МИНОБРНАУКИ РОССИИ
Санкт-Петербургский государственный
электротехнический университет «ЛЭТИ»
им. В.И. Ульянова (Ленина)
Кафедра САПР

КУРСОВАЯ РАБОТА

по дисциплине «Программированние»
Тема: Обработка текстовой информации

Студенты гр. 9892 _____ Лескин К.А.

_____ Миллер В.В.

Преподаватель _____ Кузьмин С.А.

Санкт-Петербург
2020

Задание
на курсовую работу

Студенты Лескин К.А., Миллер В.В.

Группа 9892

Тема работы: Обработка текстовой информации

Исходные данные:

Разработать программу, осуществляющую обработку текстовой информации, согласно своему варианту задания.

Задачи данной курсовой работы включают выполнение следующих этапов:

1) ввод исходного текста, хранящегося в виде файла;
2) ввод исходных данных для обработки текста (например, подстрока для определения заданного места обработки; подстрока, соответствующая заданной обработке; требуемые символьные данные; и т.п.);
3) обработку текста, соответствующую индивидуальному заданию (опираясь на технику указателей и стандартные подпрограммы обработки ASCII строк);
4) сохранение файла, соответствующего обработанному тексту;
5) вывод обработанного текста на экран.

Индивидуальный вариант задания:

В тексте, разделенном на слова, подсчитать количество гласных и согласных. Упорядочить слова по содержащимся в них гласным и/или согласным, а также по их относительному соотношению. Для слов указать количество вхождений в текст.

Содержание пояснительной записки:
Требуемые разделы пояснительной записки: «Содержание», «Введение», Основные главы, «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
Не менее 30 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 9892 _____ Лескин К.А.

_____ Миллер В.В.

Преподаватель _____ Кузьмин С.А.

Аннотация

В данной курсовой работе необходимо реализовать программу, обрабатывающую текстовые данные, хранящиеся в виде файла, в соответствии с опцией, которую выбирает пользователь. Пользовательский интерфейс включает в себя меню, которое позволяет производить все необходимые манипуляции с исходными данными. Для выполнения данной курсовой работы будут использованы знания, полученные при выполнении лабораторных работ. В результате будет получена программа полностью соответствующая требованиям к индивидуальному заданию.

Summary

In this coursework, we need to implement a program that processes text data stored as a file, in accordance with the option that the user chooses. The user interface includes a menu that allows you to make all the necessary manipulations with the original data. In this coursework we will use the knowledge gained during laboratory work. As a result, the program will be fully compliant with the requirements for the individual task.

Содержание

Введение

Целью данной курсовой работы является углубление знаний в технологии программирования типовых задач обработки текстовых данных.

В ходе выполнения работы мы воспользуемся всеми знаниями, полученными при выполнении лабораторных работ. Содержание пояснительной записки к курсовой работе соответствует стандартным требованиям к пояснительным документам для программного продукта.

В данной курсовой работе предполагается использование функций для работы со строками библиотеки <string> [1] и функции работы с файлами библиотеки <fstream> [2], изученных в ходе выполнения предыдущих лабораторных работ. Ввод исходной строки осуществляется через внешний файл.

Вывод полученной новой строки осуществляется в новый файл.

Ввод других необходимых данных для выполнения программы (например, слов или же отдельных символов) может осуществляться как через отдельный файл, так и напрямую с консоли. При этом, необходимо вывести на консоль весь ход выполнения программы (комментарии по действиям программы, копии вводимых и выводимых данных).

При разработке программы необходимо учитывать и корректно обрабатывать исключительные ситуации, которые могут возникнуть при открытии для чтения или записи несуществующего файла. Строки и слова в программе должны быть представлены с помощью специального типа данных "string".

1 Внешние форматы хранения данных

Входной файл может иметь любой формат, программа в любом случае его считает. Для примера и тестирования был создан файл bigfile без формата, содержащий текст на английском языке и включающий в себя занки препинания и пунктуации.

Содержимое входного файла bigfile представленно на рисунке 1.1.

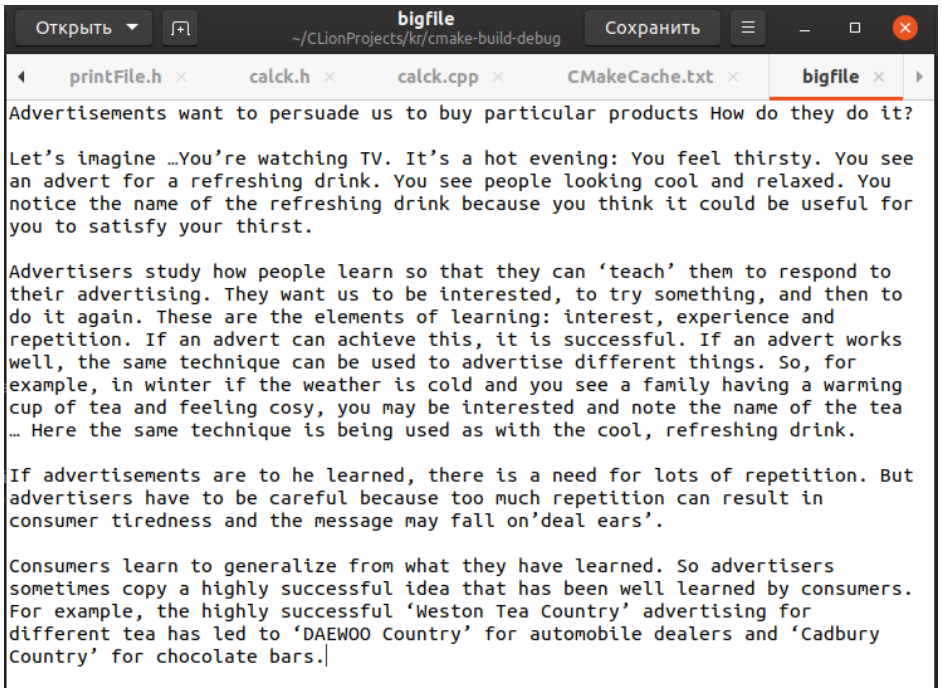

Рис. 1.1 – Входной файл bigfile

2 Внутренние форматы хранения данных

Для хранения содержимого файла используется динамический массив, состоящий из слов типа std::string. Указатель на этот массив хранится в main и инициализируется при открытии файла. Каждая ячейка массива является объектом std::string и в свою очередь хранит своё слово в отдельности.

Схема хранения данных представленна на рисунке 2.1, где n — количество слов в исходном файле.

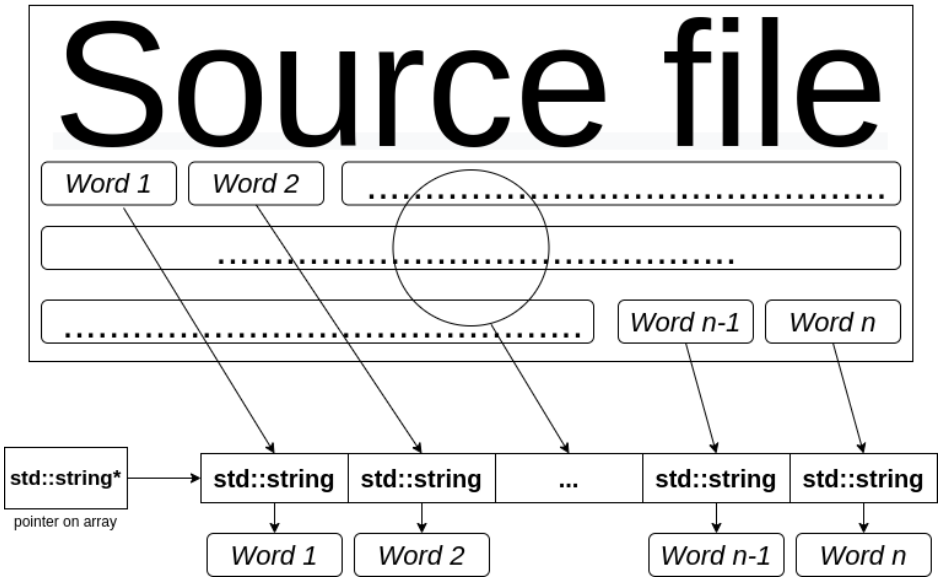

Рис. 2.1 – Формат хранения данных, полученных из файла

В данном контексте словом считается любая последовательность символов (от одного и больше), разделённая с двух сторон пробелами, либо символами табуляции (символ табуляции, символ переноса строки и пр.).

