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

# Лабораторна робота №1

**з курсу**

**Мультипарадигменне Програмування**

Виконав
студент групи ІТ-03:

Король К.В

Перевірив:

Київ 2021

## 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
```

## 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
```

**Задание №1**

**Код на гитхабе** task1.lang_with_go_to

Описание алгоритма:

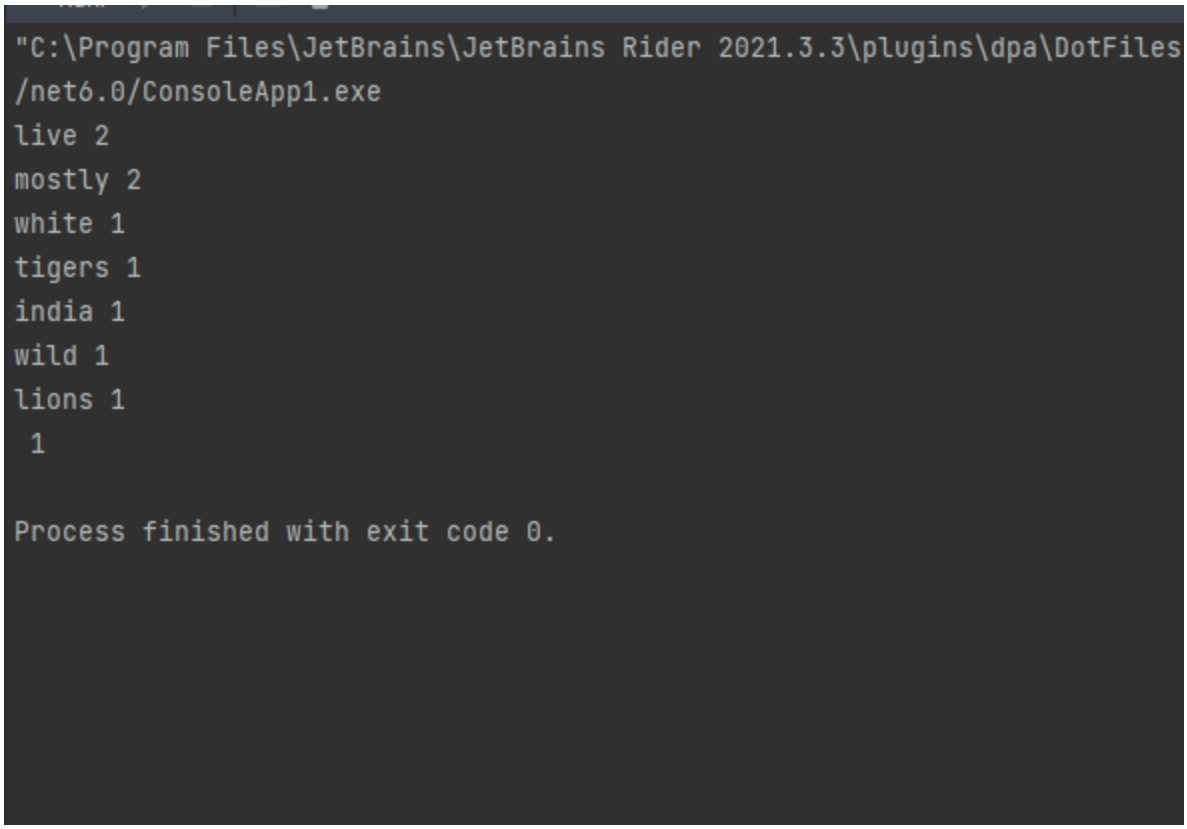
1) Считываем весь текст с файла
2) Считаем количество слов в файле
3) Дальше разделяем наш текст на слова и во время этого разделения сразу же приводим все слова к маленьким буквам. Для упрощения работы далее
4) Начинается основной цикл :

   4.1) Слово проверяется на совпадение со стоп словом. (Стоп слова мы задали ранее в массиве в нашей программе)

   4.2) - Если это стоп-слово то пропускаем его.

   - Если это не стоп-слово то проверяем его на совпадения. Работает это так, что у нас есть еще 1 массив с нашей структурой Структура(слово, повтор). Если наше слово уже присутствует в нашем массиве, тогда инкрементируем значение у поля “повтор”, если его не было в этом массиве еще, значит оно не проверялось раньше. Тогда мы добавляем его в массив и присваиваем значение “повтор” = 1
5) Сортируем наш массив со структурой с помощью Bubble sort.
6) Выводим в консоль

Результат работы:



**Задание №1**

**Код на гитхабе** task2.lang_with_go_to

Описание алгоритма:

1)Считываем весь текст с файла

2)Считаем количество слов в файле

3)Дальше разделяем наш текст на слова и записываем слова в массив с новой структурой в которой также храним страницу на которой было это слово и во время этого разделения сразу же приводим все слова к маленьким буквам. Для упрощения работы далее

```
wordsInFile[arrayIndex++] = new PairPage()
{
    Word = splitWord,
    Page = line / 45 + 1,
    IsUsed = false,
};
```

4) Начинается основной цикл :

// Алгоритм похожий с первого задания

4.1) У нас есть дополнительный массив с нашей новой структурой. В который мы записываем информацию о проверенных словах.

