Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №1
Мультипарадигменне програмування

Виконав
студент групи ІТ-03:

Очкас Д.Г.

Перевірив:

ас. Очеретяний О.К.

Київ 2020

Завдання:
Практична робота складається із двох 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

Перед початком опису алгоритмів зауважу, що зчитування даних з файлів та запис даних неможливі без використання функцій, тому для повного виконання поставлених завдань було вирішено використати виключно ті функції, які зчитують дані з файлу і записують дані у файл

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.

Алгоритм вирішення:

1. Зчитуємо стоп-слово у змінну (стоп слова розміщені у відповідному файлі з назвою "stop_words_english.txt")
2. Якщо масив стоп слів заповнений, виділяємо для нього вдвічі більшу пам'ять
3. Записуємо нове стоп слово до масиву
4. Якщо не досягнуто кінець файлу то йдемо до кроку 1
5. Якщо не досягнуто кінця файлу, де знаходиться текст
6. Зчитуємо слово з файлу
7. Переводимо слово у нижній регістр
8. Якщо дане слово - стоп-слово то переходимо до кроку 5
9. Якщо дане слово - не стоп-слово, то перевіряємо, чи воно зустрічалося раніше.
10. Якщо зустрічалося, то збільшуємо його кількість на 1

11. Якщо ні, то записуємо нове слово з кількістю 1
12. Переходимо до кроку 5
13. За допомогою звичайного сортування вставкою відсортовуємо пораховані слова за кількістю входжень в текст
14. Визначаємо, яка кількість слів має бути виведена (якщо в масиві менша кількість елементів, ніж значення константи N, то виводиться весь масив, в іншому випадку виводяться N перших значень масиву)
15. До файлу виводу записуємо дані в гарному форматі (<слово> - <кількість>)
16. Кінець

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.

Алгоритм вирішення:

1. Якщо не досягнуто кінця текстового файлу
2. Визначаємо поточний рядок
3. Визначаємо поточну сторінку
4. Зчитуємо рядок з файлу
5. Для кожного символу рядка розділяємо його на слова
6. Переводимо слово в нижній регістр
7. Для кожного слова, перевіряємо, чи зустрічалося воно раніше
8. Якщо ні, то записуємо нове слово і для нього створюємо зв'язний список з поточною сторінкою
9. Якщо так, то додаємо в зв'язний список слова поточну сторінку
10. Переходимо до кроку 1 поки не закінчиться файл
11. Після зчитування та занесення всіх даних сортуємо слова в алфавітному порядку за допомогою сортування вставкою
12. Записуємо результат в файл виводу в форматі <слово> - [номери сторінок]
13. Кінець

Вихідний код можна знайти за наступним посиланням:
https://github.com/Singachpuck/multiparadigm-programming

Висновок: Отже, після виконання цієї лабораторної роботи, я випробував на собі всі особливості програмування в 50-ті роки, використав на практиці конструкцію goto, розвинув алгоритмічне мислення та згадав, як програмувати деякі базові алгоритми (наприклад сортування вставкою)