3 Описание пользовательских функций и модулей программы

Все функции и модули, реализованные в ходе выполнения данной курсовой работы, представленны в таблице 1.

Таблица 1 – Описание пользовательских функций и модулей программы

Имя модуля	Имя функции	Назначение	Параметры функции	Возвращаемое значение функции
openFile.h	openfile	Открытие файла	int &wordsSize, std::string &name	std::string*
countVC.h	countVC	Подсчёт количества гласных и согласных в тексте	const std::string *words, int wordsSize, int &V, int &C, int &N	void
writeFile.h	writeFile	Запись изменённого текста в файл	std::string *words, unsigned int wordsSize, char opt = '6'	void
countWords.h	countWords	Подсчёт количества слов в файле (без учёта повторений)	int wordsSize, std::string nameOfFile	bool
sortVC.h	sortVC	Сортировка слов в файле по заданным параметрам	std::string *words, const int &wordsSize, char &opt	void
printFile.h	printFile	Вывод содержимого файла на экран	const std::string *words, const int &wordsSize	void
calck.h	calck	Подсчёт коэффициента слова по его гласным и согласным	const double &v, const double &c, const char &opt	double

4 Описание интерфейса пользователя

При запуске программы пользователю пердоставляется меню из девяти пунктов (рисунок 4.1).

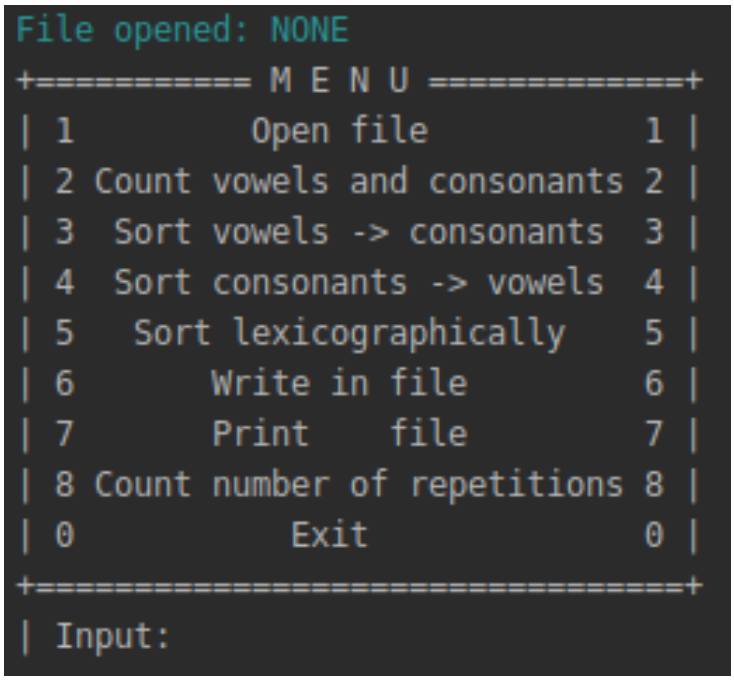

Рис. 4.1 – Главное меню

Вверху меню отображается имя текущего файла. По умолчанию при окрытии файл не открыт, и показывается NONE.

При выборе пункта меню Open file на экран выводятся файлы текущей директории. Если введено ошибочное имя файла выводится сообщение об ошибке, в противном случае над меню вместо NONE отображается имя открытого файла.

Если никакой файл не открыт, выборы путктов 2-8 влекут за собой вывод сообщения о том, что файл не открыт.

При выборе 2 пункта Count vowels and consonants на экран выводится количество гласных и согласных букв в тексте файла, а так же количество символов, которые не являются буквами. Стоит отметить, что считаются только буквы латинского алфавита.

Выбор пунктов 3-5 приводит к выводу на экран сообщения, в какой файл были записаны изменения. При неудачной попытке будет выведено сообщение об ошибке.

Пункт 6 предлагает пользователю ввести имя файла, в который он хочет записать изменения. Если файл с таким именем существует, про-

грамма предупредит об этом и попросит пользователя подтвердить перезапись существующего файла.

Пункт 7 выводит содержимое файла на экран.

Пункт 8 выводит все уникальные (не повторяющиеся) слова в формате:
[порядковый номер слова] [количесво повторов] [слово]
В конце списка выводится сообщение о количестве уникальных слов и самое часто встречающееся слово с количеством его повторений. Если все слова повторяются не более одного раза об этом будет выведено сообщение.

Выбор последнего пункта приводит к завершению программы с соответствующим сообщением.

Если пользовательский ввод содержит недопустимое значение, программа выведет сообщение о некорркtном вводе и вернёся в меню.

5 Описание алгоритма работы программы

Функция main

При запуске программы создаются переменные для хранения данных:

- символ для хранения выбранного пункта меню;
- указатель на массив слов (в данном массиве хранятся слова из файла);
- имя текущего файла;
- количество слов в текущем файле;
- три ячейки для хранения количества гланых, согласных и не-букв в тексте файла;

Запускается цикл.
ПОКА выборанный пункт меню НЕ РАВНО '0'

Выводится имя открытого файла.
ЕСЛИ имя файла пустое ПЕЧАТЬ "NONE"
ИНАЧЕ ПЕЧАТЬ имя файла

Выводится текст меню.

ВВОД пункта меню

ЕСЛИ 1 Удаляются все слова из массива. Счётчики букв устанавливаются в -1. Выводятся возможные файлы к открытию. Запускается функция openFile. Её возращаемое значение перезаётся указателю.

ЕСЛИ 2 Сначала проверяется, что в массиве есть слова. Для это смотрим на счётчик слов wordsSize. Потом проверяется, не посчитаны ли уже гласные и согласные. Для этого смотрим на счётчик гласных V. Если проверки прошли, запускается функция подсчёта countVC. В конце выводим сообщение с полученными данными.

ЕСЛИ 3 Сначала проверяется, что в массиве есть слова. Для это смотрим на счётчик слов wordsSize. Если слова есть, запускается функция sortVC.

ЕСЛИ 4 Сначала проверяется, что в массиве есть слова. Для это смотрим на счётчик слов wordsSize. Если слова есть, запускается функция sortVC.

ЕСЛИ 5 Сначала проверяется, что массив слов не пуст. Если слова есть, запускается функция std::sort из стандартной библиотеки STL, а изменения записываются в файл функцией writeFile. Инече выводится сообщение об ошибке.

ЕСЛИ 6 Сначала проверяется, что массив слов не пуст. Если слова есть, запускается функция writeFile. Инече выводится сообщение об ошибке.

ЕСЛИ 7 Запускается функция printFile.

ЕСЛИ 8 Запускается функция countWords.

ЕСЛИ 0 Программа завершается.

В остальных случаях выводится сообщение о неверном вводе.

Функция openFile

Проверяется входной параметр имени файла. Если он пустой, имя файла вводтся с клавиатуры. Иначе он сохраняется в локальную переменную. Открывается файл. В случае успеха выводится сообщение об успешном открытии и имя файла сохраняется в переменной, отвечающей за имя файла в функции main. Иначе выводится сообщение об ошибке и функция возвращает nullptr. Циклом проходимся по файлу и на каждое слово инкрементируем счётчик слов. Выделяем массив размером с количество слов в файле. Заново переоткрываем файл и считываем слова в массив. Закрываем файл и возвращаем указатель на созданный массив.

Функция countVC

- Идём по всему массиву

  Идём по каждому символу в слове

  Если символ - гласная буква, увеличиваем счётчик гласных

  Если символ - буква, увеличиваем счётчик согласных

  Иначе увеличиваем счётчик не-букв

Функция writeFile

Отталкиваемся от пункта, из которого была запущена функция.

- Если 2 - имя файла count.txt.
- Если 3 - имя файла VtoC.txt.
- Если 4 - имя файла CtoV.txt.
- Если 5 - имя файла lex.txt.

- Если 6 - имя файла вводится с клавиатуры пользователем.
- Если 8 - имя файла NumOfRep.txt.
- В остальных случаях выводится сообщение об ошибке.

Если функция запущена из 6 пункта, то проверяем, существует ли уже файл с таким именем и предупреждаем об этом пользователя, предоставляя выбор - перезаписать существующий файл или отменить операцию.

Проверяем успешность открытия файла и выводим сообщение.

Циклом записываем все слова из массива в файл.

Закрываем файл.

