

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

КУРСОВАЯ РАБОТА
по дисциплине «Программирование»
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	7
1 Внешние форматы хранения данных	8
2 Внутренние форматы хранения данных	9
3 Описание пользовательских функций и модулей программы	10
4 Описание интерфейса пользователя	11
5 Описание алгоритма работы программы	13
Функция main	13
Функция openFile	14
Функция countVC	14
Функция writeFile	14
Функция countWords	15
Функция sortVC	16
Функция printFile	16
Функция calc	17
6 Примеры работы программы	18
Заключение	29
Список использованных источников	30
Приложение А Исходный код программы	31
main.cpp	31
openFile.cpp	36
openFile.h	38
countVC.cpp	39
countVC.h	40
writeFile.cpp	41
writeFile.h	44
countWords.cpp	45
countWords.h	47
sortVC.cpp	48
sortVC.h	51
printFile.cpp	52

printFile.h	53
Colors.h	54
calck.cpp	55
calck.h	56

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.

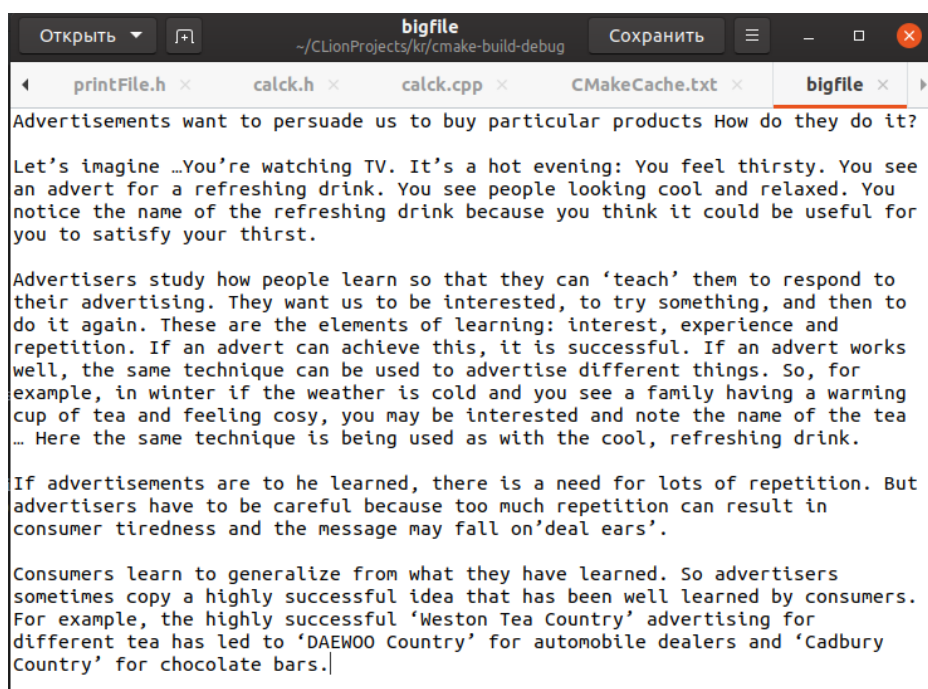


Рис. 1.1 – Входной файл bigfile

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 представлена на рисунке 2.1, где n — количество слов в исходном файле.

