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

МЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ

Лабораторна робота №1

Виконав: Студентка гр. ІТ – 02
Скачкова Анастасія Дмитрівна

Перевірив: Очеретяний Олександр Костянтинович

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**.

Ось такий вигляд матимуть ввід і відповідно вивід результату програми:
Input:
White tigers live mostly in India
Wild lions live mostly in Africa

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

Опис алгоритму:

1. Ініціалізувати масив із стоп-словами та необхідні змінні
2. Зчитати посимвольно файл за допомогою класу StreamReader
3. Перевірити наявність великої літери та зменшити, якщо вона існує
4. Сформувати слово з малих літер у змінній word
5. Порівняти змінну word із стоп-словами та обнулити при збігу
6. Занести слово у масив слів.
7. Перевірити розмірність масиву слів. Якщо кількість слів більша за його розмір, збільшити масив удвічі.
8. Продовжувати зчитування, доки символи у файлі не закінчаться.
9. Ініціалізувати масиви для унікальних слів та кількості повторів та необхідні змінні.
10. Перевірити чи було занесено поточне слово до масиву унікальних, занести, якщо ні.
11. Підрахувати кількість повторів слова з масиву унікальних у масиві слів.
12. Відсортувати масив унікальних слів у порядку спадання кількості повторень за допомогою бульбашкового сортування.

13.Вивести на консоль та записати у файл результат.

Лістинг програми:

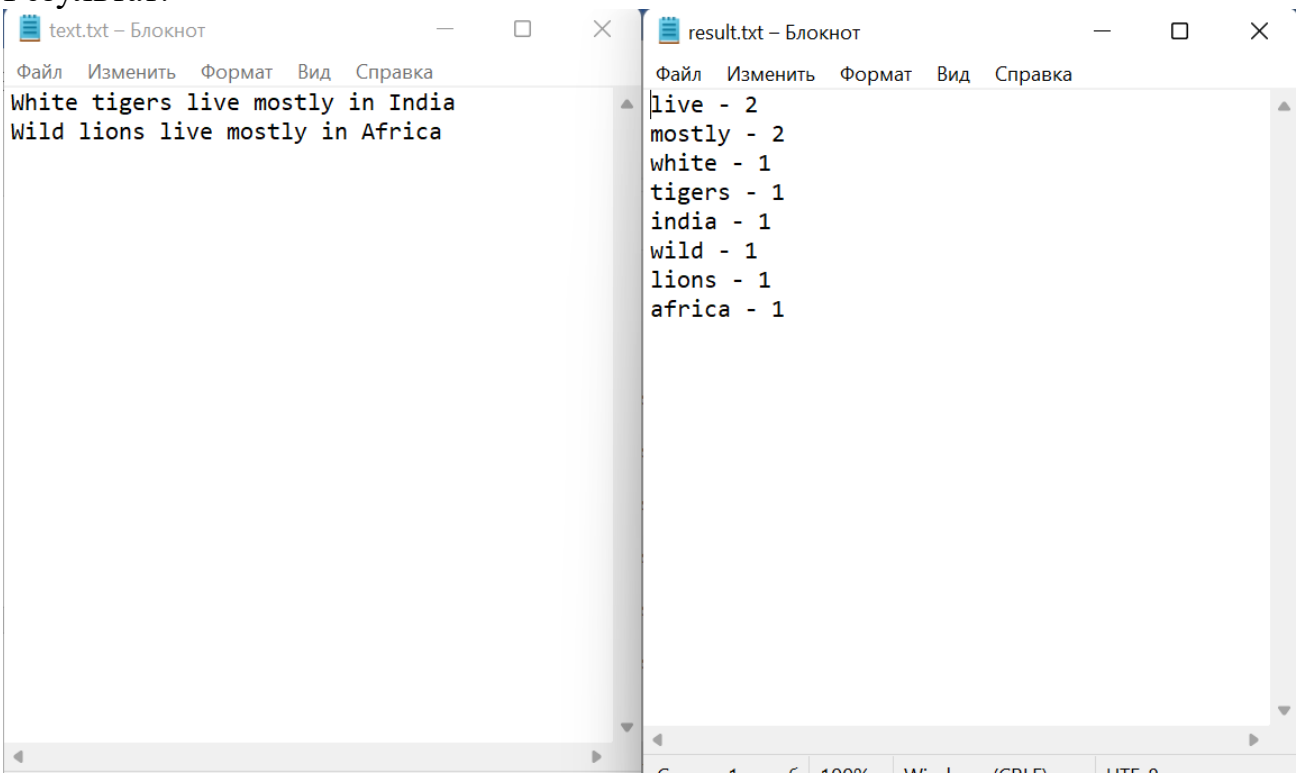
```
static void Main(string[] args)
        {
            string[] stopWords = new[] { "the", "of", "are", "a", "and", "or", "at", "by",
"for", "am", "to", "him", "they", "do", "i", "you", "we", "they", "he", "she", "it", "its",
"that", "from", "their", "this", "is", "with", "about", "like", "as", "was", "were", "have",
"has", "had", "if", "but", "in", "on", "will" };
            int length = 5;
            string[] words = new string[length];
            int[] pages = new int[length];
            int n = 0;
            string path = "text.txt";

            using (StreamReader sr = new StreamReader(path)) // read symbols for a word
            {
                string word = "";
                Symbols:
                char symb = (char)sr.Read();

                if (symb >= 'A' && symb <= 'Z')
                    symb += (char)32;
                if (symb >= 'a' && symb <= 'z')
                {
                    word = word + symb;
                    goto Symbols;
                }

                if (symb == '\n' || symb == ' ')
                {
                    if (word == "")
                    {
                        goto Symbols;
                    }
                    int ind = 0;
                    Check:
                    if (ind != stopWords.Length)
                    {
                        if (stopWords[ind] == word)
                        {
                            word = "";
                            goto Symbols;
                        }
                        ind++;
                        goto Check;
                    }

                    words[n] = word;
                    word = "";
                    n++;
                }

                if (length <= n) //if array index out
                {
                    string[] tmpWords = words;
                    int currIndex = 0;
                    length *= 2;
```

```
            words = new string[length];
            Move:
            words[currIndex] = tmpWords[currIndex];
            currIndex++;
            if (currIndex != n)
            {
                goto Move;
            }
            goto Symbols;
        }


        if (!sr.EndOfStream)
        {
            goto Symbols;
        }
    }

    string[] unique = new string[length];
    int[] countUnique = new int[length];
    bool sovp = false;
    int i = 0;
    int add = 0;
    int j = 0;
    int repeats = 0;
    int k = 0;

    // wrapping words in array with unique words
    CheckUnique:
    if(words[i]==unique[j] && j< length-1)
    {
        sovp = true;
    }
    if(j<length-1)
    {
        j++;
        goto CheckUnique;
    }
    else
    {
        if(!sovp)
        {
            unique[add] = words[i];
            add++;
        }
        sovp = false;
        i++;
        j = 0;
        if (i == length - 1)
        {
            i = 0;
            goto Check1;
        }
        goto CheckUnique;
    }



    // count of repeats
    Check1:
```

```
if (unique[i] != words[j] && j < length-1)
{
    j++;
    goto Check1;
}
if (unique[i] == words[j] && j < length-1 && words[j] != null)
{
    repeats++;
    j++;
    goto Check1;
}
if(j==length-1)
{
    countUnique[i] = repeats;
    repeats = 0;
    i++;
    j = 0;
    if (i == length - 1)
        goto Exit;


    goto Check1;
}
Exit:
i = 0;
int tmp;
string temp;

BubSort:
i++;
j = 0;
if(i < length-1)
{
    Sort:
    if (j < length - 1 - i)
    {
        if (countUnique[j] < countUnique[j + 1])
        {
            tmp = countUnique[j];
            countUnique[j] = countUnique[j + 1];
            countUnique[j + 1] = tmp;
            tmp = 0;
            temp = unique[j];
            unique[j] = unique[j + 1];
            unique[j + 1] = temp;
        }
        j++;
        goto Sort;
    }
    else
        goto BubSort;
}

using StreamWriter sw = new StreamWriter("result.txt");
Result:
if (length-1 != k && unique[k]!=null)
{
   Console.WriteLine($"{unique[k]} - {countUnique[k]}");
   sw.WriteLine($"{unique[k]} - {countUnique[k]}");
   k++;
   goto Result;
```

```
            }


        }
    }
```

