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 Ящук Денис Андрійович _____
(№ 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:

Суть алгоритму доволі проста, спочатку ми зчитуємо дані із текстового файлу записуючи їх до змінної типу string, яку надалі використовуватимемо як масив char-ів (змінна text). Також оголосимо

масив words_arr куди будемо записувати усі значущі слова знайдені в тексті.

Потім за допомогою конструкції GOTO, що замінятиме цикл вайл проходимо масив text до кінця, записуючи кожне слово до нового масиву слів (words_arr). Кінцем слова вважатимемо будь-який символ, що не є великою або малою літерою латинці ([a-zA-Z]) або дефісом(-). Для корректного підрахунку кількості слів необхідно перевести усі символи до нижнього регістру. Для коректного переводу символу до малого регістру використаємо їх ASCII код додаючи 32, якщо символ виявиться великою літерою латиниці. І нарешті перед додаванням слова до масиву перевіримо чи не є воно стоп-словом заданим попередньо.

Оголосимо масиви words_only_once_arr та words_once_count_arr, куди записуватимемо слова без повторів із масиву words_arr та їх кількість в цьому масиві відповідно. Далі знову ж за допомогою конструкції GOTO проходимо масив words_arr до його закічнення. Для кожного елемента перевіряємо чи був він записаний до масиву words_only_once_arr за допомгою GOTO. Якщо слово ще нез’являлося то додаємо його до words_only_once_arr, а до words_once_count_arr додаємо елемент зі значенням 1. Якщо ж слово вже додане до масиву words_only_once_arr, то зчитуємо його позицію й додаємо до елемента на тій же позиції з масива words_once_count_arr одиницю.

Далі сортуємо масив words_once_count_arr бульбашкою, де при зміні позицій елементів змінюємо позиції відповідних елементів з масиву words_only_once_arr.

В кінці виводимо зміст двох масивів з відповідним форматуванням.

Завдання 2:

Початок програми дублює Завдання 1, з відмінністю того, що оголошується двомірний масив pages_words_arr, куди в подальшому записуватимуться масиви слів для кожної сторінки. Кінцем сторінки вважається досягненнея 45 символу нового рядка.

Далі повторюємо процес підрахунку кількості слів.

Наступним кроком проходимо масив з унікальними словами і записуємо до масиву words_only_once_arr_less_than_100 ті з них, в яких відповідний елемент із масиву words_once_count_arr менше за 100.

Далі знову використовуємо сортування бульбашкою для масива words_only_once_arr_less_than_100. Для порівняння сусідніх слів будемо порівнювати відповідні ASCII коди char-ів у цих словах.

Після чого проходимо поелементно масив words_only_once_arr_less_than_100. Виводимо слово на екран. Наступним кроком перевіряємо для кожного слова на якій сторінці воно розміщене в масиві pages_words_arr, якщо слово знайдене то додаємо його до масиву word_pages. Далі проходимо по масиву word_pages, й виводимо його елемент на екран з відповідним форматуванням.

