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: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1042 |  |  |  | Н.В. Корзун |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2022

1. **Постановка задачи**

Написать программы согласно требованиям:

Меню пользователя должно содержать возможность

редактирования любой записи любого объекта, удаление любого

объекта, добавления нового объекта на любую позицию.

**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.

1. **Формализация задачи**
   1. **Стандартные потоки**

Данная программа разбита на 4 файлов

* main.cpp - файл содержащий функцию main и все управление вызовами.
* marray.h - файл содержит класс для хранения поезда, реализован с помощью шаблона
* train.h - заголовочный файл для класса Train
* train.cpp - файл с реализацией класса Train
  1. **Файловые и строковые потоки**
* Main.cpp - файл содержащий функцию main и алгоритм вывода нужных предложений.
* Marray.h - файл содержит класс для хранения строк, реализован с помощью шаблона
  1. **Общая характеристика двух написанных программ**

Пользователю доступен ввод данных через консоль (вручную).

Важно: в водимых строках нельзя использовать пробелы и переходы на новую строку. А в вводимые числа должны быть не отрицательны. Нельзя вводить строки, когда требуется ввести число.

В программе реализована работа с динамической памятью. В программе присутствует обработка исключений, также Подстановка макросов препроцессором.

Файлы разделены на .h и .cpp.

Для классов определены методы для просмотра и установки значений (set и get). Также созданы Конструкторы и деструкторы, вызов которых сопровождается извещением в консоль.

Через меню есть доступ ко всем членам объектов-наследников.

1. **Структуры, функции, классы**

В программах присутствуют перегруженные функции некоторых операторов, это позволяет сделать код более читаемым, и удобным.

Классы List, Marray – используются в различных программах, но они похожи, они нужны для хранения объектов.

Наследования нет, есть композиция в классе List, вложенный класс Node

1. **Исходный код**

Файл main.cpp / Стандартные потоки

#include<iostream>

#include<fstream>

#include"train.h"

#include"Marray.h"

using namespace std;

int safeInput(int minInput, int maxInput) {

while (true)

{

int method;

std::cin >> method;

if (std::cin.fail() || method <minInput || method >maxInput)

{

std::cin.clear();

std::cout << "err\n";

}

else

{

std::cin.ignore(32767, '\n');

std::cin.clear();

return method;

}

std::cin.ignore(32767, '\n');

}

}

void printMenu();

void addObject(Marray<TRAIN>& trains);

void changeObject(Marray<TRAIN>& trains);

void deleteObject(Marray<TRAIN>& trains);

void deleteAll(Marray<TRAIN>& trains);

void print(Marray<TRAIN>& trains);

void printLast(Marray<TRAIN>& trains);

void searchObjects(Marray<TRAIN>& trains);

int main() {

Marray<TRAIN> trains;

bool isExit = false;

while (!isExit) {

printMenu();

int method = safeInput(0, 7);

switch (method)

{

case 1:

addObject(trains);

break;

case 2:

changeObject(trains);

break;

case 3:

deleteObject(trains);

break;

case 4:

deleteAll(trains);

break;

case 5:

print(trains);

break;

case 6:

searchObjects(trains);

break;

case 7:

printLast(trains);

break;

case 0:

isExit = true;

break;

}

}

}

void printMenu() {

cout << "1. add obj" << endl;

cout << "2. change obj" << endl;

cout << "3. delete obj" << endl;

cout << "4. delete all obj's" << endl;

cout << "5. print all obj's" << endl;

cout << "6. search obj's" << endl;

cout << "7. print last obj" << endl;

cout << "0. exit" << endl;

cout << "choice item: ";

}

void addObject(Marray<TRAIN>& trains) {

TRAIN train;

train.inputFromConsole();

trains += train;

trains.sort();

}

void changeObject(Marray<TRAIN>& trains) {

if (trains.getSize() > 0) {

print(trains);

cout << "input pos for change: ";

trains[safeInput(1, trains.getSize()) - 1].change();

}

else

cout << "empty arr\n";

trains.sort();

}

void deleteObject(Marray<TRAIN>& trains) {

if (trains.getSize() > 0) {

print(trains);

cout << "input pos for del: ";

trains -= safeInput(1, trains.getSize()) - 1;

