ГУАП

КАФЕДРА № 14

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | А. Ю. Петров |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
|  |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1243 |  | 09.12.2024 |  | М. Э. Зайнулин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

**1. Описание задания**

Вариант 9

1.Стандартные потоки:

Определить класс с именем TRAIN, содержащий следующие поля:

● название пункта назначения;

● номер поезда;

● время отправления.

Определить методы доступа к этим полям и перегруженные операции извлечения и вставки для объектов типа TRAIN. Заранее число объектов не известно. Написать программу, выполняющую следующие действия:

● записи должны быть упорядочены по номерам поездов;

● вывод на экран информации о поезде, номер которого введен с клавиатуры;

● если таких поездов нет, выдать на дисплей соответствующее сообщение.

2. Файловые и строковые потоки:

С использованием файловых и строковых потоков написать программу, которая считывает текст из файла и выводит на экран только предложения, состоящие из заданного количества слов.

**2. Формализация**

Задача 1 реализуется с помощью двух классов: Train и List. Класс Train (определение в train.h, реализация в train.cpp) описывает поезд, храня информацию о пункте назначения, номере и времени отправления. Он включает конструкторы (без параметров, с параметрами, копирования), деструктор, методы доступа (get/set) к полям, а также публичные методы отображения информации о всех поездах и изменения данных поезда.

Класс List (определение в List.h, реализация в List.cpp) управляет списком поездов, используя структуру связного списка. Каждый узел списка (Node) содержит указатель на следующий узел и объект класса Train. List хранит указатели на начало и конец списка, а также количество элементов. Он предоставляет конструкторы (без параметров, с параметрами, копирования), деструктор, методы добавления, удаления, изменения и отображения поездов, сортировки по номеру поезда, поиска по номеру и перегруженный оператор доступа по индексу. Программа использует стандартные потоки ввода-вывода (iostream).

Задача 2 решается с помощью класса SentenseFilter (определение в sentence\_filter.h, реализация в sentence\_filter.cpp). Этот класс обрабатывает тексты, определяя количество слов. В частной области находятся поля для хранения текста, количества слов и флага подключения к файлу, а также методы разбиения текста на слова и подсчета слов. В публичной области – конструкторы (без параметров, с параметрами, копирования), деструктор и метод вывода результата обработки. Программа использует строковые (sstream) и файловые (fstream) потоки.

Функция проверки корректности вводимых данных описана в check.h и реализована в соответствующем .cpp файле. Главный файл main.cpp содержит пользовательский интерфейс для работы с обеими задачами, обработку исключений и обеспечивает взаимодействие с пользователем через консоль. Программа использует динамическое выделение памяти и не использует контейнеры из стандартной библиотеки шаблонов (STL).

**3. Исходный код программы**

https://github.com/Ksoxsane/lab2

**Файл list.h**

#ifndef TRAINLIST\_H

#define TRAINLIST\_H

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include "train.h"

struct Node

{

Node\* next;

TRAIN\* data;

};

class TrainList {

private:

Node\* head;

Node\* tail;

int count;

public:

TrainList();

TrainList(Node\* head, Node\* tail, const int counter);

TrainList(const TrainList& other);

~TrainList();

Node\* get\_head();

Node\* get\_tail();

void addTrain(TRAIN\* Tr, int index);

TrainList& delete\_train(int index);

TrainList& edit(int index);

void printAll() const;

void printTrainsToNumber(const int number);

void sortByNumber();

TrainList& operator[](int index);

};

#endif

**Файл list.cpp**

#include "list.h"

#include <algorithm>

TrainList::TrainList() : head(nullptr), tail(nullptr), count(0) {

cout << "The constructor is called without parameters for the TrainList class\n";

}

TrainList::TrainList(Node\* h, Node\* t, const int c) : head(h), tail(t), count(c) {

cout << "The constructor is called with parameters for the TrainList class\n";

}

TrainList::TrainList(const TrainList& other) : head(other.head), tail(other.tail), count(other.count) {

cout << "The copy constructor for the TrainList class is called\n\n";

}

TrainList::~TrainList() {

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp->data;

delete temp;

