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 |
| РАБОТА С ПОТОКАМИ В С++ |
| по курсу: ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 1142 |  |  |  |  |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

1. **Постановка задачи**
2. Стандартные потоки

Определить класс с именем АЕRОFLОТ, содержащий следующие поля:

● название пункта назначения рейса;

● номер рейса;

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

● записи должны быть размещены в алфавитном порядке по названиям пунктов назначения;

● вывод на экран пунктов назначения и номеров рейсов, обслуживаемых самолетом, тип которого введен с клавиатуры;

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

1. Файловые и строковые потоки

С использованием файловых и строковых потоков написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке.

1. **Формализация задачи**

Программа написана в среде разработки Visual Studio 2022. Компилятор - Visual Studio 2022.

* 1. **CPP-файлы, входящие в программу:**

TP2.cpp

KeeperA.cpp

Aeroflot.cpp

KeeperT.cpp

Senten.cpp

* 1. **Файлы заголовков, входящие в программу:**

KeeperA.cpp

Aeroflot.cpp

KeeperT.cpp

Senten.cpp

* 1. **Описание работы программы**

В файле TP2.cpp реализовано пользовательское меню, из которого конечный пользователь может производить манипуляции над объектами классов, написанных для данной работы. Также в этом файле реализована очистка консоли для упрощения пользования программой.

Через класс KeeperA реализован контейнер-список для хранения элементов класса Aeroflot. Работа контейнера сделана при помощи метода push\_back(), который помещает все новые элементы в конец списка, и pop(), который удаляет элемент по выбранному номеру в контейнере. Функция Show() выводит на экран содержимое контейнера. Перед выводом данных на экран используется метод sort(), который сортирует содержимое контейнера в алфавитном порядке на основе сортировки пузырьком. Также есть поле filter, по которому выводимые рейсы фильтруются по типу самолёта.

В классе Aeroflot хранится информация об одном рейсе, а именно: путь, номер рейса и тип самолёта. Путь рейса используется для сравнения с другими рейсами во время сортировки KeeperA.

Класс KeeperT реализует контейнер для строк, которые программа берёт из файла 1.txt. Во время считывания текста из файла программа ищет начало предложения (заглавную букву) и записывает предложение, пока не наткнётся на один из разделительных знаков (точка, вопросительный знак и т.д.). После считывания предложений, программа выводит первые три в обратном порядке.

Класс Senten исполняет функцию строкового элемента контейнера KeeperT. Содержит в себе строковое поле, внутрь которого помещается предложение.

* 1. **Таблица классов**

|  |  |  |
| --- | --- | --- |
| Название класса | Назначение класса | Наследование |
| KeeperA | Контейнер-список для хранения данных о рейсах. | Отсутствует |
| Aeroflot | Элемент класса KeeperA. Хранит в себе информацию о пути, номере и самолёте рейса. | Отсутствует |
| KeeperT | Контейнер-список для хранения предложений, полученных из файла. | Отсутствует |
| Senten | Элемент класса KeeperT. Хранит в себе предложение, прочитанное из файла. | Отсутствует |

Таблица 1 – Таблица классов.

* 1. **Иерархия классов**

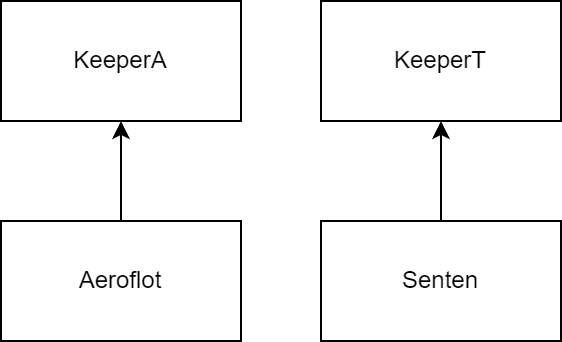
****

Рисунок 1 – Схема иерархии классов

1. **Исходный код**
   1. **TP2.cpp**

#include "KeeperA.h"

#include "KeeperT.h"

#include <list>

#include <iostream>

#define cles 1

using namespace std;

void menu();

void cls();

void men();

void disp();

void inc();

void imp();

void exp();

int main(void)