Если слово еще не встречалось, тогда создаем структуру с этим словом.

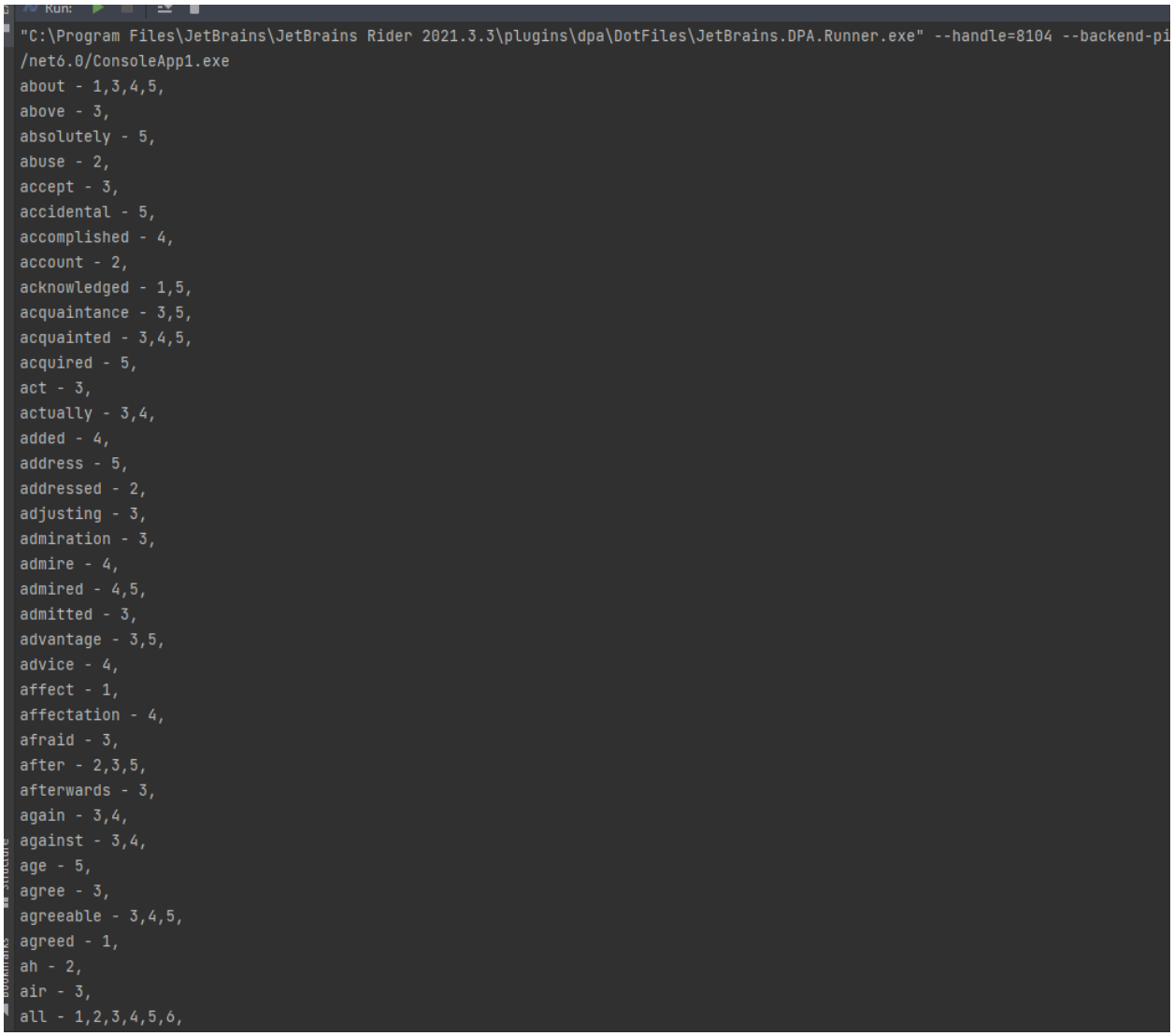
4.2) Если проверяемое слово уже было добавлено ранее в нашу структуру, значит в массив со страницами, который есть в нашей структуре, добавляем значение той страницы на которой расположено слово.

Если слово на этой странице уже встречалось, то тогда мы не записываем его, потому что мы считаем частоту встречи слова на разных страницах, а не на одной конкретной.

5) Сортируем наш массив со структурой с помощью Bubble sort по алфавиту.

6) Выводим в консоль “слово + страницы “

Результат работы:



**Вывод:**

На данной лаб. работе мы вспомнили что такое императивное программирование и выполнили два задания, что очень напомнило работу с нашим любимым Ассемблером. Мы не использовали циклы, функции, и другой сахар, который нам дает современный яп. Для перехода по коду мы использовали метки и команду goto