Мультипарадигменне Програмування
Лабораторна робота 1
Тема: імперативне програмування
Виконав: студент групи ІП-02 Мацапура Ярослав Миколайович

Звіт

Лабораторна робота була виконана на мові C++, адже вона підтримує конструкцію goto. Робота розроблена за принципами імперативного програмування.

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.

Готовий код завдання знаходиться на GitHub.

Алгоритм самого коду:
Для початку із потрібного шматка тексту ми виокремимо всі окремі слова. Ці слова ми помістили в массив, збільшуючи його кожен раз, коли це потрібно за стандартною процедурою.
Для зберігання інформації про кількість створюємо матрицю із двома стовпчиками та потрібною кількістю рядків. Пробігаємося по массиву слів та рахуємо скільки разів трапилося кожне з них. Цю інформацію записуємо в матрицю у вигляді "[WORD] [QUANT]", тобто саме слово та поруч його кількість (у вигляді рядка). Для сортування використали сортування бульбашкою та відсортували саме рядки матриці по значенню кількості, що відповідає даному рядку.

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.

Готовий код завдання знаходиться на GitHub.

Алгоритм самого коду:
Алгоритм схожий на минулий. Але тепер ми можемо виділяти слова зразу в матрицю, схожу на попередню. Кожен 45 рядок - нова сторінка, саме так ми її будемо визначати. Отже записуємо кожне слово в перший стовпець матриці і усі сторінки відповідно у другий (набір сторінок в нас знову у виді рядків, вони розділені пробілами). Якщо кількість появ більше за задане число, то цей рядок і слово ми видаляємо з матриці та саме слово заносимо у масив, так мовити, заборонених слів. Тепер кожне нове слово в матриці буде перевірено на забороненість, та буде ігноруватися, якщо опиниться в цьому списку. Після формування матриці, маємо слова та їх відповідні набори сторінок, проте якщо слово з’явилося на сторінці декілька разів, сторінка буде продубльована. Це можна було вирішити або під час додавання самих сторінок, проте так ми би не змогли б відслідкувати всі рази коли слово траплялося, що потрібно для ігнорування занадто популярних слів в тексті. Отже цю проблему ми вирішимо вже після, шляхом форматування нашого набору. Ми просто перевіряємо попереднє число та наступне, і якщо вони однакові, то виводимо це число, тобто номер сторінки, лише раз, попри дублікати в початковому записі.

Висновок:
Виконавши лабораторну роботу, ми розробили дві програми для вирішення двох схожих задач, стараючись все робити за принципами імперативного програмування. Попри те, що ми замінили цикли на конструкцію goto, я також намагався не використовувати саморобні структури, хоча їх присутність зробила би виконання в рази легше.