}

tail = nullptr;

count = 0;

cout << "The destructor for the TrainList class is called\n" << endl;

}

Node\* TrainList::get\_head() {

return this->head;

}

Node\* TrainList::get\_tail() {

return this->tail;

}

void TrainList::addTrain(TRAIN\* Tr, int index) {

if (index < 0 || index > count) {

throw out\_of\_range("Index out of range");

}

Node\* node\_to\_add = new Node;

node\_to\_add->data = Tr;

node\_to\_add->next = nullptr;

if (index == 0) {

node\_to\_add->next = head;

head = node\_to\_add;

if (count == 0) {

tail = node\_to\_add;

}

} else {

Node\* prev = head;

for (int i = 0; i < index - 1; ++i) {

prev = prev->next;

}

node\_to\_add->next = prev->next;

prev->next = node\_to\_add;

if (node\_to\_add->next == nullptr) {

tail = node\_to\_add;

}

}

++count;

}

TrainList& TrainList::edit(int index) {

if (index < 0 || index >= count) {

throw out\_of\_range("Index out of range");

}

Node\* temp = head;

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

temp = temp->next;

}

temp->data->edit();

return \*this;

}

TrainList& TrainList::delete\_train(int index) {

if (index < 0 || index >= count) {

throw out\_of\_range("Index out of range");

}

Node\* temp = head;

if (index == 0) {

head = head->next;

delete temp->data;

delete temp;

if (head == nullptr) {

tail = nullptr;

}

} else {

Node\* prev = nullptr;

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

prev = temp;

temp = temp->next;

}

prev->next = temp->next;

if (temp == tail) {

tail = prev;

}

delete temp->data;

delete temp;

}

--count;

return \*this;

}

void TrainList::printAll() const {

Node\* temp = head;

if (count == 0) {

cout << "No trains available.\n";

return;

}

int index = 1;

while (temp != 0) {

cout << index << " - Info:\n";

temp->data->print();

temp = temp->next;

index++;

}

}

void TrainList::sortByNumber() {

if (count < 2) return;

for (Node\* i = head; i != nullptr; i = i->next) {

for (Node\* j = head; j->next != nullptr; j = j->next) {

if (j->data->getNumber() > j->next->data->getNumber()) {

TRAIN\* temp = j->data;

j->data = j->next->data;

j->next->data = temp;

}

}

}

}

void TrainList::printTrainsToNumber(const int number) {

Node\* temp = head;

while (temp != nullptr) {

if (temp->data->getNumber() == number) {

temp->data->print();

return;

}

temp = temp->next;

}

cout << "Train " << number << " not found." << endl;

}

TrainList& TrainList::operator[](int index) {

if (index < 0 || index >= count) {

throw out\_of\_range("Index out of range");

}

Node\* temp = head;

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

temp = temp->next;

}

return \*this;

}

**Файл train.h**

#ifndef TRAIN\_H

#define TRAIN\_H

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

class TRAIN {

private:

string destination;

int number;

string time;

public:

TRAIN();

TRAIN(string destination, int number, string time);

string getDestination();

void setDestination(const string &destination);

int getNumber();

void setNumber(const int &number);

string gettime();

void settime(const string &timer);

void print();

void edit();

friend ostream& operator<<(ostream& os, const TRAIN& train);

friend istream& operator>>(istream& is, TRAIN& train);

};

#endif

**Файл train.cpp**

#include "Train.h"

TRAIN::TRAIN() : destination(""), number(0), time("00:00") {}

TRAIN::TRAIN(string destination, int number, string time)

: destination(destination), number(number), time(time) {}

string TRAIN::getDestination() {

return this->destination;

}

void TRAIN::setDestination(const string& d) {

this->destination = d;

}

int TRAIN::getNumber() {

return this->number;

}

void TRAIN::setNumber(const int& n) {

this->number = n;

}

string TRAIN::gettime() {

return this->time;

}

void TRAIN::settime(const string& t) {

this->time = t;

}

void TRAIN::print() {

cout << "Destination: " << this->getDestination() << endl;

cout << "Number: " << this->getNumber() << endl;

cout << "Time of Department: " << this->gettime() << endl;

cout << endl;