{

/\*string fir = "bbbbbbbbbbb";

string sec = "baaaaaaaaaa";

if (fir > sec)

cout << fir;

else

cout << sec;\*/

menu();

return 0;

}

void menu()

{

int choice = 0;

int k = 0;

int d = 0;

string filt;

cls();

KeeperA bob(0);

KeeperA aero(bob);

KeeperT texo(3);

filt = " ";

k = 0;

men();

//вывод в консоль

while (choice != 3)

{

cin >> choice;

cls();

switch (choice)

{

case 1: //////////////////////////////Open aeroflot

{

aero.push\_back();

k = 0;

cls();

disp();

aero.Show();

while (k != 5)

{

cin >> k;

switch (k)

{

case 1:////Add element

cls();

disp();

aero.push\_back();

cls();

disp();

aero.Show();

break;

case 2://///Remove element

cls();

disp();

cout << "What flight you'd like to remove?" << endl;

cin >> d;

aero.pop(d);

cls();

disp();

aero.Show();

break;

case 3:///Show all flights

cls();

disp();

aero.setfilt();

aero.Show();

break;

case 4:///Show flights flown by this type of plane

cls();

disp();

cout << "By what plane type would you like to filter the flights?" << endl;

cin >> filt;

aero.setfilt(filt);

cls();

disp();

aero.Show();

break;

case 5:///To main menu

cls();

men();

break;

default:

cls();

disp();

inc();

break;

}

}

break;

}

case 2: //////////////////////////////Output edited text from file

{

cls();

men();

texo.Show();

break;

}

case 3://////////////////////////////Exit

{

cls();

men();

break;

}

default:

{

cls();

men();

inc();

break;

}

}

}

}

void men()

{

cout << "Menu:\n(1)Aeroflot\n(2)Output text from file\n(3)Exit"<< endl;

}

void disp()

{

cout << "(1)Add element\n(2)Remove element\n(3)Show all flights\n(4)Filter by plane type\n(5)To menu" << endl << endl;

}

void cls()

{

if (cles)

system("cls");

}

void elad()

{

cout << "Element added!\n";

}

void elre()

{

cout << "Element removed!\n";

}

void inc()

{

cout << "Incorrect input!\n" << endl;

}

Код 1 – Листинг файла TP2.cpp

* 1. **KeeperA.cpp**

#include "KeeperA.h"

#include<iostream>

#include <conio.h>

#include <windows.h>

KeeperA::KeeperA()

{

Size = 0;

head = nullptr;

tail = nullptr;

filter = " ";

}

KeeperA::KeeperA(int siz)

{

Size = 0;

head = nullptr;

tail = nullptr;

filter = " ";

Spawn(siz);

}

KeeperA::~KeeperA()

{

clear();

/\*cout << "111" << endl;\*/

}

KeeperA::KeeperA(const KeeperA& other)

{

int cnt = 0;

this->Size = other.Size;

Aeroflot\* current = nullptr;

Aeroflot\* current1 = nullptr;

this->head = current;

filter = " ";

for (cnt = 0; cnt < this->Size; cnt++)

{

if (head == nullptr)

{

this->head = new Aeroflot(other.head->getdat());

this->tail = this->head;

}

else

{

current = other.head;

current1 = this->head;

while (current1->pNext != nullptr)

{

current = current->pNext;

current1 = current1->pNext;

}

current1->pNext = new Aeroflot(current->pNext->getdat(), current1->pNext, current1);

this->tail = current1->pNext;

}

}

}

void KeeperA::swap(int fir, int sec)

{

Aeroflot\* first = this->head;

Aeroflot\* second = this->head;

int cntf = 0;

int cnts = 0;

while (first != nullptr)

{

if (cntf == fir)

{

break;

}

first = first->pNext;

cntf++;

}

while (second != nullptr)

{

if (cnts == sec)

{

break;

}

second = second->pNext;

cnts++;

}

Aeroflot\* temp = new Aeroflot("temp");

temp->setdest(first->getdest());

temp->setdat(first->getdat());

temp->settype(first->gettype());

first->setdest(second->getdest());

first->setdat(second->getdat());

first->settype(second->gettype());

second->setdest(temp->getdest());

second->setdat(temp->getdat());

second->settype(temp->gettype());

delete temp;

