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

Звіт

з лабораторної роботи № 1 з дисципліни
«Мультипарадигменне програмування»

«Імперативне програмування»

Виконав(ла) *ІТ-03 Іванченко Ксенія Віталіївна* _____
(шифр, прізвище, ім'я, по батькові)

Перевірив *Очеретяний О.К.* _____
(прізвище, ім'я, по батькові)

Київ 2022

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.

Опис алгоритму розв'язку

Зчитування відбувається за допомогою os.Open() а scanner.Scan(). Далі кожен рядок посимвольно обробляється, символи "**.**", "**,**" та " " вважаються за кінець слова. Робота здійснюється з двома файлами: перший містить стоп слова, які після зчитування додаються до масиву; другий файл містить текст, в якому потрібно рахувати кількість різних слів. Для зберігання розв'язку використані 2 масиви, перший зберігає саме слово, а другий кількість повторів відповідного слова в тексті. Після посимвольної оброки йде перевірка чи є таке вже таке слово в масиві-розв'язку. Якщо є то збільшує число повторів на один, якщо немає, то додаємо слово в кінець масиву. Коли оброблений весь текст, відбувається сортування масивів-розв'язків за спаданням кількості повторів відповідного слова. Враховуючи, що треба використовувати конструкції GOTO, сортування відбувається за алгоритмом bubble sort. Далі йде вивід результату та одночасна

перевірка чи є слово, яке ми виводимо стопсловом. Якщо є, то просто його пропускаємо.

Алгоритм реалізовано мовою Go. Для тестування взято короткий опис книги «Pride and Prejudice».

Результат роботи алгоритму для task1

```
C:\Users\Admin\AppData\Local\Temp\GoLand\___go_build_task1_go__1_.exe
1 . and --- 59
2 . mr --- 47
3 . elizabeth --- 32
4 . her --- 29
5 . darcy --- 27
6 . that --- 24
7 . he --- 18
8 . she --- 18
9 . his --- 17
10 . bingley --- 13
11 . with --- 12
12 . jane --- 12
13 . bennet --- 11
14 . him --- 11
15 . has --- 10
16 . family --- 8
17 . mrs --- 7
18 . after --- 7
19 . collins --- 7
20 . wickham --- 7
```

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.

Опис алгоритму розв’язку

Алгоритм практично так сама як і попередній, але при зчитуванні також відбувається підрахування кількості рядків, кожні 45 рядків оновлюється номер сторінки і записується в двомірний масив. Далі сортування відбувається за алфавітом та при виведенні результату вже треба вивести всі слова, а не перші n.

Алгоритм реалізовано мовою Go. Для тестування взято короткий опис книги «Pride and Prejudice».

Результат роботи алгоритму для task2

```
C:\Users\Admin\AppData\Local\Temp\GoLand\___go_build_task2_go.exe
1.15-year-old---2
2.19th---1
3.20---1
4.27---2
5.about---1 2
6.absence---2
7.accept---2
8.accepted---2
9.accepts---1 2
10.accompanies---2
11.accusation---2
12.accuses---2
13.acquaintance---2
14.adaptations---1
15.address---2
16.after---1 2
17.afterward---2
18.afterwards---1
19.again---2
```

Висновок

Під час виконання лабораторної роботи була ознайомлена з основами імперативного прогрумування та виконала 2 завдання в суто імперативній парадигмі на мові Go, з використання конструкції GOTO та без циклів, функцій, вбудованого сортування чи структур даних.

GitHub