3. Опис програмного коду

Завдання 1:

```
using System;
using System.IO;


namespace Task1
{
    class task1
    {
        static void Main(string[] args)
        {
            string text = File.ReadAllText(@"WriteText.txt");
            int text_length = text.Length;
            int topN = 300;
            int i = 0;
            string current_word = "";
            string[] words_arr = new string[1000000];
            int word_count = 0;
            while_loop:
                if((text[i] >= 65) && (text[i] <= 90) || (text[i]
>= 97) && (text[i] <= 122) || text[i] == 45)
                {
                    if ((text[i] >= 65) && (text[i] <= 90))
                    {
```

```
                        current_word += (char) (text[i] + 32);
                    }
                    else
                    {
                        current_word += text[i];
                    }
                }
                else
                {
                    if (current_word != "" && current_word != null
&& current_word != "-" && current_word != "no" && current_word !=
"from" && current_word != "the" && current_word != "by" &&
current_word != "and" && current_word != "i" && current_word !=
"in" && current_word != "or" && current_word != "any" &&
current_word != "for" && current_word != "to" && current_word !=
"\"" && current_word != "a" && current_word != "on" &&
current_word != "of" && current_word != "at" && current_word !=
"is" && current_word != "\n" && current_word != "\r" &&
current_word != "\r\n" && current_word != "\n\r")
                    {
                        words_arr[word_count] = current_word;
                        word_count++;
                    }
                    current_word = "";
                }
                i++;
                if(i < text_length)
                {
                    goto while_loop;
                }
                else
                {
                    if (current_word != "" && current_word != null
&& current_word != "-" && current_word != "no" && current_word !=
"from" && current_word != "the" && current_word != "by" &&
current_word != "and" && current_word != "i" && current_word !=
"in" && current_word != "or" && current_word != "any" &&
current_word != "for" && current_word != "to" && current_word !=
"\"" && current_word != "a" && current_word != "on" &&
current_word != "of" && current_word != "at" && current_word !=
```

```
"is" && current_word != "\n" && current_word != "\r" &&
current_word != "\r\n" && current_word != "\n\r")
                    {
                        words_arr[word_count] = current_word;
                        word_count++;
                    }
                }
            string[] words_only_once_arr = new string[1000000];
            int[] words_once_count_arr = new int[1000000];




            int amount_of_words = words_arr.Length;
            i = 0;
            int insertPos = 0;
            bool shouldInsert = true;
            int j = 0;
            int dubs = 0;
            while_loop_count:
                insertPos = 0;
                shouldInsert = true;
                int current_length = words_only_once_arr.Length;
                j = 0;


                    for_loop:
                    if (j < current_length &&
words_only_once_arr[j]!=null)
                    {
                        if (words_only_once_arr[j] ==
words_arr[i])
                        {
                            insertPos = j;
                            shouldInsert = false;
```

```
                    goto end_for_loop;

                }
                j++;
                goto for_loop;
            }
        end_for_loop:
        if (shouldInsert)
        {
            words_only_once_arr[i - dubs] = words_arr[i];
            words_once_count_arr[i - dubs] = 1;
        }
        else
        {
            words_once_count_arr[insertPos] += 1;
            dubs++;
        }
        i++;
        if (i < amount_of_words && words_arr[i]!=null)
        {

            goto while_loop_count;
        }
    int length = words_once_count_arr.Length;
    j = 0;
    int inner_i = 0;


sort_loop:
        if(j < length && words_once_count_arr[j] != 0)
        {
            inner_i = 0;
            sort_inner_loop:
```

```
                        if(inner_i < length - j - 1 &&
words_once_count_arr[inner_i] != 0)
                        {
                            if (words_once_count_arr[inner_i] <
words_once_count_arr[inner_i + 1])
                            {
                                int temp =
words_once_count_arr[inner_i];
                                words_once_count_arr[inner_i] =
words_once_count_arr[inner_i + 1];
                                words_once_count_arr[inner_i + 1]
= temp;
                                string temp2 =
words_only_once_arr[inner_i];
                                words_only_once_arr[inner_i] =
words_only_once_arr[inner_i + 1];
                                words_only_once_arr[inner_i + 1] =
temp2;
                            }
                            inner_i++;
                            goto sort_inner_loop;
                        }


                    j++;
                    goto sort_loop;
                }
            int k = 0;
            print_loop:
                if (k < length && words_only_once_arr[k] != null
&& k < topN)
                {
                    Console.WriteLine("{0} - {1}",
words_only_once_arr[k], words_once_count_arr[k]);
                    k++;
                    goto print_loop;
                }
        }
    }
```

```
}
```

