**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ(МИИТ)**

Кафедра «Вычислительные системы, сети и информационная безопасность»

**Отчет По Дисциплине**

**«Алгоритмизация и программирование»**

**Лабораторная работа №4**

*Направление:* 10.03.01*Информационная безопасность*

*Профиль:**Безопасность компьютерных систем*

Выполнил:  
студент группы УИБ-112

Орлов Андрей Витальевич

Проверил:

Старший преподаватель Никольская Марина Николаевна

(должность, ФИО)

Старший преподаватель Цыганова Наталия Алексеевна

(должность, ФИО)

Москва 2021 г.

**Задание №4**

Задание: Задан текстовый файл f1. Слова разделены пробелами и знаками препинания. Определить пять наиболее встречающихся слов и число их повторений. Использовать функции, а также меню для выбора существующего или вновь создаваемого файла.

1. Таблица имён:

|  |  |  |
| --- | --- | --- |
| Таблица имён для функции is\_the\_end | | |
| Рабочие переменные функции | | |
| fer | символьный | Переменная для проверки символа на принадлежность к буквам |

|  |  |  |
| --- | --- | --- |
| Таблица имён для функции count | | |
| Рабочие переменные функции | | |
| b | символьный | Символ для поиска в строке |
| c | строковый | Строка для поиска символа |
| counter | целочисленный | Количество символа в строке |
| f | автоматический | Переменная для перебора строки в цикле |

|  |  |  |
| --- | --- | --- |
| Таблица имён для полей класса file\_string | | |
| Рабочие переменные | | |
| main\_string | строковый | Переменная для считывания строки из файла |
| string\_result | строковый | Строчка с пятью наиболее встречающимися словами, перечисленные через пробел |
| amount\_of\_spaces | целочисленный | Количество пробелов в исходной строке |
| k | целочисленный | Переменная-количество слов в исходной строке, что берётся из файла. |
| main\_words | указатель строкового типа | Массив со всеми словами в строке |
| bin\_main\_words\_array | указатель целочисленного типа | Цифровая версия строчки со всеми словами, что есть в исходной строке |
| max\_index | указатель  целочисленного  типа | Массив с пятью наиболее часто встречающимися словами по индексу |
| condition | логический | Переменная для проверки, есть ли с чем работать программе далее, или же процесс необходимо срочно прервать (в условиях) |

|  |  |  |
| --- | --- | --- |
| Таблица имён для конструктора класса file\_string | | |
| Исходные переменные | | |
| decision | логический | Переменная для тернарного оператора – выбор режима работы программы – запись или чтение файлов. |
| f | строковый | Переменная для выбора действия программы – создание нового файла, или использование уже готового. |
| dec | строковый | Переменная, в которую вводится имя файла без расширения, который необходимо открыть. |

|  |
| --- |
| Таблица имён для метода filling\_words\_array |
| Рабочие переменные |
| Все переменные метода описаны в таблице имён полей класса file\_string |

|  |  |  |
| --- | --- | --- |
| Таблица имён для метода create\_the\_file | | |
| Исходные переменные | | |
| s | строковый | Переменная для создания имени файла обычного текстового вида (\*.txt). |
| s22 | строковый | Переменная, в которую записывается строчка, которая затем будет записана в файл для дальнейшей её обработки пользователем |
| Рабочие переменные | | |
| request | Потовая запись файлов | Переменная для записи текста, задаваемого пользователем в файл |
| c | автоматический | Переменная для последовательного перебора строки в цикле |

|  |  |  |
| --- | --- | --- |
| Таблица имён для метода create\_bin\_string | | |
| Рабочие переменные | | |
| i | целочисленный | Переменная-счётчик в цикле |
| j | целочисленный | Переменная-счётчик в цикле |

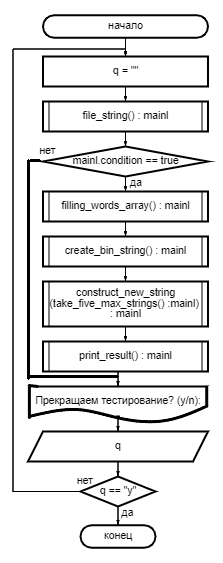
|  |  |  |
| --- | --- | --- |
| Таблица имён для метода take\_five\_max\_strings | | |
| Рабочие переменные | | |
| max | целочисленный | Переменная для поиска максимального индекса |
| y | целочисленный | Переменная для перебора индексов в массиве индексов |

|  |  |  |
| --- | --- | --- |
| Таблица имён для метода construct\_new\_string | | |
| Рабочие переменные | | |
| wowie | строковый | Переменная-индикатор исключения |
| |  |  |  | | --- | --- | --- | | Таблица имён для метода read\_the\_file | | | | Рабочие переменные | | | | helper | строковый | Переменная для потоковой считки текста из файла – обновляется при каждом считывании | | a | переменная чтения файла | Переменная потоковой передачи информации из файла, в данной программе - в строки. |   Все остальные переменные описаны в таблице имён полей класса | | | |

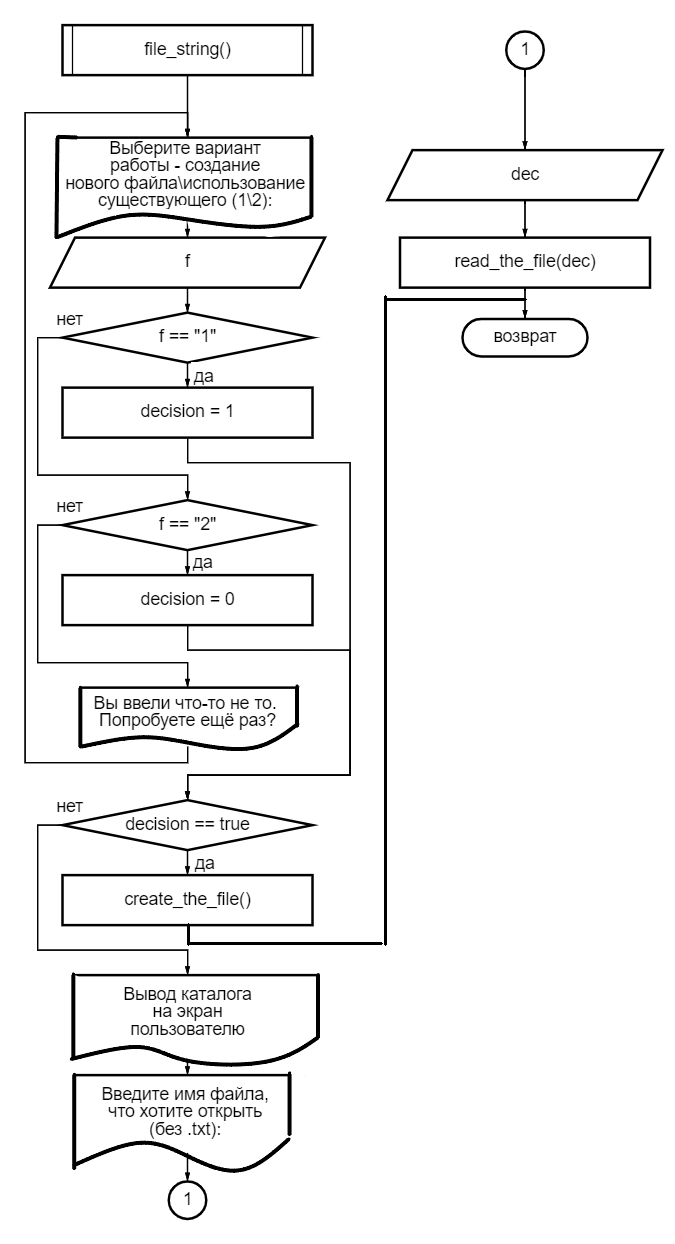
|  |  |  |
| --- | --- | --- |
| Таблица имён для метода print\_result | | |
| Рабочие переменные | | |
| f2 | переменная записи файла | Переменная, осуществляющая потоковую передачу строкового ответа в файл |

|  |  |  |
| --- | --- | --- |
| Таблица имён для основной программы | | |
| Рабочие переменные | | |
| mainl | Экземпляр класса file\_string | Объект для работы с файлами и строками |
| q | строковый | Переменная решения для тестировщика |

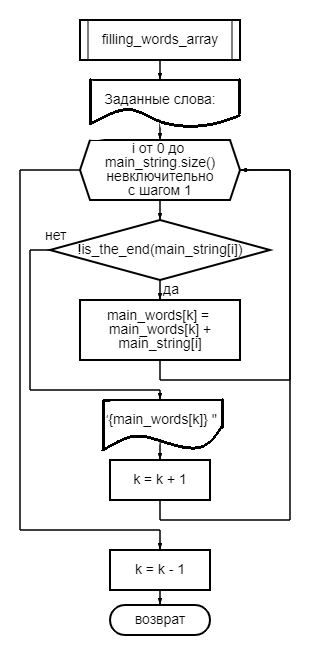
1. Блок-схемы:



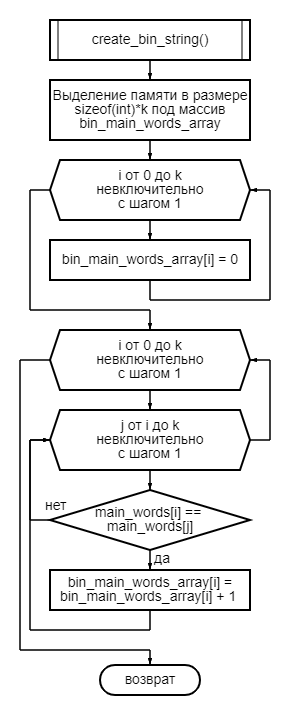
Блок-схема №1 – Основной код программы



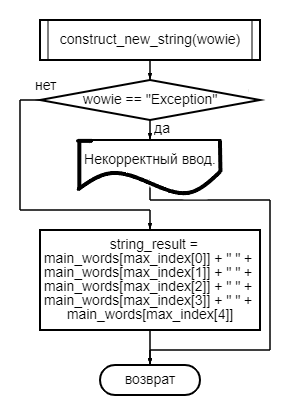
Блок-схема №2 – Конструктор класса file\_string



Блок-схема №3 – Заполнение массива слов



Блок-схема №4 – Создание цифрового массива строк из массива строк

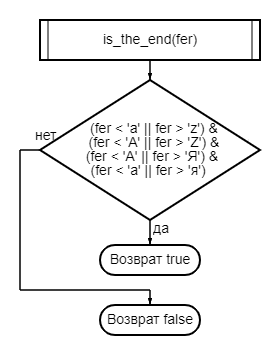


Блок-схема №5 – Конструирование новой строки

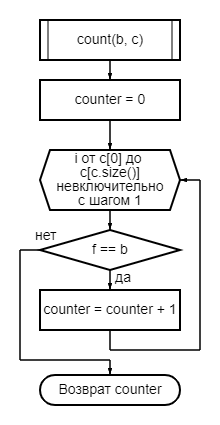
Изображение выглядит как текст

Автоматически созданное описание

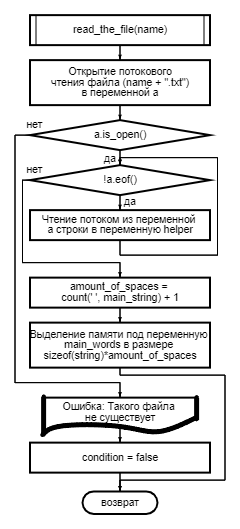
Блок-схема №6 – Печать результата



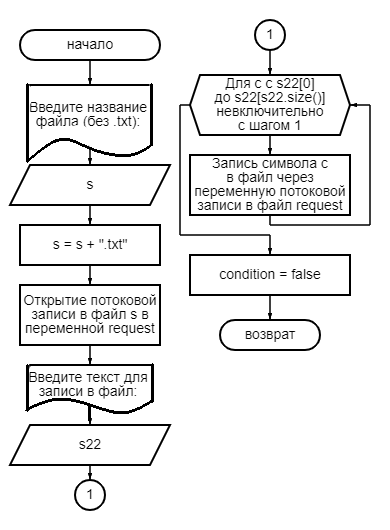
Блок-схема №7 – проверка на букву



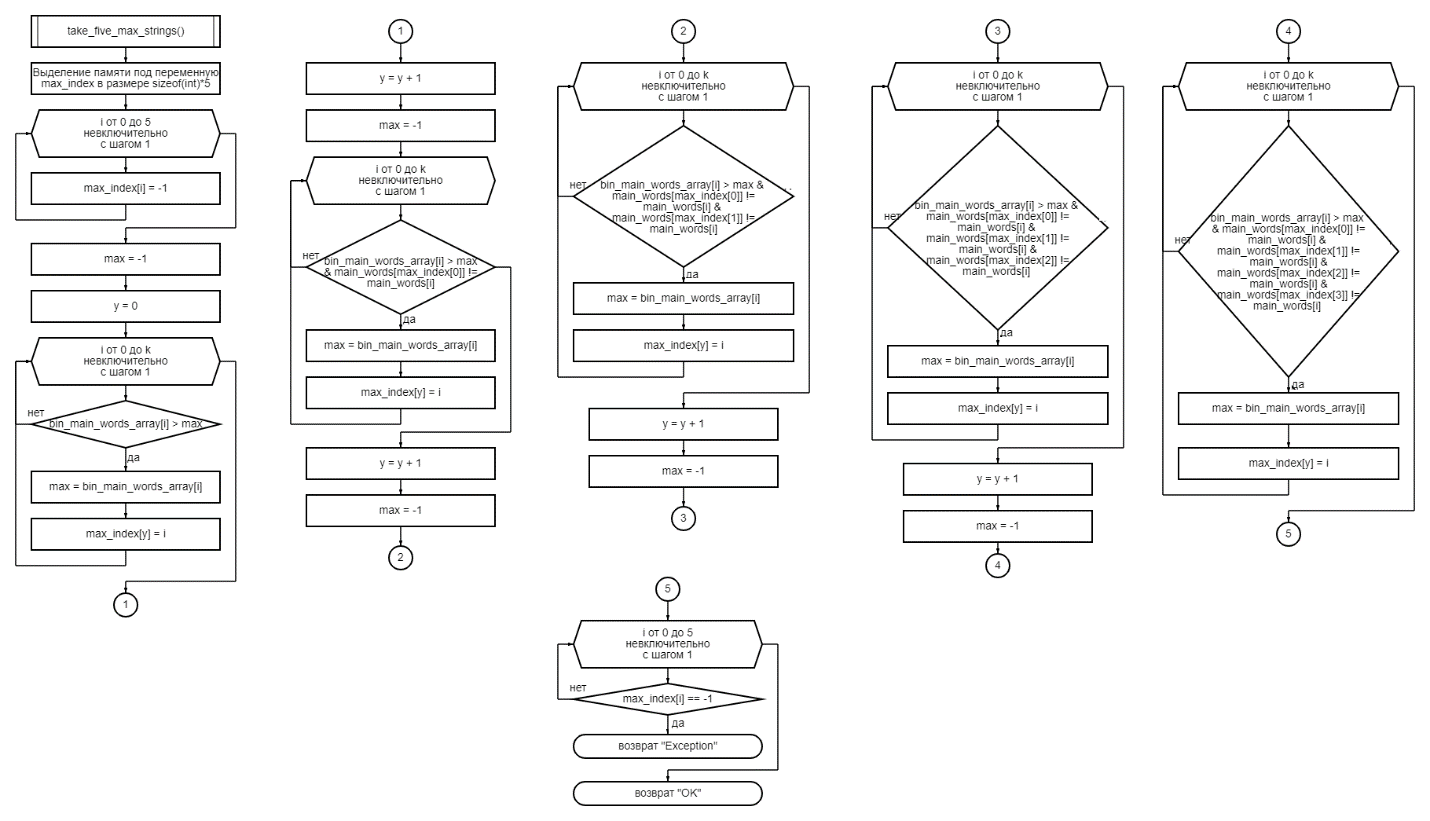
Блок-схема №8 – подсчёт количества буквы в заданной строке



Блок-схема №9 – Чтение файла



Блок-схема №10 – Создание файла



Блок-схема №11 – Забор пяти максимальных индексов

1. Отладочный пример:

1 вариант.

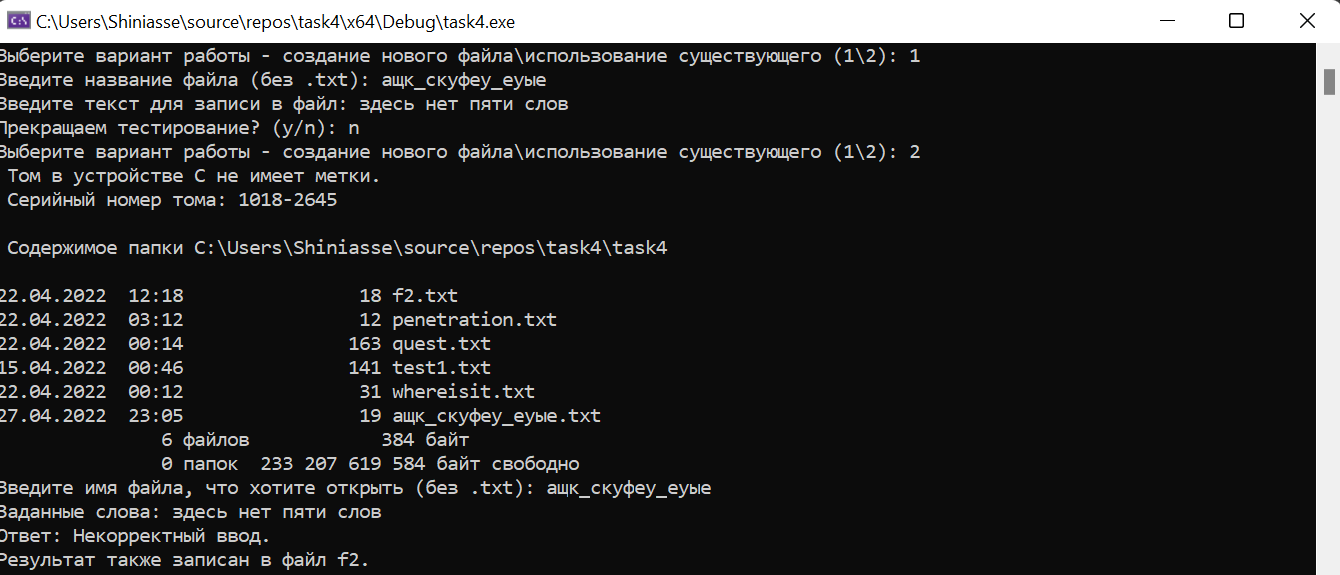
Покажем работу программы при корректном вводе.  
Изображение выглядит как текст

Автоматически созданное описание

II вариант.  
Покажем работу программы при некорректном вводе.  
Изображение выглядит как текст

Автоматически созданное описание

III Вариант.

Покажем работу программы при некорректном вводе.  


1. Код программы:

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

// Проверка, символ ли

bool is\_the\_end(char fer) {

if (

(fer < 'a' || fer > 'z') &&

(fer < 'A' || fer > 'Z') &&

(fer < 'А' || fer > 'Я') &&

(fer < 'а' || fer > 'я')

)

return true;

return false;

}

// Функция для подсчёта количества символов в заданной строке

int count(char b, string c) {

int counter = 0;

for (auto f : c)

if (f == b)

counter++;

return counter;

}

// Класс для обработки строк, что хранятся в файлах

class file\_string {

private:

string main\_string = "";

string string\_result = "";

int amount\_of\_spaces, k = 0;

string\* main\_words;

int\* bin\_main\_words\_array, \*max\_index;

public:

bool condition = true;

// Конструктор класса, отвечающий за создание\чтение файлов.

file\_string() {

bool decision; string f;

while (true) {

cout << "Выберите вариант работы - создание нового файла\\использование существующего (1\\2): ";

getline(cin, f);

if (f == "1") {

decision = 1;

break;

}

else if (f == "2") {

decision = 0;

break;

}

else

cout << "Вы ввели что-то не то. Попробуете ещё раз?" << endl;

}

switch (decision == true) {

case 1:

create\_the\_file();

break;

case 0:

system("dir \*.txt");

cout << "Введите имя файла, что хотите открыть (без .txt): ";

string dec; getline(cin, dec);

read\_the\_file(dec);

break;

}

}

// Метод для обработки случая создания файла пользователем

void create\_the\_file() {

cout << "Введите название файла (без .txt): ";

string s; getline(cin, s); s += ".txt";

ofstream request(s); string s22;

cout << "Введите текст для записи в файл: ";

getline(cin, s22);

for (auto c : s22) {

request << c;

}

condition = false;

}

// Метод для обработки случая чтения файла пользователем

void read\_the\_file(string name) {

string helper;

ifstream a(name + ".txt");

if (a.is\_open()) {

while (!a.eof()) {

a >> helper;

main\_string += helper + " ";

}

amount\_of\_spaces = count(' ', main\_string) + 1;

main\_words = new string[amount\_of\_spaces];

}

else {

cout << "Ошибка: Такого файла не существует" << endl;

condition = false;

}

}

// Метод для выделения подходящих по условию слов в массив

void filling\_words\_array() {

cout << "Заданные слова: ";

for (int i = 0; i < main\_string.size(); i++) {

if (!is\_the\_end(main\_string[i]))

main\_words[k] += main\_string[i];

else {

cout << main\_words[k] << " ";

k++;

}

}

k--;

}

// Создаем строку, каждый элемент - количество этого слова в общем массиве

void create\_bin\_string() {

bin\_main\_words\_array = new int[k];

for (int i = 0; i < k; i++)

bin\_main\_words\_array[i] = 0;

for (int i = 0; i < k; i++) {

for (int j = i; j < k; j++) {

if (main\_words[i] == main\_words[j])

bin\_main\_words\_array[i]++;

}

}

}

// Берем пять наиболее встречающихся слов

string take\_five\_max\_strings() {

max\_index = new int[5];

for (int i = 0; i < 5; i++)

max\_index[i] = -1;

int max = -1, y = 0;

for (int i = 0; i < k; i++) {

if (bin\_main\_words\_array[i] > max) {

max = bin\_main\_words\_array[i];

max\_index[y] = i;

}

}

y += 1, max = -1;

for (int i = 0; i < k; i++) {

if (bin\_main\_words\_array[i] > max &&

main\_words[max\_index[0]] != main\_words[i]) {

max = bin\_main\_words\_array[i];

max\_index[y] = i;

}

}

y += 1, max = -1;

for (int i = 0; i < k; i++) {

if (bin\_main\_words\_array[i] > max &&

main\_words[max\_index[0]] != main\_words[i] &&

main\_words[max\_index[1]] != main\_words[i]) {

max = bin\_main\_words\_array[i];

max\_index[y] = i;

}

}

y += 1, max = -1;

for (int i = 0; i < k; i++) {

if (bin\_main\_words\_array[i] > max &&

main\_words[max\_index[0]] != main\_words[i] &&

main\_words[max\_index[1]] != main\_words[i] &&

main\_words[max\_index[2]] != main\_words[i]) {

max = bin\_main\_words\_array[i];

max\_index[y] = i;

}

}

y += 1, max = -1;

for (int i = 0; i < k; i++) {

if (bin\_main\_words\_array[i] > max &&

main\_words[max\_index[0]] != main\_words[i] &&

main\_words[max\_index[1]] != main\_words[i] &&

main\_words[max\_index[2]] != main\_words[i] &&

main\_words[max\_index[3]] != main\_words[i]) {

max = bin\_main\_words\_array[i];

max\_index[y] = i;

}

}

for (int i = 0; i < 5; i++)

if (max\_index[i] == -1)

return "Exception";

return "OK";

}

// Строим новую строчку, в которой эти слова будут перечислены через пробел

void construct\_new\_string(string wowie) {

switch (wowie == "Exception") {

case 1:

string\_result = "Некорректный ввод.";

break;

case 0:

string\_result = main\_words[max\_index[0]] + " " +

main\_words[max\_index[1]] + " " +

main\_words[max\_index[2]] + " " +

main\_words[max\_index[3]] + " " +

main\_words[max\_index[4]];

break;

}

}

// Выводим строчку, которая удовлетворяет ответу в консоль и файл.

void print\_result() {

cout << "Ответ: " << string\_result << endl;

cout << "Результат также записан в файл f2.\n";

ofstream f2("f2.txt");

for (auto c : string\_result)

f2 << c;

}

};

int main() {

// Подключаем русский язык в консоли

setlocale(LC\_ALL, "rus");

system("chcp 1251 >> null");

// Для удобства тестирования программы

while (true) {

string q = "";

file\_string mainl;

if (mainl.condition == true) {

mainl.filling\_words\_array();

mainl.create\_bin\_string();

mainl.construct\_new\_string(mainl.take\_five\_max\_strings());

mainl.print\_result();

}

cout << "Прекращаем тестирование? (y/n): ";

getline(cin, q);

if (q == "y")

break;

}

system("pause >> null");

return 0;

}

1. Результат выполнения работы программы:

Изображение выглядит как текст

Автоматически созданное описание

Отладка №1 & №2 – Программа работает стабильно.

1. Вывод:

В ходе выполнения работы были изучены способы связи различных библиотек с работой классов – в частности, затрагивали тему библиотеки файловых потоков fstream. С помощью неё было реализовано меню для выбора файлов по названиям (показ имён файлов был осуществлен через консольную команду “dir”). Также была отлажена работа закрытых и открытых полей класса, также была связаны функции с работой классовых методов.

На контрольных примерах мы убедились, что код работает корректно.  
Был оформлен комплект документации на программный код.