}

else

cout << "empty arr\n";

trains.sort();

}

void deleteAll(Marray<TRAIN>& trains) {

if (trains.getSize() > 0)

trains.clear();

else

std::cout << "empty arr\n";

trains.sort();

}

void print(Marray<TRAIN>& trains) {

if (trains.getSize() == 0)

cout << "empty arr\n";

else {

cout << "arr:\n";

for (int i = 0; i < trains.getSize(); i++) {

cout << i + 1 << ".\n";

trains[i].printToConsole();

}

}

}

void printLast(Marray<TRAIN>& trains) {

if (trains.getSize() == 0)

cout << "empty arr\n";

else {

cout << "last el:\n";

int i = trains.getSize() - 1;

trains[i].printToConsole();

}

}

void searchObjects(Marray<TRAIN>& trains) {

if (trains.getSize() == 0)

cout << "empty arr\n";

else {

bool isPrint = false;

uint64\_t number;

cout << "input number\n";

cin >> number;

for (int i = 0; i < trains.getSize(); i++)

if (trains[i].get\_train\_number() == number) {

isPrint = true;

trains[i].printToConsole();

}

if (!isPrint)

cout << "empty search.\n";

}

}

Файл marray.h / Стандартные потоки

#pragma once

#include<string>

template<class T>

class Marray

{

private:

int size;

T\* data;

public:

Marray() : size(0), data(nullptr) {}

Marray(const Marray<T>& myArray);

~Marray() { clear(); }

void clear();

void sort();

const int getSize() { return size; }

T& operator[](int);

Marray<T>& operator+=(const T& obj);

Marray<T>& operator-=(const int index);

Marray<T>& operator=(const Marray<T>& myArray);

};

template<class T>

Marray<T>::Marray(const Marray<T>& myArray) {

data = new T[myArray.size];

size = myArray.size;

for (int i = 0; i < myArray.size; i++)

data[i] = myArray.data[i];

}

template<class T>

T& Marray<T>::operator[](int index)

{

std::string err = "Index " + std::to\_string(index) + " does not exist";

if (index < 0 || index >= size)

throw err;

return data[index];

}

template<class T>

Marray<T>& Marray<T>::operator+=(const T& obj) {

T\* tmp = data;

size++;

data = new T[size];

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

data[i] = tmp[i];

data[size - 1] = obj;

return \*this;

}

template<class T>

Marray<T>& Marray<T>::operator-=(const int index) {

std::string err = "Index " + std::to\_string(index) + " does not exist";

if (index < 0 || index >= size)

throw err;

T\* tmp = new T[size - 1];

int k = 0;

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

if (i != index)

tmp[k++] = data[i];

size--;

data = tmp;

return \*this;

}

template<class T>

Marray<T>& Marray<T>::operator=(const Marray<T>& myArray)

{

if (this == &myArray)

return \*this;

delete[] data;

data = new T[myArray.size];

size = myArray.size;

for (int i = 0; i < myArray.size; i++)

data[i] = myArray.data[i];

return \*this;

}

template<class T>

void Marray<T>::sort() {

for (int i = 1; i < size; i++)

for (int j = 0; j < size - i; j++)

if (data[j] > data[j + 1])

std::swap(data[j], data[j + 1]);

}

template<class T>

void Marray<T>::clear() {

delete[] data;

data = nullptr;

size = 0;

}

Файл train.h / Стандартные потоки

#pragma once

#include <iostream>

#include <string>

using namespace std;

class TRAIN

{

private:

string destination;

uint64\_t train\_number;

string departure\_time;

public:

TRAIN();

TRAIN(string destination, uint64\_t train\_number, string time);

TRAIN(const TRAIN& note);

~TRAIN();

void inputFromConsole();

void printToConsole();

void change();

string get\_destination() { return destination; }

void set\_destination(string destination) { this->destination = destination;}

uint64\_t get\_train\_number() { return train\_number; }

void set\_train\_number(uint64\_t train\_number) { this->train\_number = train\_number; }

string get\_departure\_time() { return departure\_time; }

void set\_departure\_time(string departure\_time) { this->departure\_time = departure\_time; }

TRAIN& operator=(const TRAIN& train);

bool operator > (TRAIN& train);

};