}

void TRAIN::edit() {

int choice;

cout << "Change:\n";

cout << "1. Destination\n";

cout << "2. Number\n";

cout << "3. Time of Department\n";

cin >> choice;

switch (choice) {

case 1: {

string new\_destination;

cout << "Enter new Destination: ";

cin.ignore();

getline(cin, new\_destination);

setDestination(new\_destination);

break;

}

case 2: {

int new\_number;

cout << "Enter new Number: ";

cin >> new\_number;

setNumber(new\_number);

break;

}

case 3: {

string new\_time;

cout << "Enter new Timfe of department: ";

cin.ignore();

getline(cin, new\_time);

settime(new\_time);

break;

}

default:

cout << "Ivalid!" << endl;

}

}

**Файл main.cpp**

#include "list.h"

#include <iostream>

using namespace std;

int main() {

TrainList trainList;

int choice;

do {

cout << "\nMenu:\n";

cout << "1. Add Train\n";

cout << "2. Delete Train\n";

cout << "3. Edit Trains\n";

cout << "4. Print All Trains\n";

cout << "5. Sort Trains by number\n";

cout << "6. Search Trains by number\n";

cout << "0. Exit\n";

cout << "Enter your choice: ";;

cin >> choice;

switch (choice) {

case 1: {

string destination, time;

int number, index;

cout << "Destination: ";

getline(cin >> ws, destination);

cout << "Number of Train: ";

cin >> number;

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Time of Departure: ";

getline(cin, time);

cout << "Enter index of train: ";

cin >> index;

TRAIN\* new\_train = new TRAIN(destination, number, time);

try {

trainList.addTrain(new\_train, index - 1);

cout << "Added." << endl;

} catch (const out\_of\_range& e) {

cout << e.what() << endl;

delete new\_train;

}

break;

}

case 2: {

int index;

cout << "Enter index of the train: ";

cin >> index;

try {

trainList.delete\_train(index - 1);

cout << "Deleted." << endl;

} catch (const out\_of\_range& e) {

cout << e.what() << endl;

}

break;

}

case 3: {

int index;

cout << "Enter index of the train: ";

cin >> index;

try {

trainList.edit(index - 1);

cout << "Edited." << endl;

} catch (const out\_of\_range& e) {

cout << e.what() << endl;

}

break;

}

case 4: {

trainList.printAll();

break;

}

case 5: {

trainList.sortByNumber();

cout << "Successfully sorted." << endl;

trainList.printAll();

break;

}

case 6: {

cout << "Enter the train number to search for: ";

int number;

cin >> number;

trainList.printTrainsToNumber(number);

break;

}

case 0: {

cout << "Exiting...\n" << endl;

return 0;

}

default: {

cout << "Invalid choice. Please try again.\n" << endl;

break;

}

}

} while (choice != 0);

return 0;

}

**Файл second.h**

#pragma once

#include <iostream>

#include <fstream>

#include <sstream>

#include <cstring>

#include <cctype>

#include <string>

using namespace std;

class Filter {

private:

string source;

int count;

bool isTextSource;

void split(const string& text, string\*& sentences, int&sentenceCount) const;

int count\_words(const string& sentence) const;

public:

Filter();

Filter(const string& filename, int wordCount);

Filter(const string& text, int wordCount, bool isText);

Filter(const Filter& other);

~Filter();

void result() const;

};

**Файл second.cpp**

#include "second.h"

Filter::Filter() : source(""), count(0), isTextSource(false)

{

cout << "The constructor is called without parameters for the Filter class\n";

}

Filter::Filter(const string& filename, int count) : source(filename), count(count), isTextSource(false) {

cout << "The constructor is called with parameters for the Filter class\n";

}

Filter::Filter(const string& text, int count, bool isText) : source(text), count(count), isTextSource(isText) {

cout << "The constructor is called with parameters for the Filter class\n";

}

Filter::Filter(const Filter& other) : source(other.source), count(other.count), isTextSource(other.isTextSource) {

cout << "The copy constructor for the Filter class is called\n";