}

void KeeperA::sort()

{

int i, j;

string first;

string second;

Aeroflot\* fir = this->head;

Aeroflot\* sec = fir->pNext;

for (i = 0; i < Size - 1; i++)

{

// Last i elements are already

// in place

fir = this->head;

sec = fir->pNext;

for (j = 0; j < Size - i - 1; j++)

{

first = fir->getdest();

second = sec->getdest();

if ((first > second))

this->swap(j, j + 1);

fir = fir->pNext;

sec = fir->pNext;

}

}

}

void KeeperA::pop\_front()

{

int si = this->Size;

if (si)//Фикс исключения 1

{

Aeroflot\* temp = head;

head = head->pNext;//Исключение 1: попытка чтения поля пустого элемента

delete temp;

Size--;

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperA::pop\_back()

{

int si = this->Size;

if (si)//Фикс исключения 2

{

Aeroflot\* temp = tail;

tail = tail->pPrev;//Исключение 2: попытка чтения поля пустого элемента

if (si > 1)

{

tail->pNext = nullptr;

delete temp;

Size--;

}

else

{

this->head = nullptr;

this->Size = 0;

}

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperA::pop(int d)

{

int si = this->Size;

int i = 0;

//if (si)

if (d<0 || d>si - 1)

{

cout << "Incorrect range!" << endl;

Sleep(1000);

}

else if (d == 0)

this->pop\_front();

else if (d == si - 1)

this->pop\_back();

else

{

Aeroflot\* temp = this->head;

Aeroflot\* tem1 = nullptr;

while (i != d)

{

tem1 = temp;

temp = temp->pNext;

temp->pPrev = tem1;

i++;

}

temp->pPrev->pNext = temp->pNext;

temp->pNext->pPrev = temp->pPrev;

delete temp;

Size--;

}

}

void KeeperA::push\_back(int data)

{

if (head == nullptr)

{

head = new Aeroflot(data);

tail = this->head;

}

else

{

Aeroflot\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Aeroflot(data, current->pNext, current);

tail = current->pNext;

}

Size++;

}

//void KeeperA::push\_front(int data)

//{

// if (head == nullptr)

// {

// head = new Aeroflot(data);

// tail = this->head;

// }

// else

// {

// Aeroflot\* current = this->head;

//

// current->pPrev = new Aeroflot(data, current, current->pPrev);

// head = current->pPrev;

// }

// Size++;

//}

//

//void KeeperA::push(int i, int d)

//{

// int si = this->Size;

// int cnt = 0;

// if (i<0 || i> si)

// {

// cout << "Incorrect range!" << endl;

// Sleep(1000);

// }

// else if (i == 0)

// this->push\_front(d);

// else if (i == si)

// this->push\_back(d);

// else

// {

//

// Aeroflot\* current = this->head;

// while (cnt != i - 1)

// {

// current = current->pNext;

// cnt++;

// }

// Aeroflot\* current1 = current->pNext;

// current->pNext = new Aeroflot(d, current->pNext, current);

// Size++;

// }

//

//}

void KeeperA::clear()

{

while (Size)

{

pop\_front();

}

}

void KeeperA::setfilt(string filter)

{

this->filter = filter;

}

int KeeperA::GetSize()

{

return Size;

}

int KeeperA::request\_strt()

{

int strt;

cout << "Input container length: ";

cin >> strt;

return strt;

}

void KeeperA::Spawn()

{

int strt = request\_strt();

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

{

push\_back(rand() % 100);

}

}

void KeeperA::Spawn(int siz)

{

int strt = siz;

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

{

push\_back(rand() % 100);

}

}

string KeeperA::operator[](const int index)

{

int counter = 0;

string ret;

Aeroflot\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

ret = current->getdest();

return ret;

}

current = current->pNext;

counter++;

}

}

void KeeperA::Show()

{

if (filter == " ")

cout << "Currently showing all flights" << endl << endl;

else

cout << "Currently showing flights serviced by the following plane type: "<< filter << endl << endl;