Файл train.cpp / Стандартные потоки

#include"train.h"

using namespace std;

TRAIN::TRAIN() : destination("Не задано"), train\_number(NULL), departure\_time("Y:M:D h:m:s")

{

cout << "constructor without params TRAIN" << endl;

}

TRAIN::TRAIN(string destination\_, uint64\_t train\_number\_, string time) :

destination(destination), train\_number(train\_number\_), departure\_time(time)

{

cout << "constructor with params TRAIN" << endl;

}

TRAIN::TRAIN(const TRAIN& train)

{

this->departure\_time = train.departure\_time;

this->destination = train.destination;

this->train\_number = train.train\_number;

cout << "copy constructor TRAIN" << endl;

}

TRAIN::~TRAIN()

{

cout << "destructor TRAIN" << endl;

}

void TRAIN::inputFromConsole()

{

cout << "Input destination: ";

getline(cin, destination);

cout << "Input departure time: ";

getline(cin, departure\_time);

int flag = 1;

while(flag)

try

{

cout << "Input train number: ";

cin >> train\_number;

flag = 0;

}

catch (const char\* er)

{

cout << er << endl;

cout << "exception catch!" << endl;

}

}

void TRAIN::printToConsole()

{

cout << "destination: " << destination << endl;

cout << "train\_number: " << train\_number << endl;

cout << "departure\_time: " << departure\_time<< endl;

}

void TRAIN::change()

{

cout << "input new param\n";

inputFromConsole();

}

TRAIN& TRAIN::operator=(const TRAIN& train) {

if (this == &train)

return \*this;

destination = train.destination;

train\_number = train.train\_number;

departure\_time = train.departure\_time;

return \*this;

}

bool TRAIN::operator>(TRAIN& train){

return this->train\_number > train.train\_number;

}

Файл Main.cpp / Файловые и строковые потоки.

#include <iostream>

#include <string>

#include <vector>

#include <sstream>

#include <fstream>

#include "Marray.h"

using namespace std;

/\*

С использованием файловых и строковых потоков написать программу,

которая считывает текст из файла и выводит на экран только предложения,

состоящие из заданного количества слов.

\*/

int main()

{

setlocale(LC\_ALL, "Russian");

std::ifstream ifs("in.txt");

List<std::string> vec;

//std::vector<std::string> vec;

size\_t n = 0;

std::cout << "Input count words in sentence: ";

std::cin >> n;

if (!ifs)

{

std::cerr << "ERROR" << std::endl;

return 1;

}

else

{

while (!ifs.eof())

{

std::string tmp;

getline(ifs, tmp, '.');

vec.push\_back(tmp);

}

}

size\_t cnt;

for (size\_t i = 0; i < vec.size(); ++i)

{

cnt = 0;

std::string tmp = vec[i];

std::istringstream ist(tmp);

while (ist >> tmp)

++cnt;

if (cnt == n)

std::cout << vec[i] << '.';

}

return 0;

}

Файл Marray.h / Файловые и строковые потоки.

#pragma once

using namespace std;

template<typename T>

class List

{

private:

template<typename T>

class Node

{

public:

Node\* pNext; //указатель на следующий элемент

T data;//данные

Node(T data = T(), Node\* pNext = nullptr)

{

this->data = data;

this->pNext = pNext;

}

};

int Size;

Node<T>\* head; //указатель на начало списка

public:

int size() { return Size; }

~List();

List();

void push\_back(const T data);

void pop\_front();

T& operator[](const int index);

bool operator > (List& r);

bool operator <(List& r);

};

template<typename T>

List<T>::List()

{

Size = 0;

head = nullptr;

}

template<typename T>

List<T>::~List()

{

}

template<typename T>

T& List<T>::operator[](const int index)

{

Node<T>\* current = this->head;//âðåìåííàÿ ïåðåìåííàÿÿ, ïîêàçûâàþùàÿ êîíêðåòíûé ýëåìåíò

int counter = 0;// ïåðåìåííàÿ ñ÷åò÷èê-ýëåìåíòîâ

while (current != nullptr)

{

if (counter == index)

{

return current->data;

}

current = current->pNext;

counter++;

}

}

template<typename T>

void List<T>::push\_back(const T data)

{

head = new Node<T>(data, head);

Size++;

}

template<typename T>

inline bool List<T>::operator>(List& r)

{

if (this->GetSize() > r.GetSize())

{

return true;

}

else

{

return false;

}

}

template<typename T>

inline bool List<T>::operator<(List& r)

{

if (this->GetSize() < r.GetSize())

{

return true;

}

else

{

return false;

}

}

template<typename T>

inline void List<T>::pop\_front()

{

Node<T>\* temp = head;

head = head->pNext;

delete temp;

Size--;

}

1. **Результаты работы программ**

Пример двух последовательных запусков первой программы.