}

Filter::~Filter() {

cout << "The destructor for the Filter class is called\n";

}

void Filter::result() const {

string text;

if (isTextSource) {

text = source;

}

else {

ifstream file(source);

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

cerr << "Error: " << source << endl;

return;

}

ostringstream buffer;

buffer << file.rdbuf();

text = buffer.str();

file.close();

}

cout << "Your text:\n" << text << "\n\n";

string\* sentences = nullptr;

int sentenceCount = 0;

split(text, sentences, sentenceCount);

cout << "Amount of sentences: " << sentenceCount << endl;

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

{

int words = count\_words(sentences[i]);

if (words == count) {

cout << sentences[i] << endl;

}

}

delete[] sentences;

}

void Filter::split(const string& text, string\*& sentences,

int& sentenceCount) const {

const int maxSentences = 100;

sentences = new string[maxSentences];

sentenceCount = 0;

ostringstream sentenceStream;

for (char ch : text) {

sentenceStream << ch;

if (ch == '.' || ch == '!' || ch == '?') {

if (sentenceCount < maxSentences) {

sentences[sentenceCount++] = sentenceStream.str();

sentenceStream.str("");

sentenceStream.clear();

}

}

}

if (!sentenceStream.str().empty() && sentenceCount < maxSentences) {

sentences[sentenceCount++] = sentenceStream.str();

}

}

int Filter::count\_words(const string& sentence) const {

int count = 0;

istringstream ss(sentence);

string word;

while (ss >> word) {

++count;

}

return count;

}

**Файл main2.cpp**

#include "second.h"

#include "iostream"

int main() {

try {

int wordCount;

cout << "Enter amount of words in sentence: ";

if (!(cin >> wordCount)) {

throw invalid\_argument("Incorrect input of the number of words.");

}

int choice;

cout << "Line or File?\n 1 - Line\n 2 - File\n";

if (!(cin >> choice)) {

throw invalid\_argument("Incorrect input of the source selection.");

}

if (choice == 1) {

cin.ignore();

char text[8192];

cout << "Enter text: ";

cin.getline(text, sizeof(text));

if (cin.fail()) {

throw overflow\_error("Out of range.");

}

Filter filter(text, wordCount, true);

filter.result();

} else if (choice == 2) {

char filename[256];

cout << "Enter file's name: ";

cin >> filename;

ifstream file(filename);

if (!file) {

throw runtime\_error("The file was not found or could not be opened.");

}

file.close();

Filter filter(filename, wordCount);

filter.result();

} else {

throw out\_of\_range("A non-existent source has been selected");

}

}

catch (const invalid\_argument& e) {

cerr << "Error: " << e.what() << endl;

return 1;

}

catch (const overflow\_error& e) {

cerr << "Error: " << e.what() << endl;

return 2;

}

catch (const runtime\_error& e) {

cerr << "Error: " << e.what() << endl;

return 3;

}

catch (const out\_of\_range& e) {

cerr << "Error: " << e.what() << endl;

return 4;

}

catch (...) {

cerr << " Error." << endl;

return -1;

}

return 0;