Завдання 2:

```
using System;
using System.Collections.Generic;
using System.IO;

namespace Task2
{
    class task2
    {
        static void Main(string[] args)
        {
            string text = File.ReadAllText(@"WriteText.txt");
            int text_length = text.Length;
            int i = 0;
            string current_word = "";
            string[] words_arr = new string[100000];
            string[,] pages_words_arr = new string[10000, 10000];
            int word_count = 0;
            int row_count = 0;
            int page_count = 0;
            int page_word_counter = 0;
        while_loop:
            if ((text[i] >= 65) && (text[i] <= 90) || (text[i] >=
97) && (text[i] <= 122) || text[i] == 45 || text[i] == 234 ||
text[i] == 225 || text[i] == 224)
            {
                if ((text[i] >= 65) && (text[i] <= 90))
                {
                    current_word += (char)(text[i] + 32);
                }
                else
                {
                    current_word += text[i];
```

```
                }
            }
            else
            {
                if (text[i] == '\n')
                {
                    row_count++;
                }
                if (row_count > 45)
                {
                    page_count++;
                    page_word_counter = 0;
                    row_count = 0;
                }
                if (current_word != "" && current_word != null &&
current_word != "-" && current_word != "no" && current_word !=
"from" && current_word != "the" && current_word != "by" &&
current_word != "and" && current_word != "i" && current_word !=
"in" && current_word != "or" && current_word != "any" &&
current_word != "for" && current_word != "to" && current_word !=
"\"" && current_word != "a" && current_word != "on" &&
current_word != "of" && current_word != "at" && current_word !=
"is" && current_word != "\n" && current_word != "\r" &&
current_word != "\r\n" && current_word != "\n\r")
                {

                    words_arr[word_count] = current_word;
                    word_count++;
                    pages_words_arr[page_count, page_word_counter]
= current_word;
                    page_word_counter++;
                }
                current_word = "";
            }
            i++;
            if (i < text_length)
            {
                goto while_loop;
```

```
            }
            else
            {
                if (current_word != "" && current_word != null &&
current_word != "-" && current_word != "no" && current_word !=
"from" && current_word != "the" && current_word != "by" &&
current_word != "and" && current_word != "i" && current_word !=
"in" && current_word != "or" && current_word != "any" &&
current_word != "for" && current_word != "to" && current_word !=
"\"" && current_word != "a" && current_word != "on" &&
current_word != "of" && current_word != "at" && current_word !=
"is" && current_word != "\n" && current_word != "\r" &&
current_word != "\r\n" && current_word != "\n\r")
                {
                    words_arr[word_count] = current_word;
                    word_count++;
                }
            }
            string[] words_only_once_arr = new string[100000];
            int[] words_once_count_arr = new int[100000];
            int amount_of_words = words_arr.Length;
            i = 0;
            int insertPos = 0;
            bool shouldInsert = true;
            int j = 0;
            int dubs = 0;
        while_loop_count:
            insertPos = 0;
            int current_length = words_only_once_arr.Length;
            j = 0;
            shouldInsert = true;
        for_loop:
            if (j < current_length && words_only_once_arr[j] !=
null)
            {
                if (words_only_once_arr[j] == words_arr[i])
                {
                    insertPos = j;
```

```
                    shouldInsert = false;
                    goto end_for_loop;

                }
                j++;
                goto for_loop;
            }
        end_for_loop:
            if (shouldInsert)
            {
                words_only_once_arr[i - dubs] = words_arr[i];
                words_once_count_arr[i - dubs] = 1;
            }
            else
            {
                words_once_count_arr[insertPos] += 1;
                dubs++;
            }
            i++;
            if (i < amount_of_words && words_arr[i] != null)
            {
                goto while_loop_count;
            }
            int length = words_once_count_arr.Length;
            int k = 0;
            string[] words_only_once_arr_less_than_100 = new
string[100000];
            int LastInsert = 0;
            less_than_100_loop:
                if (k < length && words_only_once_arr[k] != null )
                {
                    if (words_once_count_arr[k] <= 100)
                    {
```

```
words_only_once_arr_less_than_100[LastInsert] =
words_only_once_arr[k];
                        LastInsert++;
                    }
                    k++;
                    goto less_than_100_loop;
                }
            int write = 0;
            int sort = 0;
            bool toSwapWords = false;
            int counter = 0;
            int word_lenth_cur = 0;
            int word_lenth_next = 0;
            sort_loop:
                if(write< words_only_once_arr_less_than_100.Length
&& words_only_once_arr_less_than_100[write] != null)
                {
                    sort = 0;
                    inner_sort_loop:
                        if(sort<
words_only_once_arr_less_than_100.Length - write - 1 &&
words_only_once_arr_less_than_100[sort+1] != null)
                        {
                            word_lenth_cur =
words_only_once_arr_less_than_100[sort].Length;
                            word_lenth_next =
words_only_once_arr_less_than_100[sort + 1].Length;


                            int compare_lenth = word_lenth_cur >
word_lenth_next ? word_lenth_next : word_lenth_cur;


                            toSwapWords = false;
                            counter = 0;
                        check_alphabet:
```

