

Міністерство освіти та науки України
Національний Технічний Університет України
“Київський Політехнічний Інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

ЗВІТ
до лабораторної роботи № 1
з дисципліни
«Мультипарадигменне Програмування»

Виконав
Студент ФІОТ ІІІ
4 курсу групи ІІІ-02
Шниркова Д.О

Київ 2024

Завдання лабораторної роботи:

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

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
```

Приклад вхідних даних у файлі:



test.txt - Notepad

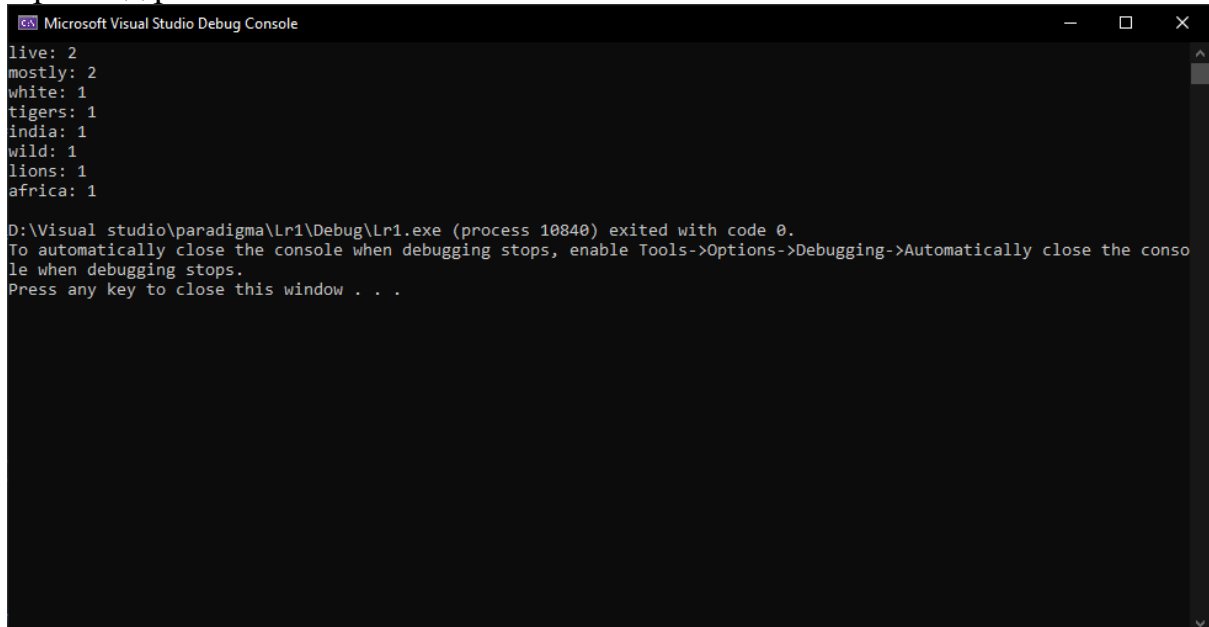
```
File Edit Format View Help  
White tigers live mostly in India  
Wild lions live mostly in Africa
```

Опис роботи алгоритму:

Цей код відкриває текстовий файл з іменем "test.txt" для зчитування. Він читає кожне слово з файлу, перетворює його у нижній регістр і перевіряє, чи належить слово до списку "stop_word" (слова, які ігноруються в обробці). Після зчитування кожного слова воно додається до вектора "words_cnt_list" разом з кількістю його входжень у тексті. Якщо слово вже присутнє у векторі "words_cnt_list", кількість його входжень збільшується на 1; в іншому випадку воно додається до вектора з кількістю входжень, рівною 1. Після завершення читання всіх слів вектор "words_cnt_list" містить унікальні слова з їх кількістю входжень. Далі відбувається сортування цього вектора за кількістю входжень кожного слова у зменшуючому порядку. Це досягається шляхом пошуку слова з

найбільшою кількістю входжень і обміном його з першим словом у векторі. Після цього виводиться слово з максимальною кількістю входжень, і процес повторюється для наступного слова. Коли всі слова відсортовано, програма завершує роботу і закриває вхідний файл.

Приклад роботи:



```
Microsoft Visual Studio Debug Console
live: 2
mostly: 2
white: 1
tigers: 1
india: 1
wild: 1
lions: 1
africa: 1

