**Общая постановка задачи:**

Разработать и реализовать алгоритм формирования перекрестных ссылок (сформировать таблицу, в которой все слова будут расположены в алфавитном порядке и для каждого слова будет указан список строк его нахождения (по возрастанию номеров строк), используя STL)

Работа должна поддерживать некоторую систему команд.

**Описание работы**

Программа представляет собой консольное приложение, предоставляющее пользователю возможность создать словарь перекрестных ссылок для текстового файла, а также взаимодействовать со словарями через определённую систему команд.

При формировании словаря перекрестных ссылок должны учитываться все слова, записанные буквами английского алфавита или цифрами, а также их комбинациями. Любые знаки препинания не учитываются. Регистр букв также не должен учитываться. Если слово встречается в ряде строк, то номера строк должны быть перечислены вместе в порядке возрастания, разделенные пробелом. Вывод самих слов осуществляется в алфавитном порядке.

*Файл text-analyzer.hpp и text-analyzer.cpp:*

Объявление и имплементация класса TextAnalyzer, обязанность которого заключается в выполнении системы команд пользователя. Объект класса создается конструктором по умолчанию. Для формирования словаря перекрестных ссылок применяется метод analyze, получающий на вход название файла, из которого будет совершаться чтение и возвращающий сформированный словарь. Метод getDictonary() позволяет иметь доступ к полученному словарю перекрестных ссылок после вызова метода analyze. При повторном анализе старый словарь удаляется.

Для вывода полученной таблицы применяется метод printAnalysis, принимающий на вход название файла или выходной поток, в который будет совершаться запись.

Имплементация класса достигается с помощью объекта контейнера STL std::map с ключом-строкой std::string и значением – последовательным контейнером STL std::vector.

*Файл main.cpp:*

Имплементация функции main для использования функционала класса TextAnalyzer с помощью консоли и считываемых фалов. Программа будет предлагать пользователю ввести файл с начальными данными, файл для вывода результата, и файл, в котором будет храниться ожидаемый результат, чтобы после выполнения команд, проверить корректность работы программы.

*Формат входных данных*

Файл с начальными данными содержит строки, в которых вначале идет номер команды, а затем данные для её корректной работы. Гарантировано: каждая строка содержит данные, разделитель между данными – один пробел. Корректность данных, а также сами вводимые файлы необходимо проверять. Строка может содержать данные, которые не нужны для выполнения команды, их необходимо игнорировать. В случае невозможности работы с указанными файлами, а также некорректностью данных в них, пользователю должно быть выведено предупреждение, а программа должна прекратить свое выполнение.

На выбор предлагаются следующие команды, закодированные определёнными цифрами:

1 – analyzeText

2 – uniteDictionaries

3 – subtractionDictionaries

4 – getIntersectionDictionaries

5 – searchWord

Каждая строка файла соответственно содержит

1 < inFilename > < way > …

2 < countFiles > < File1 > < File2 > < File3 > …

3 < countFiles > < File1 > < File2 > < File3 > …

4 < countFiles > < File1 > < File2 > < File3 > …

5 < inFilename > < word > …

**Система команд**

* Команда analyzeText <inFilename> <ostream> <way> принимает файл, из которого читаем данные, формирует словарь перекрестных ссылок и выводит результат в другой файл или на консоль в зависимость от выбранного пути. Во время анализа текста для словаря текст читается построчно, каждая строка анализируется с помощью регулярного выражения /[a-zA-Z0-9]+/, позволяющего определить все слова и числа в тексте без лишних символов. Way задаётся словами console – если результат необходимо вывести на консоль, file – если результат необходимо вывести в файл.

Результат: Таблица с двумя столбцами. Первый столбец – слова, второй – номера строк, где встречается соответствующее слово. Ширина первого столбца выбирается относительно самого длинного слова или шапки таблицы.