Функция countWords

Сначала проверяется, что в массиве есть слова. Для это смотрим на счётчик слов wordsSize. Далее создаём копию массива и сортируем её лексиграфически[3][4]. Так, одинаковые слова сгруппируются и не будут разбросаны по всему массиву. Например массив

$$1\ 4\ 8\ 5\ 7\ 3\ 8\ 8\ 4\ 3\ 2\ 2\ 9\ 4\ 8\ 1 \tag{1}$$

Перестроится в

$$1\ 1\ 2\ 2\ 3\ 3\ 4\ 4\ 4\ 5\ 7\ 8\ 8\ 8\ 8\ 9 \tag{2}$$

- Цикл по i от 0 до wordsSize
  - Выводим номер слова
  - Если i достигло конца массива, выводится счётчик количества слов и само слово. Цикл завершается
  - Цикл по j от i до wordsSize
    - Если слова по индексам i и j не равны
      - Выводится счётчик количества слов и само слово.
      - Если счётчик количества слов больше максимума, то максимумом устанавливается текущий счётчик, а в самое повторяющееся слово записывается текущее слово.
      - счётчик слов сбрасывается в 1
      - i = j
      - Увеличивается счётчик уникальных слов на 1
    - Иначе увеличивается счётчик количества слов на 1

    * Если j достигло конца массива, i приравнивается количеству слов в массиве. Выводится счётчик количества слов и само слово.

Выводится количество уникальных слов.

Если максимум больше 1, выводится самое часто встречающееся слово и его количество повторений. Иначе выводится сообщение о том, что все слова повторяются единожды.

Функция sortVC

Отталкиваемся от пункта, из которого была запущена функция.
Если не 3 и не 4 - выходим с ошибкой.

- Цикл по i от 0 до конца массива слов.
  - Цикл по каждому символу в слове.
    * Если символ - гласная буква, Увеличиваем количество гласных букв в i-ом слове на 1.
    * Если символ - буква, Увеличиваем количество согласных букв в i-ом слове на 1.
  - Расчитываем коэффициент k1 для i-го слова с помощью функции calc.
  - Если i больше 0

    Цикл по j от 0 до i
    * Расчитываем коэффициент k2 для j-го слова с помощью функции calc.
    * Если k2 $<$ k1, Меняем слова в массиве местами.

Записываем изменения в файл с помощью функции writeFile.

Функция printFile

Сначала проверяется, что в массиве есть слова. Для это смотрим на счётчик слов wordsSize. Если есть то:

- Цикл по i от 0 до конца массива
  - Выводим слово.
  - Если i+1 делится на 10 без остатка, переносим каретку на следующую строку.

Функция calc

На вход подаётся количество гласных (v) и согласных (c) букв слова и номер пункта меню, из которого запускалась функция.

- Если оба параметра больше 0
  - Если пункт меню равен 3
    k = v / c
  - Иначе
    k = c / v
- Если согласных в слове нет
  - Если пункт меню равен 3
    k = v + 1000
  - Иначе
    k = 0
- Если гласных в слове нет
  - Если пункт меню равен 3
    k = 0
  - Иначе
    k = c + 1000
- Иначе k = -1

Возвращаем k

6 Примеры работы программы

На рисунке 4.1 пердставлено главное меню программы.

Введём случайную последовательность символов. Ожидается, что программа выведет сообщение о некоррктном вводе и вернёся в меню. Результат на рисунке 6.1.

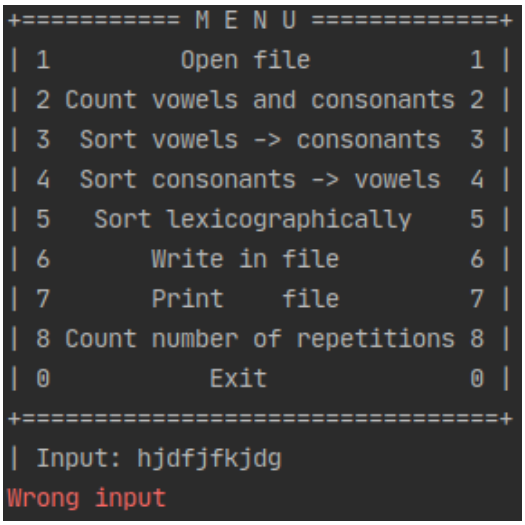

Рис. 6.1 – Ошибочный ввод

Так как никакой файл не открыт, полный функционал программы недоступен. Попоробуем выбрать пункт меню 2. Ожидается сообщение об ошибке и возврат в меню. Результат на рисунке 6.2.

Откроем загатовленный файл bigfile с помощью пункта меню 1 Open file. Ожидается, что на экран выведутся файлы текущей директории, а после открытия над меню вместо NONE будет отображается имя открытого файла. Результат на рисунке 6.3.

Выберем 2 пункт меню. Ожидается, что на экран выведется количество гласных и согласных букв в тексте файла, а так же количество символов, которые не являются буквами. Результат на рисунке 6.4

Выберем 3 пункт меню. Ожидается, что на экран выведется сообщение о том, что изменённое содержимое файла записано в новый файл VtoC.txt. Результат на рисунке 6.5. Содержимое файла VtoC.txt на рисунке 6.6.

Выберем 4 пункт меню. Ожидается, что на экран выведется сообщение о том, что изменённое содержимое файла записано в новый файл CtoV.txt. Результат на рисунке 6.7. Содержимое файла CtoV.txt на рисунке 6.8.

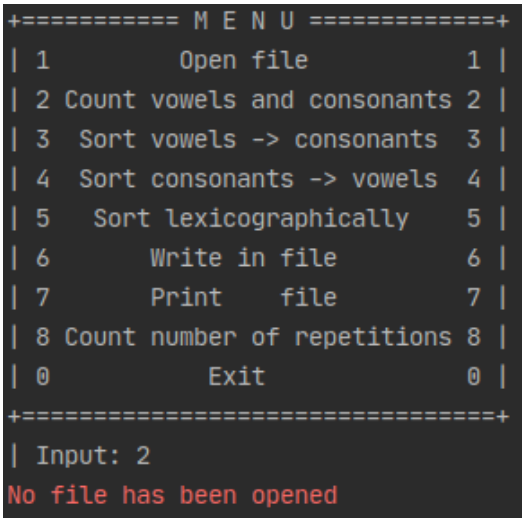

Рис. 6.2 – Выбор пункта меню, работающего с файлом, при отсутствии открытого файла