При первом запуске создаем первый объект, выводим его, меняем его совершая ошибку, которую программа поймала. Все-таки меняем его, добавляем второй объект, выводим их, и ищем нужный – первый.

При повторном запуске не создаем ничего, и пытаемся проделать различные действия с пустым массивом, на что получаем закономерный результат – сообщение о том что массив пуст.

Пример двух последовательных запусков второй программы.

При первом запуске в текстовом файле одни данные, при втором другие.

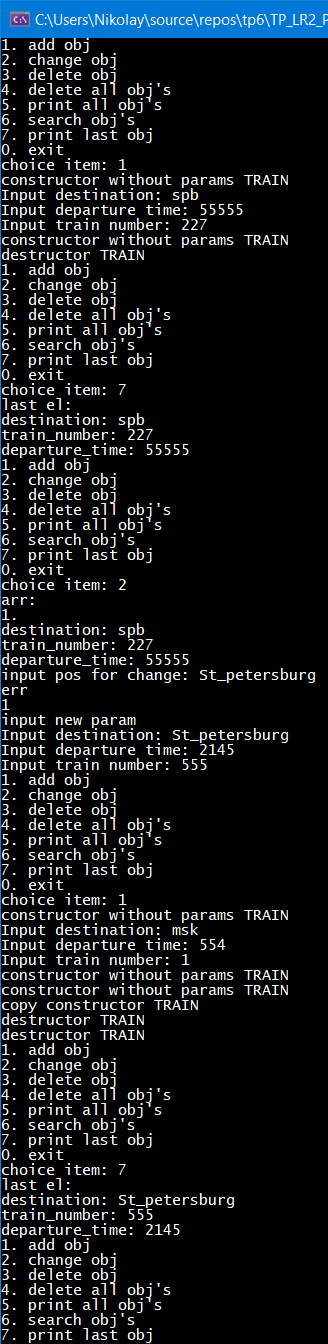
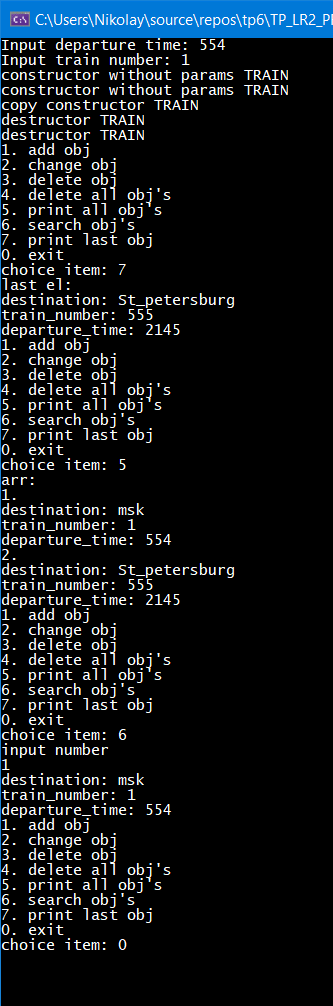