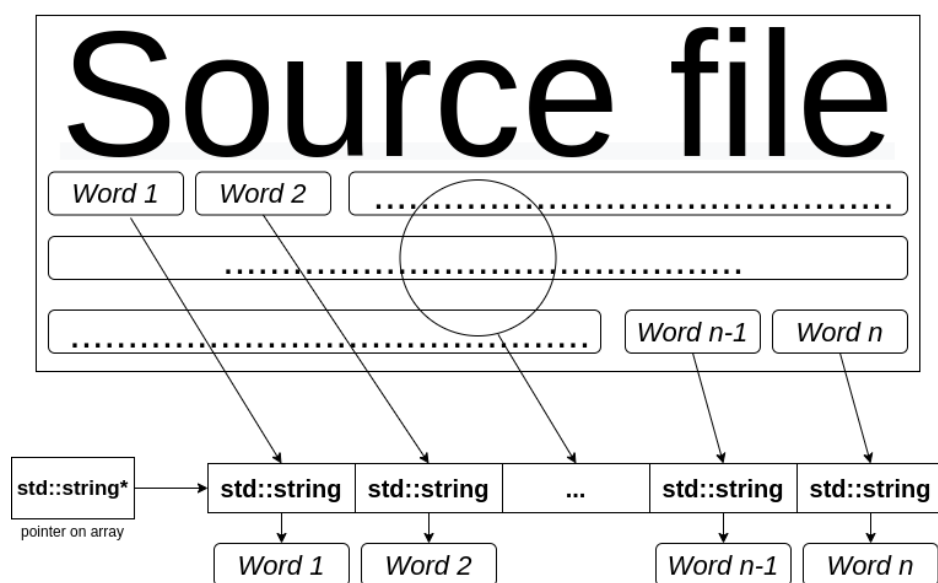


Рис. 2.1 – Формат хранения данных, полученных из файла

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, представлены 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 предоставляется меню из девяти пунктов (рисунок 4.1).

```
File opened: NONE
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file  7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input:
```

Рис. 4.1 – Главное меню

Вверху меню отображается имя текущего файла. По умолчанию при открытии 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 пунктов 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] [количество 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 некорректном вводе и вернётся 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 гласных, согласных и не-букв в тексте файла;

Запускается цикл.

ПОКА выбранный 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 возвращаемое 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`. Иначе выводится сообщение об ошибке.

ЕСЛИ 6 Сначала проверяется, что массив слов не пуст. Если слова есть, запускается функция `writeFile`. Иначе 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 вводится 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 лексикографически[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 (1)

Перестроится в

1 1 2 2 3 3 4 4 4 5 7 8 8 8 8 9 (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 представлено главное меню программы.

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

```
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file  7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input: hjdfjfkjdg
Wrong input
```

Рис. 6.1 – Ошибочный ввод

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

Откроем заготовленный 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 отображаться 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.

```

+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file   7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input: 2
No file has been opened

```

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

```

File opened: NONE
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file   7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input: 1
Found possible files:
0  bigfile CMakeCache.txt CMakeFiles cmake_install.cmake copy
0
Input name of file: bigfile
262 words found

```

Рис. 6.3 – Открытие файла

```

File opened: bigfile
+===== M E N U =====+
| 1          Open file          1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants  3 |
| 4 Sort consonants -> vowels  4 |
| 5 Sort lexicographically     5 |
| 6      Write in file         6 |
| 7      Print   file          7 |
| 8 Count number of repetitions 8 |
| 0          Exit              0 |
+=====+
| Input: 2

Vowels: 513
Consonants: 702
Non-letters: 75
--

```

Рис. 6.4 – Подсчёт количества гласных и согласных букв

```

File opened: bigfile
+===== M E N U =====+
| 1          Open file          1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants  3 |
| 4 Sort consonants -> vowels  4 |
| 5 Sort lexicographically     5 |
| 6      Write in file         6 |
| 7      Print   file          7 |
| 8 Count number of repetitions 8 |
| 0          Exit              0 |
+=====+
| Input: 3
File "VtoC.txt" opened successfully
Changes has been written to VtoC.txt
--

```

Рис. 6.5 – Сортировка от гласных к согласным

```

You You You You you you you you a a
a a a a ...You're your idea buy see see
are see tea may tea are too may Tea tea
'DAEWOO again. automobile imagine because achieve because to persuade us
to do they do it? feel an people cool notice
name of it be useful to people so they to
to They us to be to to do it of
experience repetition. If an it is If an same be
used to So, in if is family of cosy, be
note name of Here same is used as cool, If
to he is need of repetition. have to be repetition
in on'deal ears'. to generalize they have So copy been
by to technique advertise technique sometimes chocolate evening: looking relaxed.
satisfy example, weather feeling learned, careful message learned. learned example,
Country' Country' dealers 'Cadbury Country' particular could study learn 'teach'
their interested, These interested being there learn elements learning: interest,
consumer Advertisers advertising. advertisers advertisers advertising Advertisements advertisements How It's
hot advert for and the the for how can try
something, and the and advert can advert the can different
for winter the and having cup and and the the
the the for But can result tiredness and the Consumers
highly has consumers. For the highly 'Weston for different has
led for and for refreshing refreshing successful. refreshing successful successful
thirsty. respond warming want products Let's watching that them want
then this, well, cold with lots much fall from what
that well bars. drink. drink think works drink. thirst. things.
TV. ...