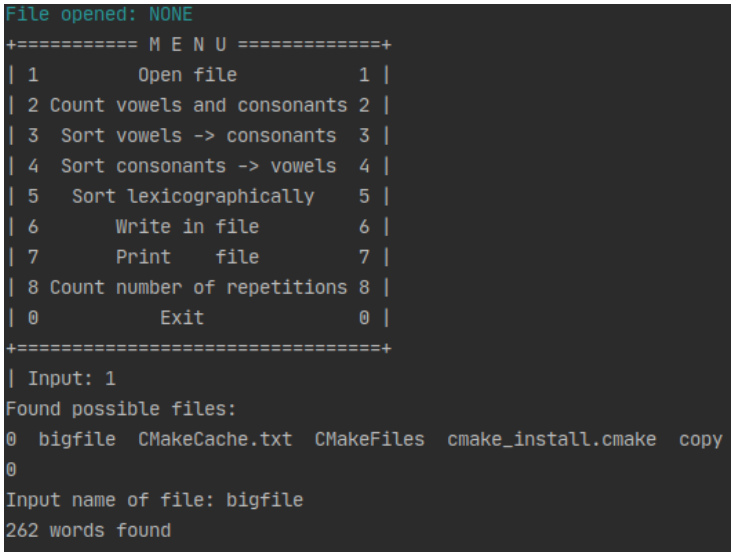

Рис. 6.3 – Открытие файла

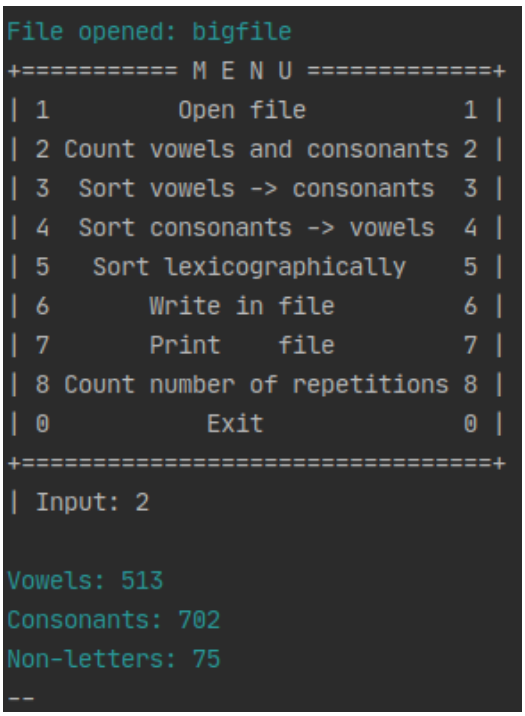

Рис. 6.4 – Подсчёт количества гласных и согласных букв

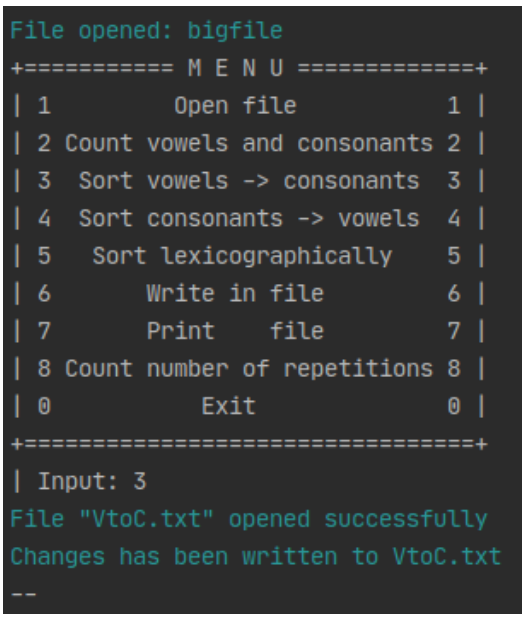

Рис. 6.5 – Сортировка от гласных к согласным

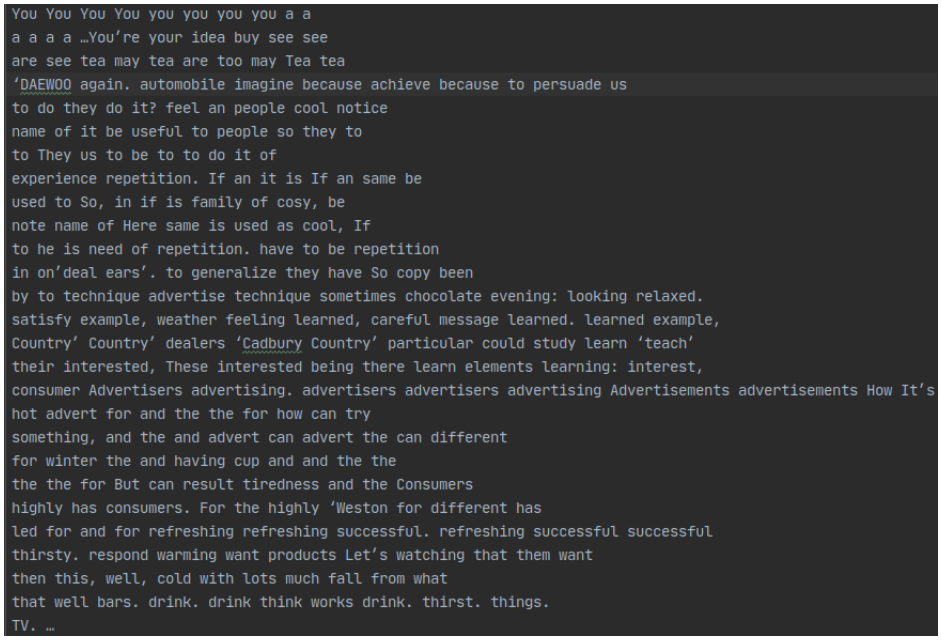

Рис. 6.6 – Содержимое файла VtoC.txt

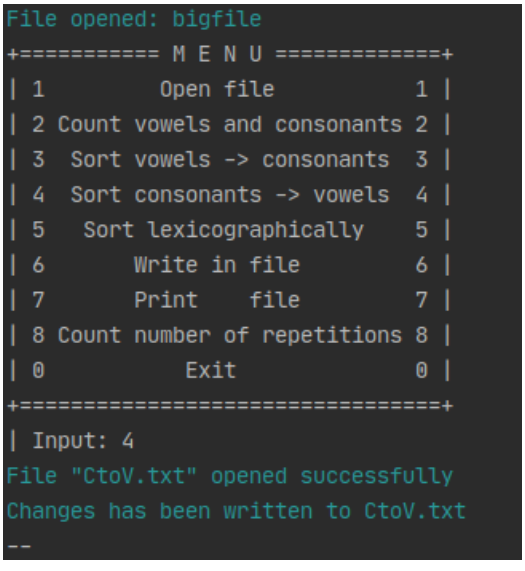

Рис. 6.7 – Сортировка от согласных к гласным

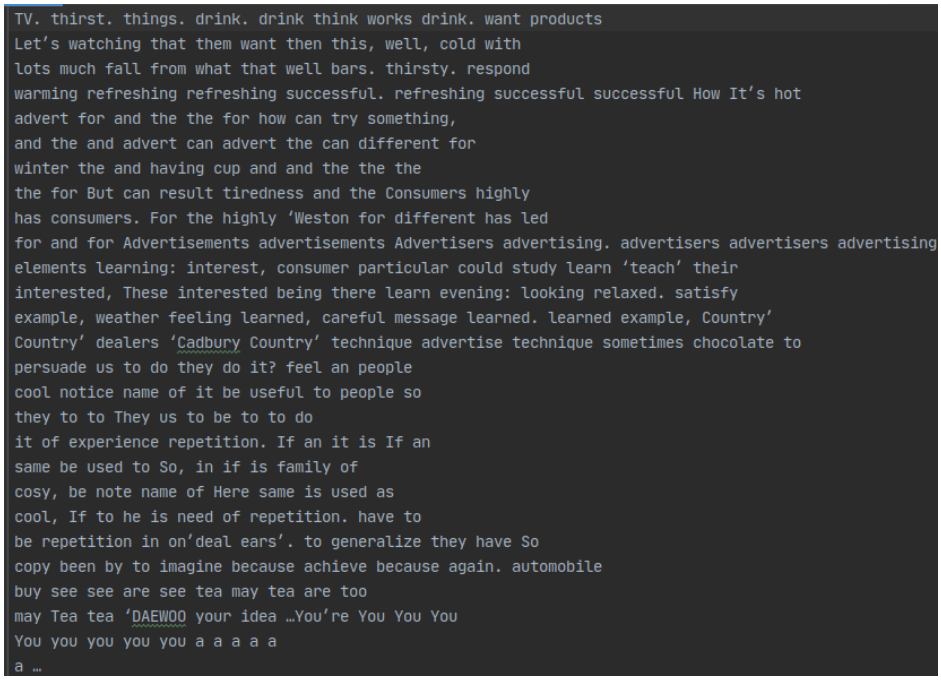

Рис. 6.8 – Содержимое файла CtoV.txt

Выберем 5 пункт меню. Ожидается, что на экран выведется сообщение о том, что изменённое содержимое файла записано в новый файл lex.txt. Результат на рисунке 6.9. Содержимое файла lex.txt на рисунке 6.10.

Пункт 6 предлагает пользователю ввести имя файла, в который он хочет записать изменения.

Выберем 6 пункт меню и попытаемтся открыть существующий файл "0". Ожидается, что на экран выведется сообщение о том, что такой файл уже существует. Отказ от изменений представлен на рисунке 6.11. Согласие на рисунке 6.12. Содержимое файла 0 рисунке 6.13.

Так как файл слишком большой, он не умещается на экран. Для демонстрации работы пунктов 7-8 откроем другой файл с помощью пункта меню 1 (рисунок 6.14).

Выберем 7 пункт меню. Ожидается, что на экран выведется содержимое открытого файла. Результат на рисунке 6.15

Выберем 7 пункт меню. Ожидается, что на экран выведется список уникальных слов, количество их повторений и сообщение о том, что все слова повторяются единожды. Результат на рисунке 6.16.