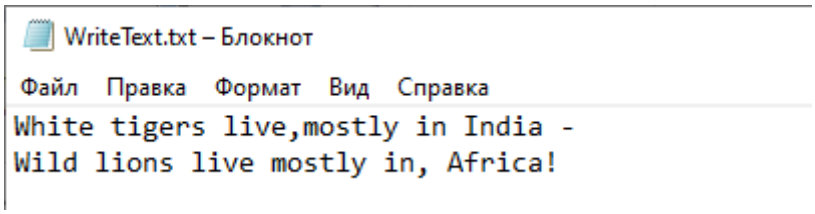
```
                            if
(words_only_once_arr_less_than_100[sort][counter] >
words_only_once_arr_less_than_100[sort + 1][counter])
                            {
                                toSwapWords = true;
                                goto check_alphabet_end;
                            }
                            if
(words_only_once_arr_less_than_100[sort][counter] <
words_only_once_arr_less_than_100[sort + 1][counter])
                            {
                                goto check_alphabet_end;
                            }
                            counter++;
                            if (counter < compare_lenth)
                            {
                                goto check_alphabet;
                            }
                        check_alphabet_end:
                        if (toSwapWords)
                        {
                            string temp =
words_only_once_arr_less_than_100[sort];

words_only_once_arr_less_than_100[sort] =
words_only_once_arr_less_than_100[sort + 1];
                            words_only_once_arr_less_than_100[sort
+ 1] = temp;
                        }
                            sort++;
                            goto inner_sort_loop;
                        }
                    write++;
                    goto sort_loop;
                }
                k = 0;
```

```
        int less_than_100_length =
words_only_once_arr_less_than_100.Length;




        print_loop:
                if (k < less_than_100_length &&
words_only_once_arr_less_than_100[k] != null)
                {
                    Console.Write("{0} - ",
words_only_once_arr_less_than_100[k]);
                    int first_dim = 0;
                    int second_dim = 0;
                    int[] word_pages = new int[100];
                    int pageInsert = 0;


                    check_page:
                        if(first_dim< 10000 &&
pages_words_arr[first_dim,0] != null)
                        {
                            second_dim = 0;
                            check_page_word:
                            if (second_dim < 10000 &&
pages_words_arr[first_dim, second_dim] != null)
                            {
                                if(pages_words_arr[first_dim,
second_dim] == words_only_once_arr_less_than_100[k])
                                {
                                    word_pages[pageInsert] =
first_dim + 1;

                                    pageInsert++;
                                    first_dim++;
                                    goto check_page;
                                }
                                second_dim++;
                                goto check_page_word;
```

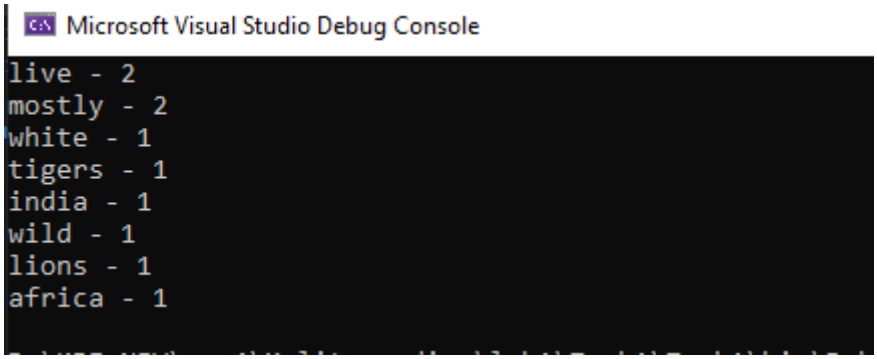
```
                            }


                            first_dim++;
                            goto check_page;
                        }
                    int tired_counte = 0;
                    pagination_loop:
                        if(tired_counte<100 &&
word_pages[tired_counte] != 0)
                    {
                        if(tired_counte!=99 &&
word_pages[tired_counte + 1] != 0)
                        {
                            Console.Write("{0}, ",
word_pages[tired_counte]);
                        }
                        else
                        {
                            Console.Write("{0}",
word_pages[tired_counte]);
                        }
                        tired_counte++;
                        goto pagination_loop;
                    }
                    Console.WriteLine();
                    k++;
                    goto print_loop;
                }
            }
        }
    }
```

