {
                                        pagesTmp[p] = pages[i][p];
                                        p++;
                                        if (p < amount[i] - 1)
                                            goto copy;
                                    }
                                    pages[i] = pagesTmp;
                                    pages[i][amount[i] - 1] = this_page;
                                }
                                goto endLoop;
                            }
                            i++;
                            goto newWords;
                        }


                    uniqueWords: //a new word appears
                        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;
                copyLoop:
                    {
                        if (i == length)
                        {
                            words = newWords;
                            amount = newCounts;
                            pages = newPages;
                            goto endCopyLoop;
                        }
                        newWords[i] = words[i];
                        newCounts[i] = amount[i];
                        newPages[i] = pages[i];
                        i++;
                        goto copyLoop;
                    }
                }


            endCopyLoop:
                words[length] = word;
                amount[length] = 1;
                pages[length] = new int[] { this_page };
                length++;
                word = "";
            }


        endLoop:
            j++;
            if (j < str.Length)
                goto loop;
        }
        if (!reader.EndOfStream)
            goto read;
    }


end:
    reader.Close();
    //sorting
    int l;
```

```
            int k;
            int[] page;
            i = 1;
        sort:
            {
                l = amount[i];
                word = words[i];
                page = pages[i];
                k = i - 1;
            whileSort:
                {
                    if (k >= 0)
                    {
                        int symb = 0;
                    compare: //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 compare;
                            }
                        }


                        amount[k + 1] = amount[k];
                        words[k + 1] = words[k];
                        pages[k + 1] = pages[k];
                        k--;
                        goto whileSort;
                    }
                }
            endWhile:
                amount[k + 1] = l;
                words[k + 1] = word;
                pages[k + 1] = page;
                i++;
                if (i < length)
                    goto sort;
            }


            //output
```

```
                StreamWriter writer = new StreamWriter(outPath);
                i = 0;
            write:
                {
                    if (amount[i] <= 100)
                    { //if there are less than 100 words
                        writer.Write(words[i] + " - " + pages[i][0]);
                        int j = 1;
                    pagesLoop:
                        {
                            if (j == amount[i])
                                goto endPagesLoop;
                            if (pages[i][j] != pages[i][j - 1]) // split pages
                                writer.Write(", " + pages[i][j]);
                            j++;
                            goto pagesLoop;
                        }
                    endPagesLoop:
                        writer.WriteLine();
                    }
                    i++;
                    if (i < length)
                        goto write;
                }


                writer.Close();

        }
        }
    }
```

1. Алгоритм роботи програми цілком схожий з попереднім з завдання 1, програмний код був цілком перебудованим з коду першого завдання з деяким додаванням
2. Пострічно зчитуємо файл
3. Через кожні 45 стрічок змінюємо номер поточної сторінки
4. Добавляємо кількість входжень слова та також сторінку, де воно зустрілося
5. Перевіряємо, чи зустрічалось це слово більш 100 разів, якщо так – ігноруємо його
6. Сортуємо масив за алфавітом
7. Записуємо результат у файл

Функції

Для цього завдання було використано такі функції, як зчитування та запис тексту у файл.