* Команда subtractionDictionaries < arrayDictionaries > < ostream > принимает массив словарей и поток вывода, работает по принципу вычитания всех словарей из первого (из первого вычитаем второй, из результата третий и т.д., пока есть словари). Вычитание происходит только в случае полного совпадения пары ключ-значение.
* Команда uniteDictionaries < arrayDictionaries > < ostream > принимает массив словарей и поток вывода, работает по принципу объединения всех словарей в первом. Объединение строки словаря не происходит только при полном совпадении пары ключ-значение. Если ключи одинаковы, а значения нет, тогда происходит объединение значений. Например в dictionary1 {cat 2} в dictionary2 {cat 1} в результате должны получить dictionary1 {cat 1 2}.
* Команда getIntersectionDictionaries < arrayDictionaries > < ostream > принимает массив словарей и поток вывода, работает по принципу пересечения всех словарей в первом. Пересечение происходит только в случае полного совпадения пары ключ-значение.
* Команда searchWord <dictionary> <word> проверяет нахождения слова в словаре. Имеет тип bool и в зависимости от результата выводит соответствующее сообщение.

**Интеграционное тестирование**

В рамках настоящей работы предусмотрены следующие интеграционные тесты:

|  |  |  |  |
| --- | --- | --- | --- |
| Описание | Данные | Ожидаемый результат | Соответствие фактического результата ожидаемому (+/-) |
| Проверка правильности формирования словаря перекрёстных ссылок | cat dog moon  moon solt advanter  space want moon cat |  | + |
| В файле для словаря встречается пустая строка | a b c d a  a b c d e f g h g  b c e |  | + |
| Проверка на незначимость регистра входных данных файла словаря | ab AB  Ab aB AB |  | + |
| Проверка на незначимость других символов во входных данных файла словаря | a?-<a a?в.a,\_a |  | + |
| Проверка правильности работы команды searchWord (в словаре есть данные) | Содержание файла:  cat dog moon  moon solt advanter  space want moon cat  Передаваемое слово:  moon | Key-value pair present in dictionary | + |
| Проверка правильности работы команды searchWord (в словаре нет данных) | Содержание файла:  Передаваемое слово:  moon | Key - value pair not present in dictionary | + |
| Объединение словарей (все словари имеют данные) | File1:  cat dog moon  moon solt advanter  File2:  moon solt advanter dog  space want moon cat  cat  File3:  dog moon abx  moon solt advanter dog |  | + |
| Объединение словарей (один из словарей не имеет данных) | File1:  cat dog moon  moon solt advanter  File2:  File3:  dog moon abx  moon solt advanter dog |  | + |
| Вычитание словарей  (все словари имеют данные) | File1:  cat dog moon  moon solt advanter  File2:  moon solt advanter dog  space want moon cat  cat  File3:  dog moon abx  moon solt advanter dog |  | + |
| Вычитание словарей  (первый словарь не имеет данных) | File1:  File2:  moon solt advanter dog  space want moon cat  cat  File3:  dog moon abx  moon solt advanter dog |  | + |
| Пересечение словарей (все словари имеют данные) | File1:  cat dog moon  moon solt advanter  File2:  moon solt advanter dog  space want moon cat  cat  File3:  dog moon abx  moon solt advanter dog |  | + |
| Пересечение словарей (один из словарей не имеет данные) | File1:  cat dog moon  moon solt advanter  File2:  moon solt advanter dog  space want moon cat  cat  File3: |  | + |
| Пересечение словарей (первый словарь не имеет данные) | File1:  File2:  moon solt advanter dog  space want moon cat  cat  File3:  dog moon abx  moon solt advanter dog |  | + |
| Открытие не существующего файла со словарями | Testing.txt | Can't open file | + |
| Считывание файла с входными данными, запись результата в файл (result.txt) и сравнение файлов result.txt и expected.txt | Входной файл(test.txt)    dictionary1.txt    dictionary2.txt    dictionary3.txt | Файл ожидаемого результата (expected.txt) | + |

**Исходный код**