```

Рис. 6.6 – Содержимое файла VtoC.txt

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file   7 |
| 8 Count number of repetitions 8 |
| 0          Exit      0 |
+=====+
| Input: 4
File "CtoV.txt" opened successfully
Changes has been written to CtoV.txt
--

```

Рис. 6.7 – Сортировка от согласных к гласным

```

TV. thirst. things. drink. drink think works drink. want products
Let's watching that them want then this, well, cold with
lots much fall from what that well bars. thirsty. respond
warming refreshing refreshing successful. refreshing successful successful How It's hot
advert for and the the for how can try something,
and the and advert can advert the can different for
winter the and having cup and and the the the
the for But can result tiredness and the Consumers highly
has consumers. For the highly 'Weston for different has led
for and for Advertisements advertisements Advertisers advertising. advertisers advertisers advertising
elements learning: interest, consumer particular could study learn 'teach' their
interested, These interested being there learn evening: looking relaxed. satisfy
example, weather feeling learned, careful message learned. learned example, Country'
Country' dealers 'Cadbury Country' technique advertise technique sometimes chocolate to
persuade us to do they do it? feel an people
cool notice name of it be useful to people so
they to to They us to be to to do
it of experience repetition. If an it is If an
same be used to So, in if is family of
cosy, be note name of Here same is used as
cool, If to he is need of repetition. have to
be repetition in on'deal ears'. to generalize they have So
copy been by to imagine because achieve because again. automobile
buy see see are see tea may tea are too
may Tea tea 'DAEWOO your idea ..You're You You You
You you you you you a a a a
a ...

```

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 попытаемся 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 уместается 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

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file   7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input: 5
File "lex.txt" opened successfully
Changes has been written to lex.txt
--

```

Рис. 6.9 – Сортировка по алфавиту

```

Advertisements Advertisers But Consumers Country' Country' Country' For Here How
If If If It's Let's So So, TV. Tea These
They You You You You a a a a a
a achieve advert advert advert advertise advertisements advertisers advertisers advertising
advertising. again. an an an and and and and and
and and and are are as automobile bars. be be
be be be because because been being buy by can
can can can careful chocolate cold consumer consumers. cool cool,
copy cosy, could cup dealers different different do do do
drink drink. drink. ears'. elements evening: example, example, experience fall
family feel feeling for for for for for for for
from generalize has has have have having he highly highly
hot how idea if imagine in in interest, interested interested,
is is is is it it it it? learn learn
learned learned, learned. learning: led looking lots may may message
much name name need note notice of of of of
of on'deal particular people people persuade products refreshing refreshing refreshing
relaxed. repetition repetition. repetition. respond result same same satisfy see
see see so something, sometimes study successful successful successful. tea
tea tea technique technique that that the the the the
the the the the the the the their them then
there they they they things. think thirst. thirsty. this, tiredness
to to to to to to to to to to
to to to too try us us used used useful
want want warming watching weather well well, what winter with
works you you you you your 'Cadbury 'DAEWOO 'Weston 'teach'
... ..You're

```

Рис. 6.10 – Содержимое файла lex.txt

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file          1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file      6 |
| 7      Print   file      7 |
| 8 Count number of repetitions 8 |
| 0      Exit              0 |
+=====+
| Input: 6
Input name of file: 0
File "0" already exists!
Are you sure want to rewrite "0"? (y,n): n
Terminating...

```

Рис. 6.11 – Отказ от перезаписи существующего файла

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file          1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file      6 |
| 7      Print   file      7 |
| 8 Count number of repetitions 8 |
| 0      Exit              0 |
+=====+
| Input: 6
Input name of file: 0
File "0" already exists!
Are you sure want to rewrite "0"? (y,n): y
Opening...
File "0" opened successfully
Changes has been written to 0

```

Рис. 6.12 – Согласие на перезапись существующего файла


```

Advertisements Advertisers But Consumers Country' Country' Country' For Here How
If If If It's Let's So So, TV. Tea These
They You You You You a a a a
a achieve advert advert advert advertise advertisements advertisers advertisers advertising
advertising. again. an an an and and and and
and and and are are as automobile bars. be be
be be be because because been being buy by can
can can can careful chocolate cold consumer consumers. cool cool,
copy cosy, could cup dealers different different do do do
drink drink. drink. ears'. elements evening: example, example, experience fall
family feel feeling for for for for for for
from generalize has has have have having he highly highly
hot how idea if imagine in in interest, interested interested,
is is is is it it it it? learn learn
learned learned, learned. learning: led looking lots may may message
much name name need note notice of of of of
of on'deal particular people people persuade products refreshing refreshing refreshing
relaxed. repetition repetition. repetition. respond result same same satisfy see
see see so something, sometimes study successful successful successful. tea
tea tea technique technique that that the the the
the the the the the the the their them then
there they they they things. think thirst. thirsty. this, tiredness
to to to to to to to to to
to to to too try us us used used useful
want want warming watching weather well well, what winter with
works you you you you your 'Cadbury 'DAEWOO 'Weston 'teach'
... ..You're

```

Рис. 6.13 – Содержимое файла 0

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file  6 |
| 7      Print   file   7 |
| 8 Count number of repetitions 8 |
| 0      Exit          0 |
+=====+
| Input: 1
Found possible files:
0 bigfile bigfile_2 CMakeCache.txt CMakeFiles
0
Input name of file: file
8 words found

```

Рис. 6.14 – Открытие файла при уже открытом другом файле

```

File opened: file
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6 Write in file      6 |
| 7 Print file        7 |
| 8 Count number of repetitions 8 |
| 0 Exit              0 |
+=====+
| Input: 7
one two three four five six seven eight

```

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.

```

File opened: file
+===== M E N U =====+
| 1      Open file          1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5 Sort lexicographically 5 |
| 6      Write in file      6 |
| 7      Print   file       7 |
| 8 Count number of repetitions 8 |
| 0          Exit          0 |
+=====+
| Input: 8
8 words found
[1]    1      eight
[2]    1      five
[3]    1      four
[4]    1      one
[5]    1     seven
[6]    1      six
[7]    1     three
[8]    1      two
Unique words: 8
All words are repeated only once

```

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

```

File opened: bigfile
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5  Sort lexicographically  5 |
| 6      Write in file    6 |
| 7      Print   file    7 |
| 8 Count number of repetitions 8 |
| 0          Exit        0 |
+=====+
| Input: 8
262 words found
[1]  1      Advertisements
[2]  1      Advertisers
[3]  1      But

```

Рис. 6.17 – Вывод количества повторений слов (начало)

```

[158]  1      'teach'
[159]  1      ...
[160]  1      ...You're
Unique words: 160
Most common word: "to" repeated 13 times
--
File opened: bigfile
+===== M E N U =====+
| 1      Open file      1 |
| 2 Count vowels and consonants 2 |
| 3 Sort vowels -> consonants 3 |
| 4 Sort consonants -> vowels 4 |
| 5  Sort lexicographically  5 |
| 6      Write in file    6 |
| 7      Print   file    7 |
| 8 Count number of repetitions 8 |
| 0          Exit        0 |
+=====+
| Input:

```

Рис. 6.18 – Вывод количества повторений слов (конец)

```
File opened: bigfile
===== M E N U =====
1 Open file 1 |
2 Count vowels and consonants 2 |
3 Sort vowels -> consonants 3 |
4 Sort consonants -> vowels 4 |
5 Sort lexicographically 5 |
6 Write in file 6 |
7 Print file 7 |
8 Count number of repetitions 8 |
0 Exit 0 |
=====
Input: 0
Finishing...
kirill@kirill-vivobook-asusLaptop-x512da-x512da:~/CLionProjects/kr/cmake-build-debug$
```

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 источников

- [1] Заголовочный файл стандартной библиотеки `<string>` — [cppreference.com](https://ru.cppreference.com/w/cpp/header/string). (n.d.). Retrieved May 15, 2020, from <https://ru.cppreference.com/w/cpp/header/string>
- [2] `std::basic_fstream` — [cppreference.com](https://ru.cppreference.com/w/cpp/io/basic_fstream). (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](https://ru.cppreference.com/w/cpp/algorithm/sort). (n.d.). Retrieved May 15, 2020, from <https://ru.cppreference.com/w/cpp/algorithm/sort>

Приложение А

Листинг программного кода

main.cpp

```

1 #include <algorithm>
2 #include "openFile.h"
3 #include "countVC.h"
4 #include "writeFile.h"
5 #include "countWords.h"
6 #include "Colors.h"
7 #include "sortVC.h"
8 #include "printFile.h"
9
10 int main()
11 {
12     char choice = '1';
13     std::string *words{nullptr}, nameOfFile = "";
14     int wordsSize = -1;
15     int V{-1}, C{-1}, N{-1};
16
17     while(choice != '0')
18     {
19         std::cout << C_BLUE << "File opened: ";
20         if(nameOfFile.empty()) std::cout << "NONE";
21         else std::cout << nameOfFile;
22         std::cout << C_NONE << std::endl;
23
24         std::cout << "+===== M E N U =====+"
25             << std::endl;
26         std::cout << "| 1          Open file          1 |"
27             << std::endl;
28         std::cout << "| 2 Count vowels and consonants 2 |"
29             << std::endl;
30         std::cout << "| 3 Sort vowels -> consonants 3 |"
31             << std::endl;
32         std::cout << "| 4 Sort consonants -> vowels 4 |"
33             << std::endl;
34         std::cout << "| 5 Sort lexicographically 5 |"
35             << std::endl;
36         std::cout << "| 6 Write in file 6 |"
37             << std::endl;
38         std::cout << "| 7 Print file 7 |"

```

```

32         << std::endl;
std::cout << "| 8 Count number of repetitions 8 |"
33         << std::endl;
std::cout << "| 0                               Exit                               0 |"
34         << std::endl;
std::cout << "=====+"
35         << std::endl;
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
73                 opened" << C_NONE << std::endl;
74             break;
75         }
76         else if(wordsSize == 0)
77         {
78             std::cout << C_BLUE << "Looks like this
79                 file is empty" << C_NONE << std::
80                 endl;
81             break;
82         }
83         sortVC(words, wordsSize, choice);
84         std::cout << "—" << std::endl;
85     }
86     break;
87     case '4': //sort C->V
88     {
89         if(wordsSize == -1)
90         {
91             std::cout << C_RED << "No file has been
92                 opened" << C_NONE << std::endl;
93             break;
94         }
95         else if(wordsSize == 0)
96         {
97             std::cout << C_BLUE << "Looks like this
98                 file is empty" << C_NONE << std::
99                 endl;
100             break;
101         }
102         sortVC(words, wordsSize, choice);
103         std::cout << "—" << std::endl;
104     }
105     break;
106     case '5': //sort lexicographically
107     {

```

```

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" <<
110                 C_NONE << std::endl << C_BLUE
111                 << "Use \"Open file\" to use this option"
112                 << C_NONE << std::endl;
113     }
114     break;
115 case '6': //write in file
116     {
117         if(words)
118             writeFile(words, wordsSize);
119         else
120             std::cout << C_RED << "No data to write" <<
121                 C_NONE << std::endl;
122     }
123     break;
124 case '7': // print file
125     {
126         printFile(words, wordsSize);
127     }
128     break;
129 case '8': //count words
130     {
131         if(!countWords(wordsSize, nameOfFile)) break
132             ;
133         //else writeFile(words, wordsSize, choice);
134     }
135     break;
136 case '0':
137     std::cout << C_BLUE << "Finishing..." <<
138         C_NONE << std::endl;
139     delete [] words;
140     break;
141 default:
142     std::cout << C_RED << "Wrong input" <<
143         C_NONE << std::endl;
144     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>
18                        >::max(), '\n');
19    }
20    else
21    {
22        nameOfFile = name;
23    }
24
25    std::ifstream fin;
26    fin.open(nameOfFile);
27
28    if (fin.is_open()) {
29        if(nameOfFile.empty()) std::cout << C_BLUE << "File
30        \"\" << C_RED << nameOfFile << C_BLUE << "\"\
31        opened successfully\" << C_NONE << std::endl;
32        name = nameOfFile;
33    }
34    else {
35        std::cout << C_RED << "Error: could not open file
36        \"\" << nameOfFile << "\"\" << C_NONE << std::endl
37        ;
38        wordsSize = -1;
39        return nullptr;
40    }
41    wordsSize = 0;
```

```

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

```
1  //
2  //  Created by kirill on 29.02.2020.
3  //
4
5  #include <iostream>
6  #include "countVC.h"
7
8  void countVC(const std::string *words, const int wordsSize,
9              int &V, int &C, int &N)
10 {
11     V = 0; C = 0; N = 0;
12     for(int i = 0; i < wordsSize; ++i)
13     {
14         for(auto c : words[i])
15         {
16             if(c == 'a' || c == 'e' || c == 'i' || c == 'y' ||
17                c == 'o' || c == 'u' || c == 'A' || c == 'Y' ||
18                c == 'E' || c == 'I' || c == 'O' || c == 'U'
19                )
20                 ++V;
21             else if(isalpha(c))
22                 ++C;
23             else
24                 ++N;
25         }
26     }
27 }
```

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

```
1 //
2 // Created by kirill on 29.02.2020.
3 //
4
5 #include "writeFile.h"
6
7 void writeFile(std::string *words, unsigned int wordsSize,
8 char opt)
9 {
10     std::ofstream fout;
11     std::string nameOfFile;
12
13     switch(opt)
14     {
15         case '2':
16         {
17             nameOfFile = "count.txt";
18         }
19         break;
20         case '3':
21         {
22             nameOfFile = "VtoC.txt";
23         }
24         break;
25         case '4':
26         {
27             nameOfFile = "CtoV.txt";
28         }
29         break;
30         case '5':
31         {
32             nameOfFile = "lex.txt";
33         }
34         break;
35         case '6':
36         {
37             std::cout << "Input name of file: ";
38             //input(nameOfFile, "Input name of file:");
39             std::cin >> nameOfFile;
40             std::cin.clear();
41             std::cin.ignore(std::numeric_limits<std::
42                 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
51             option number" << std::endl <<
52             "Terminating..." << C_NONE << std::
53             endl;
54     }
55     return;
56 }
57
58 if(opt == '6')
59 {
60     std::fstream test(nameOfFile, std::ios::in);
61     if (test.is_open()) {
62         std::cout << C_RED << "File \"" << nameOfFile
63             << "\" already exists!" << C_NONE << std::
64             endl;
65         std::cout << C_BLUE << "Are you sure want to
66             rewrite \"" << nameOfFile << "\"? (y,n): "
67             << C_NONE;
68         char ch;
69         std::cin >> ch;
70         std::cin.clear();
71         std::cin.ignore(std::numeric_limits<std::
72             streamsize>::max(), '\n');
73         if (ch != 'y') {
74             std::cout << C_RED << "Terminating..." <<
75                 C_NONE << std::endl;
76             test.close();
77             return;
78         } else
79             std::cout << C_BLUE << "Opening..." <<
80                 C_NONE << std::endl;
81     }
82     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,
13               char opt = '6');
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" <<
12            C_NONE;
13        std::cout << std::endl;
14        return false;
15    }
16    else if (wordsSize == 0)
17    {
18        std::cout << C_BLUE << "Looks like this file is
19            empty" << C_NONE << std::endl;
20        return false;
21    }
22
23    std::string *wordstmp = openFile(wordsSize, nameOfFile)
24        ;
25    std::sort(wordstmp, wordstmp + wordsSize);
26    int uniqueWords = 1, counter = 1;
27    std::string mostCommonWord{"NONE"};
28    int max = 1;
29
30    for (int i = 0; i < wordsSize; i++)
31    {
32        std::cout << '[' << uniqueWords << " ] ";
33        if (i == wordsSize - 1)
34        {
35            std::cout << '\t' << counter << '\t' <<
36                wordstmp[i] << std::endl;
37            break;
38        }
39        for (int j = i + 1; j < wordsSize; ++j)
40        {
41            if (wordstmp[i] != wordstmp[j])
42            {
43                uniqueWords++;
44                counter = 1;
45            }
46            else
47            {
48                counter++;
49            }
50        }
51    }
52    mostCommonWord = wordstmp[0];
53    max = counter;
54    for (int i = 1; i < uniqueWords; i++)
55    {
56        if (counter > max)
57        {
58            mostCommonWord = wordstmp[i];
59            max = counter;
60        }
61    }
62    return true;
63 }
```

```

39         std::cout << '\t' << counter << '\t' <<
40         wordstmp[i] << std::endl;
41         if (counter > max)
42         {
43             max = counter;
44             mostCommonWord = wordstmp[i];
45         }
46         counter = 1;
47         i = j;
48         ++uniqueWords;
49         break;
50     }
51     else
52     {
53         ++counter;
54     }
55     if (j == wordsSize - 1)
56     {
57         i = wordsSize;
58         std::cout << '\t' << counter << '\t' <<
59         wordstmp[i] << std::endl;
60     }
61 }
62
63 std::cout << C_BLUE << "Unique words: " << uniqueWords
64 << std::endl;
65 if (max > 1)
66     std::cout << "Most common word: \" <<
67     mostCommonWord << "\" repeated \" << max << "
68     times\" << std::endl;
69 else
70     std::cout << "All words are repeated only once\" <<
71     std::endl << C_NONE;
72
73 std::cout << "--\n";
74 }

```

countWords.h

```
1 //  
2 // Created by kirill on 04.03.2020.  
3 //  
4  
5 #ifndef KR_COUNTWORDS_H  
6 #define KR_COUNTWORDS_H  
7  
8 #include <string>  
9 #include <fstream>  
10 #include <iostream>  
11 #include <algorithm>  
12 #include "openFile.h"  
13 #include "Colors.h"  
14  
15 bool countWords(int wordsSize, std::string nameOfFile);  
16  
17 #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:"
39         << Vowels[i] << std::endl;
40     std::cout << " | e[0m";
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);

```

$$80 \mid \}$$

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 &
8               wordsSize)
9 {
10     if (wordsSize != -1)
11     {
12         for (int i = 0; i < wordsSize; ++i)
13         {
14             std::cout << words[i] << " ";
15             if ((i + 1) % 10 == 0)
16                 std::cout << std::endl;
17         }
18     }
19
20     if (wordsSize == 0)
21         std::cout << C_BLUE << "Looks like this file is
22         empty" << C_NONE;
23
24     if (wordsSize == -1)
25         std::cout << C_RED << "No file has been opened" <<
26         C_NONE;
27
28     std::cout << std::endl;
29 }
```

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

```
1 //
2 // Created by kirill on 06.03.2020.
3 //
4
5 #include "calck.h"
6
7 double calck(const double &v, const double &c, const char &
  opt)
8 {
9     double k1;
10
11     if(c > 0 && v > 0)
12     {
13         if (opt == '3') k1 = v / c;
14         else k1 = c / v;
15     }
16     else if(c == 0 && v > 0)
17     {
18         if (opt == '3') k1 = v + 1000;
19         else k1 = 0;
20     }
21     else if(v == 0 && c > 0)
22     {
23         if (opt == '3') k1 = 0;
24         else k1 = c + 1000;
25     }
26     else k1 = -1;
27
28     return k1;
29 }
```

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
```