Выберем 7 пункт меню при файле, где разные слова имеют разное количество повторений. Ожидается, что на экран выведется список

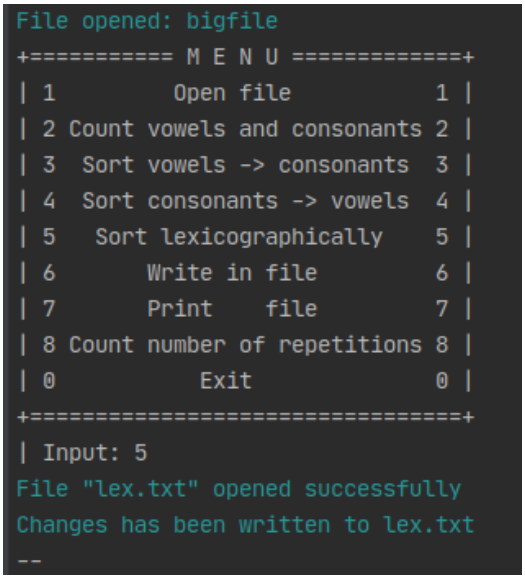

Рис. 6.9 – Сортировка по алфавиту

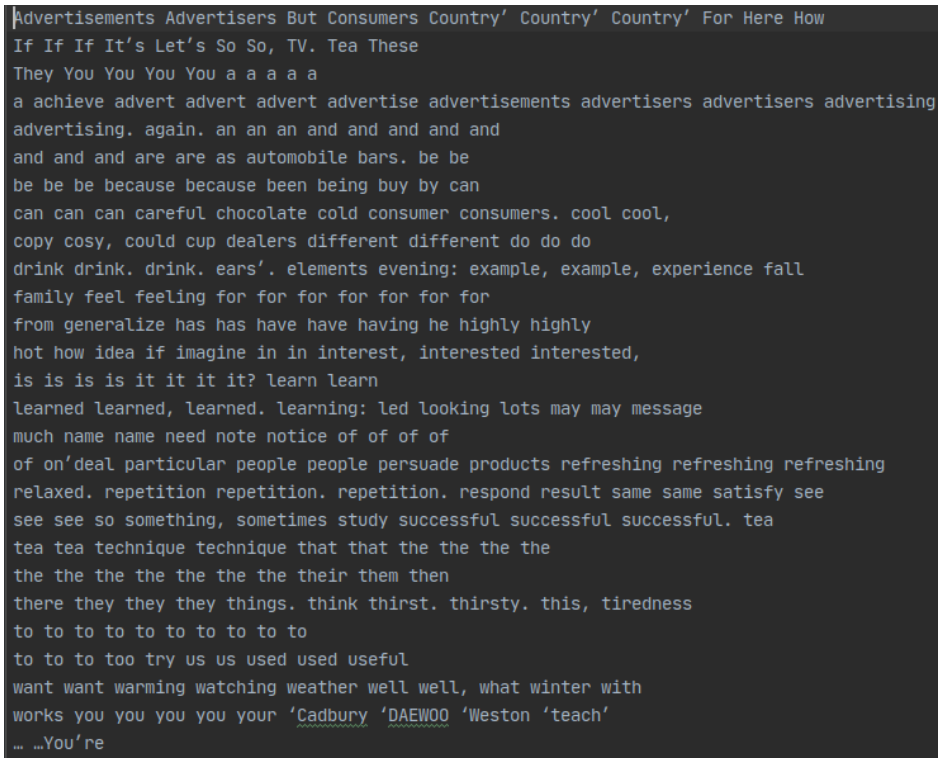

Рис. 6.10 – Содержимое файла lex.txt

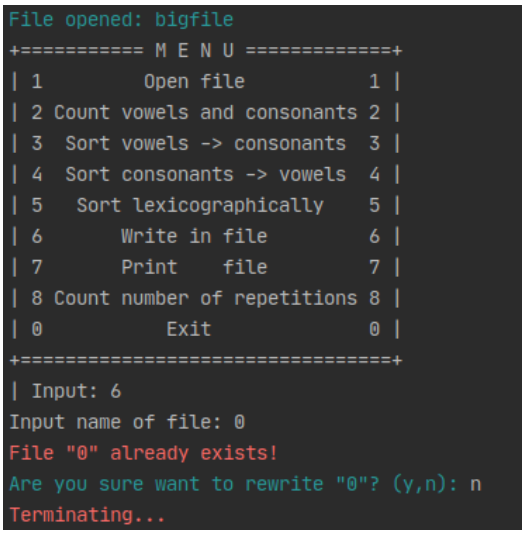

Рис. 6.11 – Отказ от перезаписи существующего файла

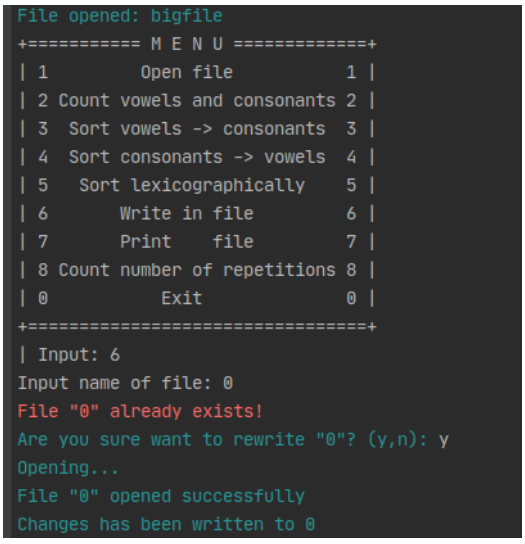

Рис. 6.12 – Согласие на перезапись существующего файла

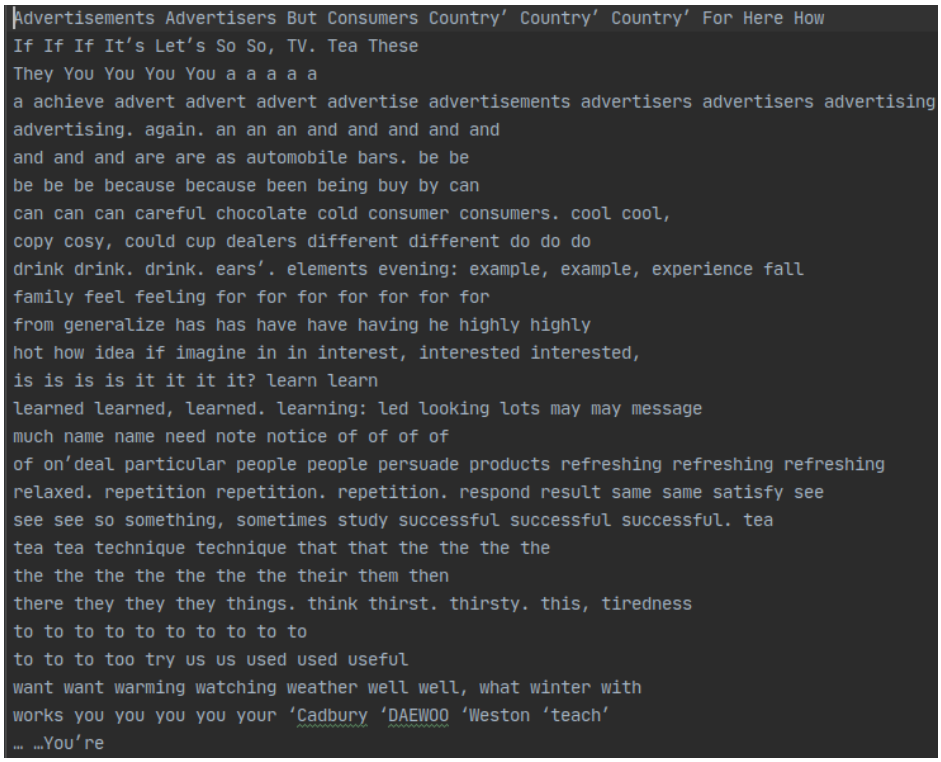

Рис. 6.13 – Содержимое файла 0

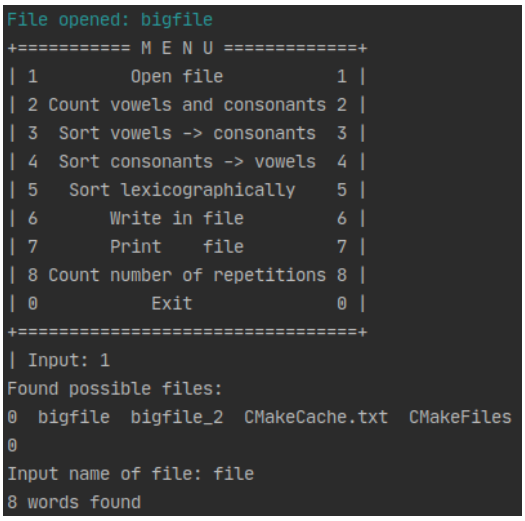

Рис. 6.14 – Открытие файла при уже открытом другом файле

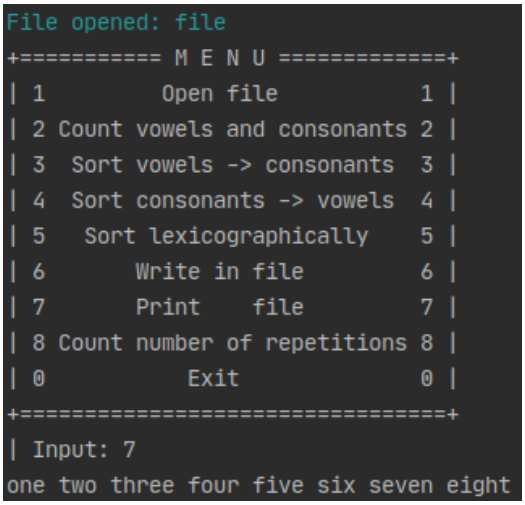

Рис. 6.15 – Вывод содержимого файла на экран

уникальных слов, количество их повторений и сообщение о самом часто встречающемся слове. Результат на рисунках 6.17 и 6.18.