}

**4. Результат работы программы**

На рис. 1 продемонстрировано меню первой программы

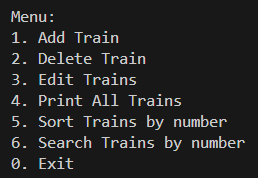


Рисунок 1 – Меню первой программы

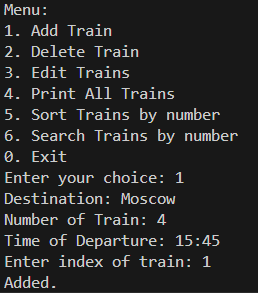


Рисунок 2 – Добавление поезда

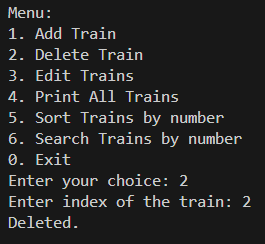


Рисунок 3 – Удаление поезда

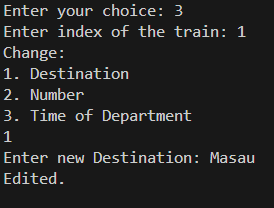


Рисунок 4 – Редактирование поезда

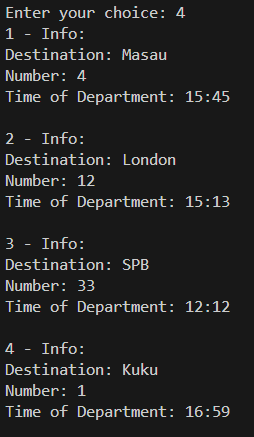


Рисунок 5 – Отображение списка поездов

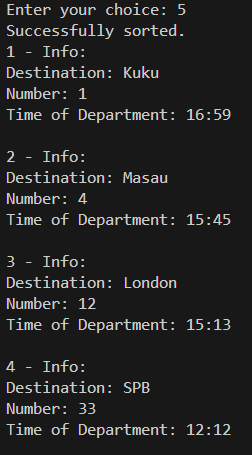


Рисунок 6 – Сортировка поездов по номеру

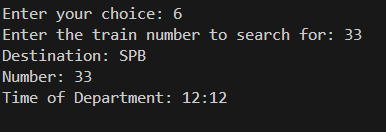


Рисунок 7 – Поиск поезда по номеру

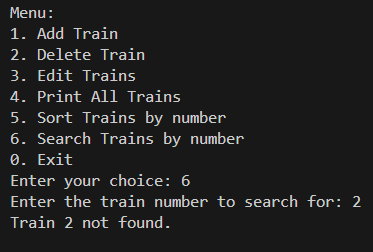


Рисунок 8 – Поиск несуществующего поезда

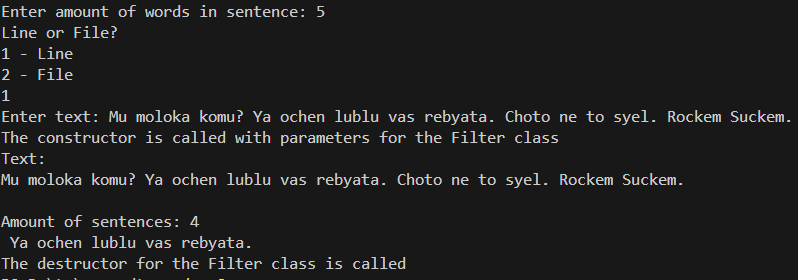


Рисунок 9 – Демонстрация работы второй части задания (1)

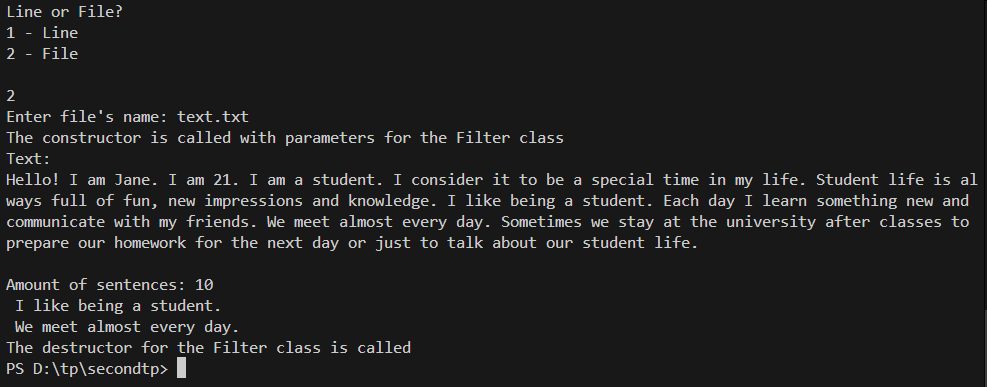


Рисунок 10 – Демонстрация работы второй части задания (2)

**5. Выводы**

В данной лабораторной работе требовалось изучить перегрузку унарных и бинарных операторов при помощи методов класса и дружественных функций. При тестировании работы не было выявлено ошибок, программа выдала правильный результат, из чего можно сделать вывод, что лабораторная работа выполнена успешно.