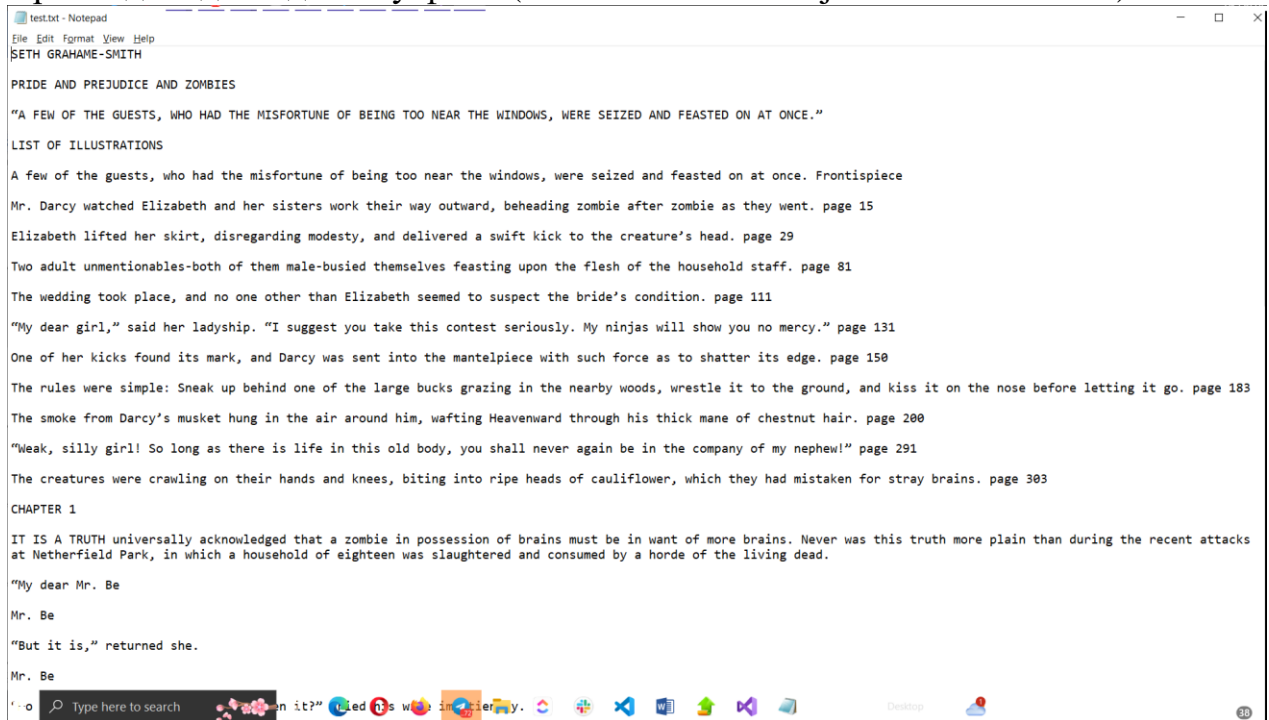
D:\Visual studio\paradigma\Lr1\Debug\Lr1.exe (process 10840) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

2. 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
```

Приклад вхідних даних у файлі(книга Pride and Prejudice and Zombies):



Опис роботи алгоритму:

Приклад роботи:

```
Select Microsoft Visual Studio Debug Console  
female: 36  
few: 1 24 36  
filled: 33  
filthy: 34  
fine: 10  
fit: 22  
five: 10 24  
flesh: 3 35  
follow: 33  
following: 17  
for: 7 10 12 15 18 20 23 26 31  
force: 4  
forget,: 18  
fortnight.ГҒҘ: 19  
fortune,: 14  
fortune,: 9  
found: 4  
four: 10  
free: 36  
freshly: 35  
fretfully.: 19  
friends.: 15  
from: 5 35  
gates: 22  
gentlemanlike,: 25  
girl: 29  
girl!: 6  
girl,ГҒҘ: 3  
girls: 14 29  
girls!ГҒҘ: 10 21  
give: 31  
giving: 26  
go.: 5  
go,: 16  
good: 20 21  
good-looking: 25  
gowns: 34  
grabbed: 33  
grazing: 5  
green: 35  
grey: 35  
ground,: 5  
guests,: 1 36  
had: 1 7 16 17 19 25 32 33 36  
hair.: 6  
hall,: 34  
hand: 33  
handle: 33  
hands: 6  
handsome: 29 31  
happily: 24  
has: 8 11 13 14  
hate: 27  
have: 15 18 27 29  
have,: 21  
he: 9 10 12 16 20 24 25 26 27 31  
he,: 27  
head,: 37  
head.: 2  
heads: 6  
hear: 20  
heard: 15 19  
her: 2 3 4 15 17 19 30 31 32 33 37
```

Висновок

При виконанні цієї роботи ознайомились з поняттям імперативної моделі програмування. В ході роботи було виконано 2 задачі на аналіз тексту за використанням зазначеної вище моделі. Набули навичок «застарілого» стилю написання коду та поглибили його розуміння.