Выберем 0 пункт меню чтобы завершить программу. Результат на рисунке 6.19.

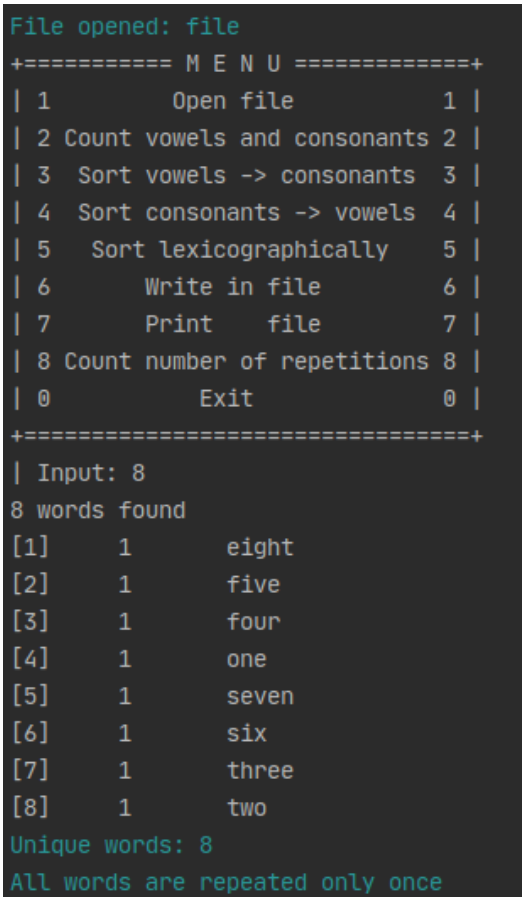

Рис. 6.16 – Вывод количества повторений слов. Все слова повторяются единожды

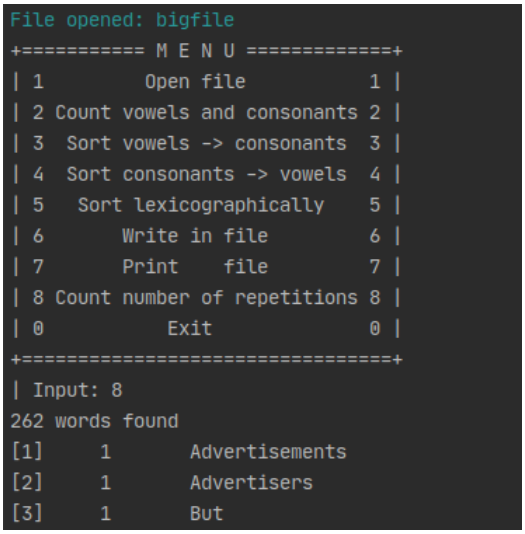

Рис. 6.17 – Вывод количества повторений слов (начало)

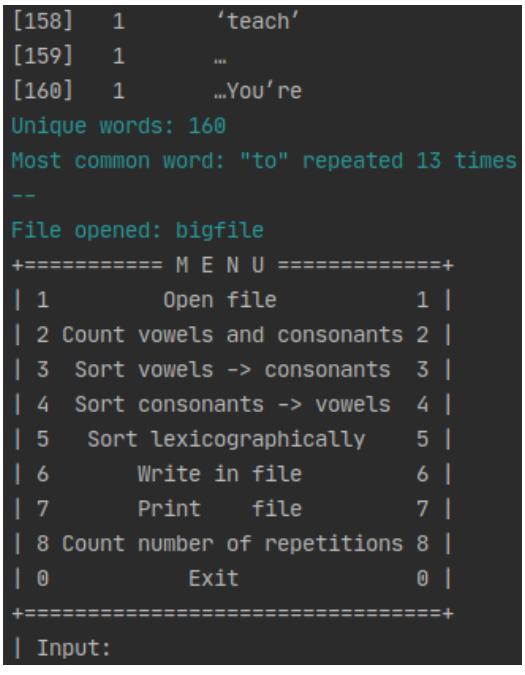

Рис. 6.18 – Вывод количества повторений слов (конец)

Рис. 6.19 – Выход из программы

Заключение

В результате выполнения данной курсовой работы мы закрепили и применили на практике знания, полученные в ходе выполнения лабораторных работ. Реализованная программа соответствует поставленным задачам и безошибочно выполняет свою работу. Функционал программы позволяет не только выполнять вычисления, но и реализует полноценное взаимодействие с пользователем, корректно обрабатывать его запросы и выдавать ему ожидаемый результат. Выполнение данной лабораторной работы позволило углубить наши знания в технологии программирования типовых задач обработки текстовых данных и принципах программной реализации взаимодействия с файлами.

Список использованных сточников

[1] Заголовочный файл стандартной библиотеки <string> — cppreference.com. (n.d.). Retrieved May 15, 2020, from https://ru.cppreference.com/w/cpp/header/string

[2] std::basic_fstream — cppreference.com. (n.d.). Retrieved May 15, 2020, from https://ru.cppreference.com/w/cpp/io/basic_fstream

[3] Лексикографический порядок — Википедия. (n.d.). Retrieved May 15, 2020, from https://ru.wikipedia.org/wiki/Лексикографический_порядок

[4] std::sort — cppreference.com. (n.d.). Retrieved May 15, 2020, from https://ru.cppreference.com/w/cpp/algorithm/sort