Результат:



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:

abatement - 89
abhorrence - 101, 145, 152, 241, 274, 281
abhorrent - 253
abide - 158, 292

Опис алгоритму:

1. Ініціалізувати масив із стоп-словами та необхідні змінні
2. Зчитати посимвольно файл за допомогою класу StreamReader

3. Перевірити наявність великої літери та зменшити, якщо вона існує
4. Сформувати слово з малих літер у змінній word
5. Збільшувати змінну row при кожному переході на новий рядок та підрахувати сторінку для кожного слова.
6. Порівняти змінну word із стоп-словами та обнулити при збігу
7. Занести слово та номер сторінки у масив слів.
8. Перевірити розмірність масиву слів. Якщо кількість слів більша за його розмір, збільшити масив удвічі.
9. Продовжувати зчитування, доки символи у файлі не закінчаться.
10. Ініціалізувати масиви для унікальних слів та кількості повторів, необхідні змінні, а також двомірний масив із сторінками
11. Перевірити чи було занесено поточне слово до масиву унікальних, занести, якщо ні.
12. Підрахувати кількість повторів слова з масиву унікальних у масиві слів та занести сторінки поточного слова у двомірний масив.
13. Відсортувати масив унікальних слів в алфавітному порядку за допомогою бульбашкового сортування.
14. Вивести на консоль та записати у файл результат.