Рисунок 1. Скриншот экрана консоли, программа 1, пример 1 начало

Рисунок 2. Скриншот экрана консоли, программа 1, пример 1 продолжение

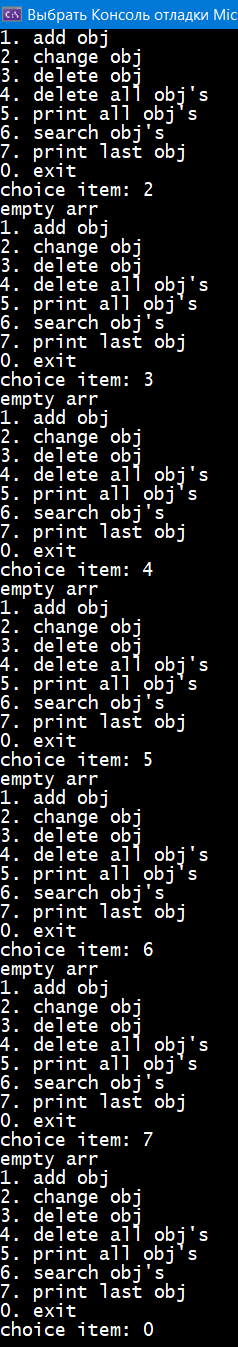


Рисунок 3. Скриншот экрана консоли, программа 1, пример 2

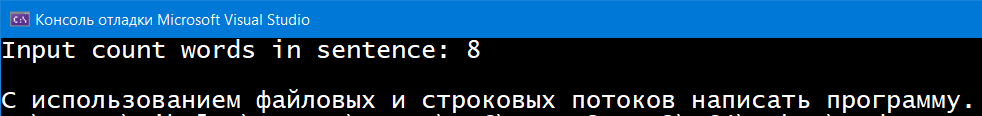


Рисунок 4. Скриншот экрана консоли, программа 2, пример 1

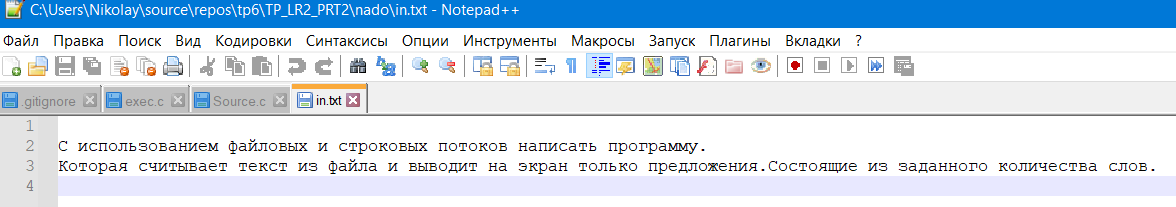


Рисунок 5. Скриншот текстового файла, программа 2, пример 1

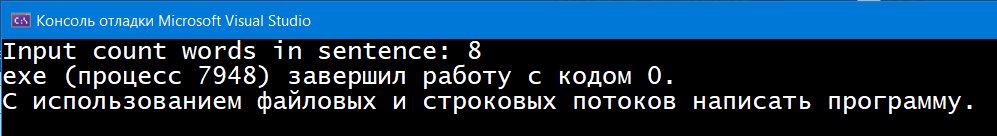


Рисунок 6. Скриншот экрана консоли, программа 2, пример 1

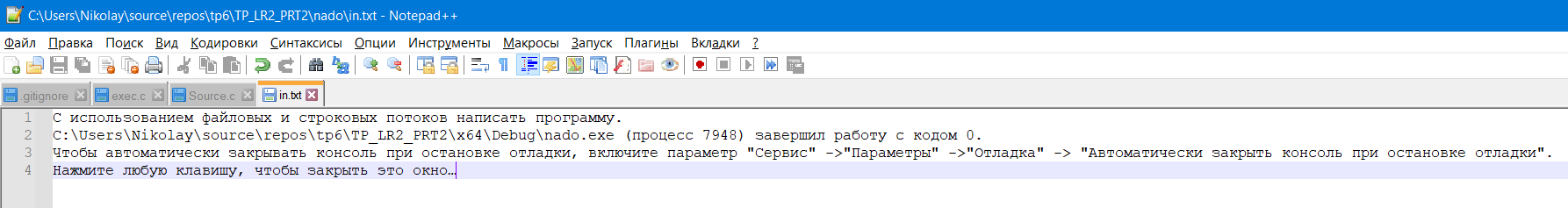


Рисунок 7. Скриншот текстового файла, программа 2, пример 1

1. **Выводы**

В процессе выполнения лабораторной работы мы применили навыки поученные в процессе выполнения предыдущих лабораторных работ. Изучили принципы написания программ на языке C++. Разработали программы, согласно указанным требованиям. Программа работает корректно, что видно из представленных результатов работы.