Приложение А
Листинг программного кода
main.cpp

```
#include <algorithm>
#include "openFile.h"
#include "countVC.h"
#include "writeFile.h"
#include "countWords.h"
#include "Colors.h"
#include "sortVC.h"
#include "printFile.h"

int main()
{
    char choice = '1';
    std::string *words{nullptr}, nameOfFile = "";
    int wordsSize = -1;
    int V{-1}, C{-1}, N{-1};

    while(choice != '0')
    {
        std::cout << C_BLUE << "File opened: ";
        if(nameOfFile.empty()) std::cout << "NONE";
        else                   std::cout << nameOfFile;
        std::cout << C_NONE << std::endl;

        std::cout << "+========== M E N U ==========+"
           << std::endl;
        std::cout << "| 1         Open file          1 |"
           << std::endl;
        std::cout << "| 2 Count vowels and consonants 2 |"
           << std::endl;
        std::cout << "| 3  Sort vowels -> consonants  3 |"
           << std::endl;
        std::cout << "| 4  Sort consonants -> vowels  4 |"
           << std::endl;
        std::cout << "| 5   Sort lexicographically    5 |"
           << std::endl;
        std::cout << "| 6       Write in file         6 |"
           << std::endl;
        std::cout << "| 7       Print    file         7 |"
```

```
        << std::endl;
32      std::cout << "| 8 Count number of repetitions 8 |"
        << std::endl;
33      std::cout << "| 0            Exit               0 |"
        << std::endl;
34      std::cout << "+=================================+"
        << std::endl;
35      std::cout << "| Input: "; std::cin >> choice;
36      std::cin.clear(); std::cin.ignore(std::
        numeric_limits<std::streamsize>::max(), '\n');
37
38      switch(choice)
39      {
40          case '1': //open file
41              delete [] words;
42              V = -1; C = -1; N = -1; nameOfFile = "";
43              std::cout << "Found possible files:" << std
                    ::endl << system("ls") << '\r' << std::
                    endl;
44              words = openFile(wordsSize,nameOfFile);
45              std::cout << std::endl;
46              break;
47          case '2': //count vowels & consonants
48          {
49              if(wordsSize == -1)
50              {
51                  std::cout << C_RED << "No file has been
                        opened" << C_NONE << std::endl;
52                  break;
53              }
54              if(V==-1)
55              {
56                  countVC(words, wordsSize, V, C, N);
57              }
58
59              std::cout << C_BLUE << std::endl;
60              std::cout << "Vowels: " << V << std::endl;
61              std::cout << "Consonants: " << C << std::
                    endl;
62              std::cout << "Non-letters: " << N << std::
                    endl;
63              std::cout << C_NONE;
64              //writeFile(words,wordsSize,choice);
```

```
65                  std::cout << "--" << std::endl;
66              }
67                  break;
68              case '3': //sort V->C
69              {
70                  if(wordsSize == -1)
71                  {
72                      std::cout << C_RED << "No file has been
                            opened" << C_NONE << std::endl;
73                      break;
74                  }
75                  else if(wordsSize == 0)
76                  {
77                      std::cout << C_BLUE << "Looks like this
                            file is empty" << C_NONE << std::
                           endl;
78                      break;
79                  }
80                  sortVC(words,wordsSize,choice);
81                  std::cout << "--" << std::endl;
82              }
83              break;
84              case '4': //sort C->V
85              {
86                  if(wordsSize == -1)
87                  {
88                      std::cout << C_RED << "No file has been
                            opened" << C_NONE << std::endl;
89                      break;
90                  }
91                  else if(wordsSize == 0)
92                  {
93                      std::cout << C_BLUE << "Looks like this
                            file is empty" << C_NONE << std::
                           endl;
94                      break;
95                  }
96                  sortVC(words,wordsSize,choice);
97                  std::cout << "--" << std::endl;
98              }
99              break;
100             case '5': //sort lexicographically
101             {
```

```
102                 if(words)
103                 {
104                 std::sort(words,words+wordsSize);
105                 writeFile(words,wordsSize,choice);
106                 std::cout << "---" << std::endl;
107                 }
108                 else
109                 std::cout << C_RED << "No data to sort" <<
                        C_NONE << std::endl << C_BLUE
110                 << "Use \"Open file\" to use this option"
                        << C_NONE << std::endl;
111             }
112                 break;
113             case '6': //write in file
114             {
115                 if(words)
116                 writeFile(words,wordsSize);
117                 else
118                 std::cout << C_RED << "No data to write" <<
                         C_NONE << std::endl;
119             }
120                 break;
121             case '7': // print file
122             {
123                 printFile(words,wordsSize);
124             }
125             break;
126             case '8': //count words
127             {
128                 if(!countWords(wordsSize,nameOfFile)) break
                        ;
129                 //else writeFile(words,wordsSize,choice);
130             }
131             break;
132             case '0':
133                 std::cout << C_BLUE << "Finishing..." <<
                        C_NONE << std::endl;
134                 delete [] words;
135                 break;
136             default:
137                 std::cout << C_RED << "Wrong input" <<
                        C_NONE << std::endl;
138                 break;
```

```
139             }
140         }
141         return 0;
142 }
```

openFile.cpp

```
1  //
2  // Created by kirill on 29.02.2020.
3  //
4
5  #include "openFile.h"
6
7  std::string* openFile(int &wordsSize, std::string &name)
8  {
9      std::string *words;
10     std::string nameOfFile, tmpstr;
11
12     if(name.empty())
13     {
14         std::cout << "Input name of file: ";
15         std::cin >> nameOfFile;
16         std::cin.clear();
17         std::cin.ignore(std::numeric_limits<std::streamsize
               >::max(),'\n');
18     }
19     else
20     {
21         nameOfFile = name;
22     }
23
24     std::ifstream fin;
25     fin.open(nameOfFile);
26
27     if (fin.is_open()) {
28         if(nameOfFile.empty())std::cout << C_BLUE << "File
               \"" << C_RED << nameOfFile << C_BLUE << "\"
               opened successfully" << C_NONE << std::endl;
29         name = nameOfFile;
30     }
31     else {
32         std::cout << C_RED << "Error: could not open file
               \"" << nameOfFile << "\"" << C_NONE << std::endl
               ;
33         wordsSize = -1;
34         return nullptr;
35     }
36     wordsSize = 0;
37
```

```
38
39      for (fin >> tmpstr; !fin.eof(); fin >> tmpstr) {
40          ++wordsSize;
41      }
42      if(!nameOfFile.empty())std::cout << wordsSize << "
            words found" << std::endl;
43
44      fin.close();
45      fin.open(nameOfFile);
46
47      words = new std::string[wordsSize];
48
49      for (int i = 0; i < wordsSize; ++i) {
50          fin >> tmpstr;
51          words[i] = tmpstr;
52          //std::cout << words[i] << std::endl;
53      }
54
55      fin.close();
56
57      return words;
58  }
```

openFile.h

```
1  //
2  // Created by kirill on 29.02.2020.
3  //
4
5  #ifndef KR_OPENFILE_H
6  #define KR_OPENFILE_H
7
8  #include <iostream>
9  #include <string>
10 #include <fstream>
11 #include "Colors.h"
12
13 std::string* openFile(int &wordsSize, std::string &name);
14
15 #endif //KR_OPENFILE_H
```

countVC.cpp

```cpp
//
// Created by kirill on 29.02.2020.
//

#include <iostream>
#include "countVC.h"

void countVC(const std::string *words, const int wordsSize,
    int &V, int &C, int &N)
{
    V = 0; C = 0; N = 0;
    for(int i = 0; i < wordsSize; ++i)
    {
        for(auto c : words[i])
        {
            if(c == 'a' || c == 'e' || c == 'i' || c == 'y'
                ||
               c == 'o' || c == 'u' || c == 'A' || c == 'Y'
                   ||
               c == 'E' || c == 'I' || c == 'O' || c == 'U'
                  )
                ++V;
            else if(isalpha(c))
                ++C;
            else
                ++N;
        }
    }
}
```

countVC.h

```
1  //
2  // Created by kirill on 29.02.2020.
3  //
4
5  #ifndef KR_COUNTVC_H
6  #define KR_COUNTVC_H
7
8  #include <string>
9
10 void countVC(const std::string *words, int wordsSize, int &
       V, int &C, int &N);
11
12 #endif //KR_COUNTVC_H
```

writeFile.cpp

```
//
// Created by kirill on 29.02.2020.
//

#include "writeFile.h"

void writeFile(std::string *words, unsigned int wordsSize,
    char opt)
{
    std::ofstream fout;
    std::string nameOfFile;

    switch(opt)
    {
        case '2':
        {
            nameOfFile = "count.txt";
        }
        break;
        case '3':
        {
            nameOfFile = "VtoC.txt";
        }
        break;
        case '4':
        {
            nameOfFile = "CtoV.txt";
        }
        break;
        case '5':
        {
            nameOfFile = "lex.txt";
        }
        break;
        case '6':
        {
            std::cout << "Input name of file: ";
            //input(nameOfFile, "Input name of file:");
            std::cin >> nameOfFile;
            std::cin.clear();
            std::cin.ignore(std::numeric_limits<std::
                streamsize>::max(),'\n');
```

```
41          }
42          break;
43          case '8':
44          {
45              nameOfFile = "NumOfRep.txt";
46          }
47          break;
48          default:
49          {
50              std::cout << C_RED << "Error: Could not detect
                    option number" << std::endl <<
51                          "Terminating..." << C_NONE << std::
                            endl;
52          }
53          return;
54      }
55
56      if(opt == '6')
57      {
58          std::fstream test(nameOfFile, std::ios::in);
59          if (test.is_open()) {
60              std::cout << C_RED << "File \"" << nameOfFile
                    << "\" already exists!" << C_NONE << std::
                    endl;
61              std::cout << C_BLUE << "Are you sure want to
                    rewrite \"" << nameOfFile << "\"? (y,n): "
                    << C_NONE;
62              char ch;
63              std::cin >> ch;
64              std::cin.clear();
65              std::cin.ignore(std::numeric_limits<std::
                    streamsize>::max(), '\n');
66              if (ch != 'y') {
67                  std::cout << C_RED << "Terminating..." <<
                        C_NONE << std::endl;
68                  test.close();
69                  return;
70              } else
71                  std::cout << C_BLUE << "Opening..." <<
                        C_NONE << std::endl;
72          }
73
74          test.close();
```

```
75      }
76
77      fout.open(nameOfFile);
78
79      if (fout.is_open())
80          std::cout << C_BLUE << "File \"" << nameOfFile << "
                \" opened successfully" << C_NONE << std::endl;
81      else {
82          std::cout << C_RED << "Error: could not open file
                \"" << nameOfFile << "\"" << C_NONE << std::endl
                ;
83          return;
84      }
85
86      for(int i = 0; i < wordsSize; ++i)
87      {
88          fout << words[i] << " ";
89          if((i+1)%10 == 0)
90          fout << std::endl;
91      }
92
93      std::cout << C_BLUE << "Changes has been written to "
           << nameOfFile << C_NONE << std::endl;
94
95      fout.close();
96
97  }
```

writeFile.h

```
1  //
2  // Created by kirill on 29.02.2020.
3  //
4
5  #ifndef KR_WRITEFILE_H
6  #define KR_WRITEFILE_H
7
8  #include <fstream>
9  #include <iostream>
10 #include "Colors.h"
11
12 void writeFile(std::string *words, unsigned int wordsSize,
       char opt = '6');
13
14 #endif //KR_WRITEFILE_H
```

countWords.cpp

```
1  //
2  // Created by kirill on 04.03.2020.
3  //
4
5  #include "countWords.h"
6
7  bool countWords(int wordsSize, std::string nameOfFile)
8  {
9      if (wordsSize == -1)
10     {
11         std::cout << C_RED << "No file has been opened" <<
               C_NONE;
12         std::cout << std::endl;
13         return false;
14     }
15     else if (wordsSize == 0)
16     {
17         std::cout << C_BLUE << "Looks like this file is
               empty" << C_NONE << std::endl;
18         return false;
19     }
20
21     std::string *wordstmp = openFile(wordsSize, nameOfFile)
           ;
22     std::sort(wordstmp, wordstmp + wordsSize);
23     int uniqueWords = 1, counter = 1;
24     std::string mostCommonWord{"NONE"};
25     int max = 1;
26
27     for (int i = 0; i < wordsSize;)
28     {
29         std::cout << '[' << uniqueWords << "] ";
30         if (i == wordsSize - 1)
31         {
32             std::cout << '\t' << counter << '\t' <<
                   wordstmp[i] << std::endl;
33             break;
34         }
35         for (int j = i + 1; j < wordsSize; ++j)
36         {
37             if (wordstmp[i] != wordstmp[j])
38             {
```

```
39                  std::cout << '\t' << counter << '\t' <<
                        wordstmp[i] << std::endl;
40                  if (counter > max)
41                  {
42                      max = counter;
43                      mostCommonWord = wordstmp[i];
44                  }
45                  counter = 1;
46                  i = j;
47                  ++uniqueWords;
48                  break;
49              }
50              else
51              {
52                  ++counter;
53              }
54
55              if (j == wordsSize - 1)
56              {
57                  i = wordsSize;
58                  std::cout << '\t' << counter << '\t' <<
                        wordstmp[i] << std::endl;
59              }
60          }
61      }
62
63      std::cout << C_BLUE << "Unique words: " << uniqueWords
            << std::endl;
64      if (max > 1)
65          std::cout << "Most common word: \"" <<
                mostCommonWord << "\" repeated " << max << "
                times" << std::endl;
66      else
67          std::cout << "All words are repeated only once" <<
                std::endl << C_NONE;
68
69      std::cout << "--\n";
70  }
```

countWords.h

```
//
// Created by kirill on 04.03.2020.
//

#ifndef KR_COUNTWORDS_H
#define KR_COUNTWORDS_H

#include <string>
#include <fstream>
#include <iostream>
#include <algorithm>
#include "openFile.h"
#include "Colors.h"

bool countWords(int wordsSize, std::string nameOfFile);

#endif //KR_COUNTWORDS_H
```

sortVC.cpp

```
1  //
2  // Created by kirill on 06.03.2020.
3  //
4
5  #include "sortVC.h"
6
7  void sortVC(std::string *words, const int &wordsSize, char
       &opt)
8  {
9      if(opt != '3' && opt != '4')
10     {
11         std::cout << C_RED << "Error: could not detect sort
               option" << C_NONE << std::endl;
12         return;
13     }
14
15     int Vowels[wordsSize];
16     int Consonants[wordsSize];
17     //int Non_letters[wordsSize];
18
19     for (int i = 0; i < wordsSize; ++i)
20     {
21         Vowels[i] = 0;
22         Consonants[i] = 0;
23         //Non_letters[i] = 0;
24
25         for(auto c : words[i])
26         {
27             if(c == 'a' || c == 'e' || c == 'i' || c == 'y'
                   ||
28                c == 'o' || c == 'u' || c == 'A' || c == 'Y'
                       ||
29                c == 'E' || c == 'I' || c == 'O' || c == 'U'
                       )
30                 Vowels[i] += 1;
31             else if(isalpha(c))
32                 Consonants[i] += 1;
33             //else
34                 //Non_letters[i] += 1;
35         }
36         /*
37         if(Vowels[i] == 0) std::cout << "\e[31m";
```

```
38          std::cout << "{tmp}" << words[i] << " \t\tVowels:"
               << Vowels[i] << std::endl;
39          std::cout << "\e[0m";
40           */
41
42          double k1, k2;
43
44          k1 = calck(Vowels[i],Consonants[i],opt);
45
46          if(i>0)
47          {
48              for(int j = 0; j < i; ++j)
49              {
50                  k2 = calck(Vowels[j],Consonants[j],opt);
51
52                  if(k2 < k1)
53                  {
54                      int vtmp = Vowels[i];
55                      Vowels[i] = Vowels[j];
56                      Vowels[j] = vtmp;
57
58                      int ctmp = Consonants[i];
59                      Consonants[i] = Consonants[j];
60                      Consonants[j] = ctmp;
61
62                      std::string stmp = words[i];
63                      words[i] = words[j];
64                      words[j] = stmp;
65                  }
66              }
67          }
68      }
69      /*
70      for(int i = 0; i < wordsSize; ++i)
71      {
72          double k3 = calck(Vowels[i],Consonants[i],opt);
73          std::cout << k3 << " ";
74          if((i+1)%15==0) std::cout << std::endl;
75      }
76      std::cout << std::endl;
77      */
78
79      writeFile(words,wordsSize,opt);
```

```
80| }
```

sortVC.h

```
1  //
2  // Created by kirill on 06.03.2020.
3  //
4
5  #ifndef KR_SORTVC_H
6  #define KR_SORTVC_H
7
8  #include <string>
9  #include <iostream>
10 #include "Colors.h"
11 #include "writeFile.h"
12 #include "calck.h"
13
14 void sortVC(std::string *words, const int &wordsSize, char
       &opt);
15
16 #endif //KR_SORTVC_H
```

printFile.cpp

```
1  //
2  // Created by kirill on 06.03.2020.
3  //
4
5  #include "printFile.h"
6
7  void printFile(const std::string *words, const int &
       wordsSize)
8  {
9      if (wordsSize != -1)
10     {
11         for (int i = 0; i < wordsSize; ++i)
12         {
13             std::cout << words[i] << " ";
14             if ((i + 1) % 10 == 0)
15                 std::cout << std::endl;
16         }
17     }
18
19     if(wordsSize == 0)
20         std::cout << C_BLUE << "Looks like this file is
               empty" << C_NONE;
21     if(wordsSize == -1)
22         std::cout << C_RED << "No file has been opened" <<
               C_NONE;
23
24     std::cout << std::endl;
25 }
```

printFile.h

```
1  //
2  // Created by kirill on 06.03.2020.
3  //
4
5  #ifndef KR_PRINTFILE_H
6  #define KR_PRINTFILE_H
7
8  #include <string>
9  #include <iostream>
10 #include "Colors.h"
11
12 void printFile(const std::string *words, const int &
       wordsSize);
13
14 #endif //KR_PRINTFILE_H
```

Colors.h

```
1  //
2  // Created by kirill on 05.03.2020.
3  //
4
5  #ifndef KR_COLORS_H
6  #define KR_COLORS_H
7
8  #define C_RED "\e[31m"
9  #define C_NONE "\e[0m"
10 #define C_BLUE "\e[36m"
11
12 #endif //KR_COLORS_H
```

calck.cpp

```
//
// Created by kirill on 06.03.2020.
//

#include "calck.h"

double calck(const double &v, const double &c, const char &
    opt)
{
     double k1;

     if(c > 0 && v > 0)
     {
          if (opt == '3') k1 = v / c;
          else            k1 = c / v;
     }
     else if(c == 0 && v > 0)
     {
          if (opt == '3') k1 = v + 1000;
          else            k1 = 0;
     }
     else if(v == 0 && c > 0)
     {
          if (opt == '3') k1 = 0;
          else            k1 = c + 1000;
     }
     else k1 = -1;

     return k1;
}
```

calck.h

```
1  //
2  // Created by kirill on 06.03.2020.
3  //
4
5  #ifndef KR_CALCK_H
6  #define KR_CALCK_H
7
8  double calck(const double &v, const double &c, const char &
       opt);
9
10 #endif //KR_CALCK_H
```