Лістинг програми:

```
static void Main(string[] args)
        {
            string[] stopWords = new[] { "the", "of", "are", "a", "and", "or", "at", "by",
"for", "am", "to", "him", "they", "do", "does", "dont", "i", "you", "your", "yours", "we",
"they", "he", "she", "it", "its", "that", "from", "their", "this",
                "is", "with", "about", "like", "as", "was", "were", "have", "has", "had",
"if", "but", "in", "on", "will" , "why", "who", "which", "while"};
            int length = 5;
            (string, int)[] words = new (string, int)[length];
            int n = 0;
            int row = 1;
            int page = 1;
            string path = "text.txt";

            using (StreamReader sr = new StreamReader(path)) // read symbols for a word
            {
                string word = "";
                Symbols:
                char symb = (char)sr.Read();

                if (symb >= 'A' && symb <= 'Z')
                    symb += (char)32;
                if (symb >= 'a' && symb <= 'z')
                {
                    word = word + symb;
                    goto Symbols;
                }

                if (symb == '\n' || symb == ' ')
                {
```

```
            if (word == "")
            {
                goto Symbols;
            }

            if (symb == '\n')
            {
                row++;
                page = row / 45 + 1;
            }

            int ind = 0;
            Check:
            if (ind != stopWords.Length)
            {
                if (stopWords[ind] == word)
                {
                    word = "";
                    goto Symbols;
                }
                ind++;
                goto Check;
            }

            words[n].Item1 = word;
            words[n].Item2 = page;
            word = "";
            n++;
        }

        if (length <= n) //if array index out
        {
            (string, int)[] tmpWords = words;
            int cur = 0;
            length *= 2;
            words = new (string, int)[length];
            Move:
            words[cur] = tmpWords[cur];
            cur++;
            if (cur != n)
            {
                goto Move;
            }
            goto Symbols;
        }



        if (!sr.EndOfStream)
        {
            goto Symbols;
        }
    }
    //for (int i = 0; i < length; i++)
    //    Console.WriteLine(words[i]);

    string[] unique = new string[length];
    int[] countUnique = new int[length];
    int[,] pages = new int[length,length];
    bool sovp = false;
    int i = 0;
```

```
int add = 0;
int j = 0;
int repeats = 0;
int k = 0;
int str = 0;

// wrapping words in array with unique words
CheckUnique:
if (words[i].Item1 == unique[j] && j < length - 1)
{
    sovp = true;
}
if (j < length - 1)
{
    j++;
    goto CheckUnique;
}
else
{
    if (!sovp)
    {
        unique[add] = words[i].Item1;
        add++;
    }
    sovp = false;
    i++;
    j = 0;
    if (i == length - 1)
    {
        i = 0;
        goto Check1;
    }
    goto CheckUnique;
}



// count of repeats
Check1:

if (unique[i] != words[j].Item1 && j < length - 1)
{
    j++;
    goto Check1;
}
if (unique[i] == words[j].Item1 && j < length - 1 && words[j].Item1 != null)
{
    repeats++;
    pages[i, str] = words[j].Item2;
    str++;
    j++;
    goto Check1;
}
if (j == length - 1)
{
    countUnique[i] = repeats;
    repeats = 0;
    i++;
    str = 0;
    j = 0;
    if (i == length - 1)
```

```
                goto Exit;


            goto Check1;
        }
        Exit:
        i = 0;
        int index = 0;
        int tmp;
        int ch;
        string temp;

        BubSort:
        i++;
        j = 0;
        ch = 0;
        if (i < length - 1)
        {
            Sort:
            if (j < length - 1 - i)
            {
                ch = 0;
                Chars:
                if (unique[j + 1] != null)
                {
                    if (unique[j][ch] > unique[j + 1][ch])
                    {
                        temp = unique[j];
                        unique[j] = unique[j + 1];
                        unique[j + 1] = temp;

                        tmp = countUnique[j];
                        countUnique[j] = countUnique[j+1];
                        countUnique[j+1] = tmp;

                        index = 0;
                        Swap:
                        if (pages[j + 1, index] != 0 || pages[j, index] != 0)
                        {
                                tmp = pages[j, index];
                                pages[j, index] = pages[j + 1, index];
                                pages[j + 1, index] = tmp;
                                index++;
                                goto Swap;
                        }
                    }
                    if (unique[j][ch] == unique[j + 1][ch] && ch < unique[j + 1].Length-
1 && ch < unique[j].Length-1)
                    {
                        ch++;
                        goto Chars;
                    }
                }
                j++;
                goto Sort;
            }
            else
                goto BubSort;
        }

        using StreamWriter sw = new StreamWriter("result.txt");
```

```
            Result:
            if (length - 1 != k && unique[k] != null)
            {
                Console.Write($"{unique[k]} -  ");
                sw.Write($"{unique[k]} - ");
                Str:
                if (str != length && pages[k, str] != 0)
                {
                    If (pages[k, str]!=pages[k,str+1])
                    {
                        Console.Write($"{pages[k, str]} ");
                        sw.Write($"{pages[k, str]} ");
                    }
                    str++;
                    goto Str;
                }
                Console.WriteLine("");
                sw.WriteLine("");
                str = 0;
                k++;
                goto Result;
            }
        }
    }
```

Результат: