Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное  
учреждение высшего образования  
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»  
(ФГАОУ ВО «ЮФУ»)

Институт компьютерных технологий и информационной безопасности

ОТЧЕТ  
по дисциплине «Объектно-ориентированное программирование»

**ЛАБОРАТОРНЫЕ РАБОТЫ**

Выполнил  
студент группы КТбо2-8 Мищенко Д.И

Принял  
доцент ИКТИБ, к. т. н. Лутай В.Н

Таганрог 2024

Содержание

[ЛАБОРАТОРНАЯ РАБОТА №1 “КЛАССЫ И ОБЪЕКТЫ” 3](#_Toc176955126)

[Цель работы 3](#_Toc176955127)

[Задание 3](#_Toc176955128)

[1.1 Спецификация класса 4](#_Toc176955129)

[1.2 Диаграмма класса 5](#_Toc176955130)

[1.3 Листинг программы 6](#_Toc176955131)

[1.4 Контрольные вопросы 11](#_Toc176955132)

**ЛАБОРАТОРНАЯ РАБОТА №1 “КЛАССЫ И ОБЪЕКТЫ”**

**Цель работы:**

Ознакомление с технологией построения классов, разработки методов, перегрузки операций и обработки исключений.

**Задание:**

Создать класс Указатель (Index), в котором реализован метод поиска задаваемого слова в некотором тексте и фиксации номера соответствующей страницы в массиве или в объекте типа string. В последнем номера страниц разделяются пробелом. Анализируемый текст содержится в файле и размечен по страницам. Страниц не менее 5, на каждой странице не менее 5 строк текста.

1.1 Спецификация класса

Класс Index реализует поиск слова в текстовом файле(название файла задается) и выводит все номера страниц где это слово присутствует.

**Приватные поля:**

1. **MAX\_PAGES\_** — Максимальное количество страниц, до которого будет считываться файл.
2. **MAX\_LINES\_ON\_PAGE**\_ — Максимальное количество строк на страницу, до которого будет считываться файл.
3. **pages\_** — Статический массив для хранения страниц
4. **pageCount\_** — Текущее количество страниц
5. **indexResult\_** — Для хранения результатов поиска

**Публичные методы:**

1. **Конструктор** Index() : pageCount\_(0) {} — создает класс и обнуляет текущие количество страниц
2. **Метод** loadText(const string& fileName) — Метод для загрузки текста из файла.

* Параметр: имя файла.
* Возвращаемое значение: отсутствует.

1. **Метод** findWord(const string& word); — Метод для поиска слова в тексте и фиксации номеров страниц

* Параметр: слово которое нужно найти.
* Возвращаемое значение: отсутствует.

1. **Метод** printResult() const; — Метод для вывода результатов поиска.

* Параметр: отсутствует.
* Возвращаемое значение: отсутствует.

1. **Деструктор** ~Index()

1.2 Диаграмма класса

Изображение выглядит как текст, снимок экрана, мультимедиа, гаджет

Автоматически созданное описание

1.3 Листинг программы

Файл index.h

#include <iostream> // Для вывода на консоль (std::cout)

#include <fstream> // Для работы с файлами (std::ifstream)

#include <sstream> // Для работы с потоками строк (std::stringstream)

#include <stdexcept> // Для обработки исключений (std::runtime\_error)

#include <string> // Для работы со строками (std::string)

using std::cout;

using std::cin;

using std::endl;

using std::ifstream;

using std::string;

using std::stringstream;

using std::runtime\_error;

using std::to\_string;

using std::exception;

class Index {

public:

static const int MAX\_PAGES\_ = 100; // Максимальное количество страниц

static const int MAX\_LINES\_ON\_PAGE\_ = 100; // Максимальное количество строк на страницу

string pages\_[MAX\_PAGES\_]; // Статический массив для хранения страниц

int pageCount\_; // Текущее количество страниц

string indexResult\_; // Для хранения результатов поиска

public:

// Создает класс и обнуляет текущие количество страниц

Index() : pageCount\_(0) {}

// Метод для загрузки текста из файла

void loadText(const string& fileName);

// Метод для поиска слова в тексте и фиксации номеров страниц

void findWord(const string& word);

// Метод для вывода результатов поиска

void printResult() const;

~Index() {};

};

Файл Index.cpp

#include "index.h"

// Метод загрузки текста из файла и разбиения на страницы

void Index::loadText(const string& fileName) {

ifstream file(fileName);

// Обработка ошибки открытия файла

if (!file.is\_open()) {

throw runtime\_error("Ошибка открытия файла!");

}

string line;

stringstream pageContent; // Для накопления строк страницы

int lineCounter = 0;

while (getline(file, line)) {

// Проверка максимального количества страниц

if (pageCount\_ >= MAX\_PAGES\_) {

throw runtime\_error("Достигнут лимит по количеству страниц 100");

}

pageContent << line << "\n"; // Добавляем строку в текущую страницу

lineCounter++;

// Обработка ошибки если в файле меньше 5 строк

if (lineCounter - 1 < 5 and line.empty()) {

throw runtime\_error("В файле должно быть не менее 5 строк!");

}

// Обработка ошибки если в файле нат одной странице больше 100 строк

if (lineCounter > MAX\_LINES\_ON\_PAGE\_) {

throw runtime\_error("Достигнут лимит по количеству строк на странице (100)" );

}

// Условие для создания новой страницы: либо превышено максимальное количество строк, либо встречена пустая строка

if (line.empty()) {

pages\_[pageCount\_] = pageContent.str(); // Сохраняем накопленную страницу

pageCount\_++; // Увеличиваем количество страниц

pageContent.str(""); // Очищаем для следующей страницы

pageContent.clear(); // Сбрасываем флаги потока

lineCounter = 0; // Сбрасываем счетчик строк

}

}

// Сохраняем последнюю страницу, если она существует

if (lineCounter > 0 and pageCount\_ < MAX\_PAGES\_) {

pages\_[pageCount\_] = pageContent.str();

pageCount\_++;

}

// Обработка ошибки если в файле меньше 5 страниц

if (pageCount\_ < 5) {

throw runtime\_error("В файле должно быть не менее 5 страниц!");

}

// Закрытие файла

file.close();

}

// Метод поиска слова и фиксации номеров страниц

void Index::findWord(const string& word) {

indexResult\_.clear(); // Очищаем предыдущие результаты

for (int i = 0; i < pageCount\_; ++i) {

if (pages\_[i].find(word) != string::npos) {

indexResult\_ += to\_string(i + 1) + " "; // Добавляем номер страницы в результат

}

}

if (indexResult\_.empty()) {

indexResult\_ = "Слово не найдено!";

}

}

void Index::printResult() const {

cout << "Слово найдено на страницах: " << indexResult\_ << endl;

}

Файл main.cpp

#include "index.h"

int main() {

setlocale(LC\_ALL, "Russian");

try {

string text, phindWord;

cout << "Введите имя файла.txt(файл будет считываться только до 100 страниц и до 100 строк): ";

cin >> text;

cout << "Введите слово которое надо найти: ";

cin >> phindWord;

cout << endl;

Index index;

index.loadText(text);

index.findWord(phindWord); // Поиск слова

index.printResult(); // Показать найденные страницы

}

catch (const exception& e) {

cout << "Ошибка: " << e.what() << endl;

}

return 0;

}

1.4 Контрольные вопросы

1. Для чего служит конструктор класса ? Может ли в классе быть несколько конструкторов? Чем должны отличаться различные конструкторы одного и того же класса?

2. Для чего служит деструктор класса? Имеет ли деструктор параметры?

3. Какие сообщения и в какой последовательности будут выведены на монитор после выполнения нижеследующей программы?

class Alpha {

public:

int x = 0;

Alpha() { cout << "Конструктор #1\n"; }

Alpha(int m) { cout << "Конструктор #2\n"; x = m; }

Alpha(const Alpha& other) { cout << "Конструктор #3\n"; x = other.x; }

~Alpha() { cout << "Деструктор\n"; }

};

int main() {

Alpha a1;

Alpha& a2 = a1;

Alpha\* a3 = nullptr;

a3 = new Alpha

Alpha a4[2];

Alpha a5(4);

Alpha a6(a1);

Alpha a7 = a1;

delete a3;

}

4. Имеется следующий класс class Alpha { Alpha(const Alpha& a){/\*...\*/} }; Какие члены класса в данном случае будут автоматически определены после завершения компиляции класса?

5. Почему операция «<<»во второй строке работает по-разному в разных точках выражения ?

int a=4;

//1 cout<<(a<<3)

6. Что такое отношение включения и как оно изображается на диаграммах? Что такое делегирование?