4. Скріншоти роботи програмного застосунку

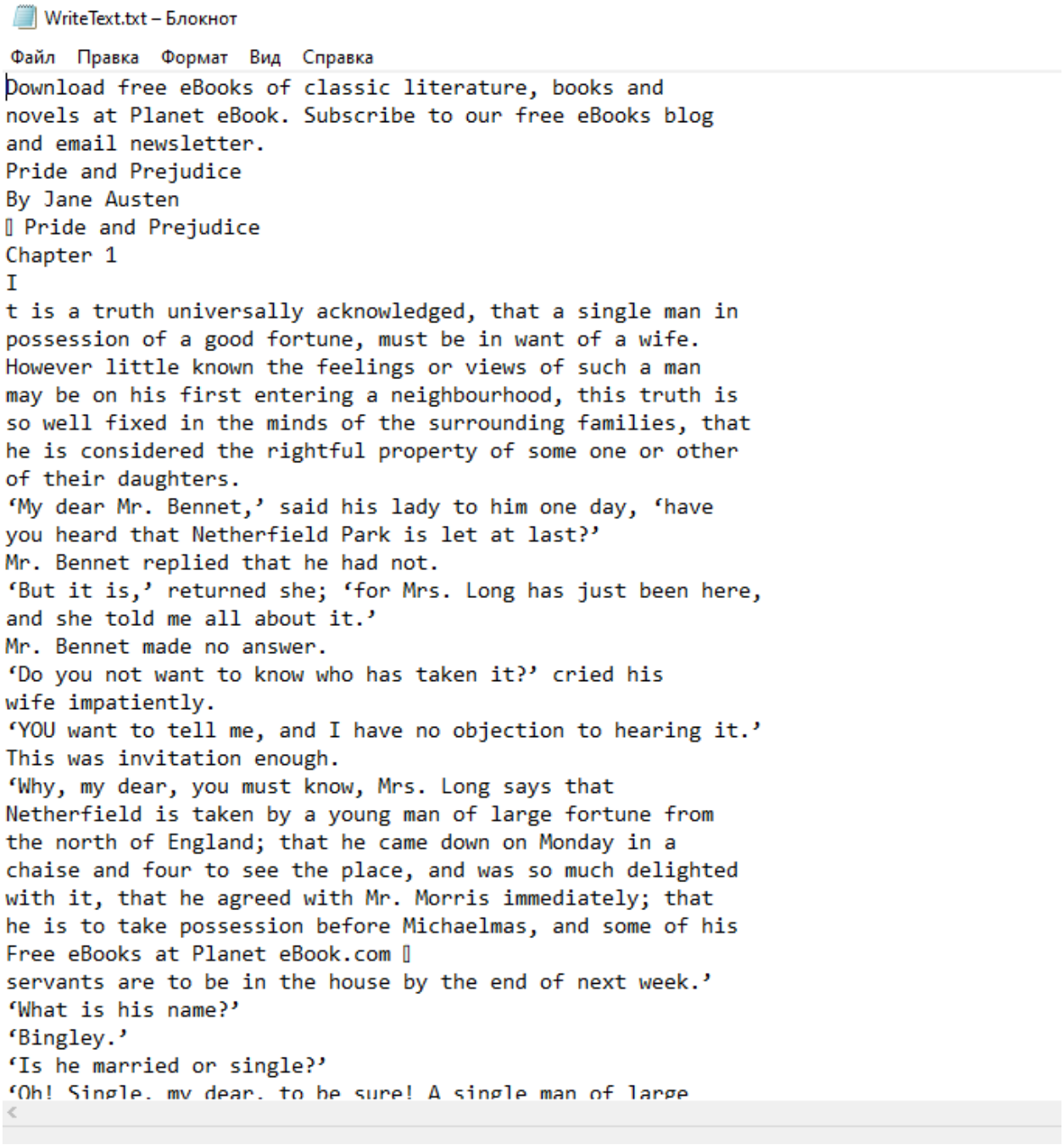
Завдання 1:
Input:



Output:



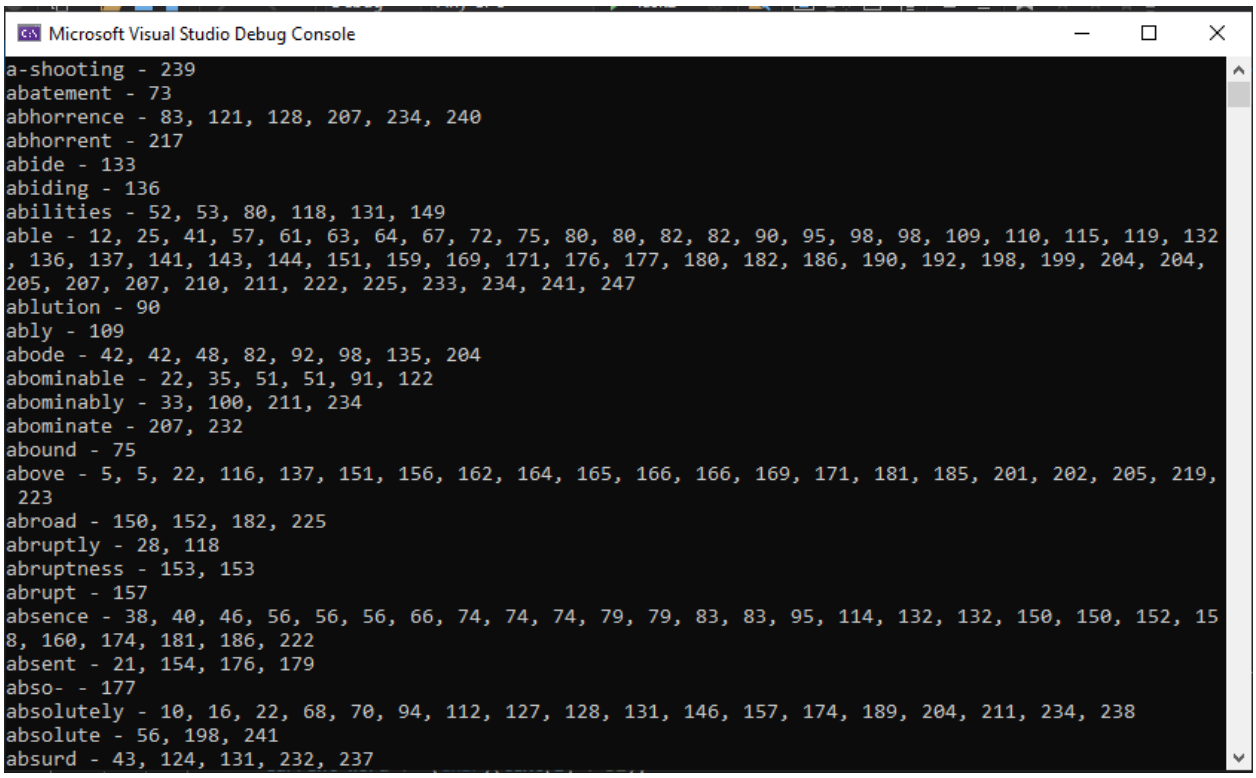
Завдання 2:
Input:


WriteText.txt – Блокнот

Файл Правка Формат Вид Справка

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:


Microsoft Visual Studio Debug Console
a-shooting - 239
abatement - 73
abhorrence - 83, 121, 128, 207, 234, 240
abhorrent - 217
abide - 133
abiding - 136
abilities - 52, 53, 80, 118, 131, 149
able - 12, 25, 41, 57, 61, 63, 64, 67, 72, 75, 80, 80, 82, 82, 90, 95, 98, 98, 109, 110, 115, 119, 132, 136, 137, 141, 143, 144, 151, 159, 169, 171, 176, 177, 180, 182, 186, 190, 192, 198, 199, 204, 204, 205, 207, 207, 210, 211, 222, 225, 233, 234, 241, 247
ablution - 90
ably - 109
abode - 42, 42, 48, 82, 92, 98, 135, 204
abominable - 22, 35, 51, 51, 91, 122
abominably - 33, 100, 211, 234
abominate - 207, 232
abound - 75
above - 5, 5, 22, 116, 137, 151, 156, 162, 164, 165, 166, 166, 169, 171, 181, 185, 201, 202, 205, 219, 223
abroad - 150, 152, 182, 225
abruptly - 28, 118
abruptness - 153, 153
abrupt - 157
absence - 38, 40, 46, 56, 56, 56, 66, 74, 74, 74, 79, 79, 83, 83, 95, 114, 132, 132, 150, 150, 152, 158, 160, 174, 181, 186, 222
absent - 21, 154, 176, 179
abso- - 177
absolutely - 10, 16, 22, 68, 70, 94, 112, 127, 128, 131, 146, 157, 174, 189, 204, 211, 234, 238
absolute - 56, 198, 241
absurd - 43, 124, 131, 232, 237