Aeroflot\* current = this->head;

string cur\_typ;

this->sort();

if (current != nullptr)

{

while (current->pNext != nullptr)

{

cur\_typ = current->gettype();

if (filter != " " && cur\_typ == filter)

cout<<\*current;

else if (filter == " ")

cout << \*current;

current = current->pNext;

}

cur\_typ = current->gettype();

if (filter != " " && cur\_typ == filter)

cout << \*current;

else if (filter == " ")

cout << \*current;

}

else

cout << "The list is empty!!!" << endl;

}

Код 2 – Листинг файла KeeperA.cpp

* 1. **Aeroflot.cpp**

#include "Aeroflot.h"

//#include <iostream>

//#include <fstream>

Aeroflot::Aeroflot(int data, Aeroflot\* pNext, Aeroflot\* pPrev)

{

this->data = data;

this->pNext = pNext;

this->pPrev = pPrev;

setdest();

settype();

//Aeroflot\* temp = this;

cin >> \*this;

}

Aeroflot::Aeroflot(string temp)

{

this->data = data;

this->pNext = pNext;

this->pPrev = pPrev;

setdest();

settype();

}

void Aeroflot::setdest(string dest)

{

this->destin = dest;

}

void Aeroflot::setdat(int dat)

{

this->data = dat;

}

void Aeroflot::settype(string typ)

{

this->type = typ;

}

string Aeroflot::getdest()

{

return this->destin;

}

int Aeroflot::getdat()

{

return this->data;

}

string Aeroflot::gettype()

{

return this->type;

}

void Aeroflot::display()

{

cout << destin << endl << data << endl << type << endl << endl;

}

istream& operator>>(std::istream& input, Aeroflot& aero)

{

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

string depart;

string destin;

cout << "\nPlace of departure: ";

getline(input, depart);

cout << "\nDestination: ";

getline(input, destin);

aero.setdest(depart + " - " + destin);

cout << "\nPlane type: ";

getline(input, aero.type);

/\*cout << "\nNumber of the flight: ";

input >> aero.data;\*/

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

return input;

}

ostream& operator<<(ostream& output, const Aeroflot& aero)

{

output << "Destination: " << aero.destin << endl<<"Flight number: "<<aero.data << endl <<"Plane type: "<< aero.type << endl << endl;

return output;

}

Код 3 – Листинг файла Aeroflot.cpp

* 1. **KeeperT.cpp**

#include "KeeperT.h"

#include<iostream>

#include <conio.h>

#include <windows.h>

class Anyof

{

public:

bool operator()(char\* arr, char symbol)

{

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == symbol) {

return true;

}

}

return false;

}

bool operator()(char symbol, char\* arr)

{

return anyof(arr, symbol);

}

bool operator()(std::string arr, char symbol)

{

for (int i = 0; arr[i] != '\0'; i++) {

if (arr[i] == symbol) {

return true;

}

}

return false;

}

bool operator()(char symbol, std::string arr)

{

return anyof(arr, symbol);

}

}anyof;

KeeperT::KeeperT()

{

Size = 0;

head = nullptr;

tail = nullptr;

}

KeeperT::KeeperT(int siz)

{

Size = 0;

head = nullptr;

tail = nullptr;

Spawn(siz);

int phase = 0;

ifstream fin("1.txt");

string str = "";

char separator[] = { '. ', '? ', '! ', '… ', '\0' };

char first\_let[] = {'A', 'B','C','D' ,'E' ,'F' ,'G' ,'H' ,'I' ,'J' ,'K' ,'L' ,'M' ,'N' ,'O' ,'P' ,'Q' ,'R' ,'S' ,'T' ,'U' ,'V' ,'W' ,'X' ,'Y' ,'Z' };

char symbol;

Senten\* current = this->head;

while (fin.get(symbol))

{

switch (phase)

{

case 0://Поиск заглавной

if (anyof(first\_let, symbol))

{

str += symbol;

phase = 1;

}

break;

case 1://Запись предложения

str += symbol;

if (anyof(separator, symbol))

{

if (current == nullptr)

{

phase = 99;

break;

}

current->settext(str);

current = current->pNext;

str = "";

phase = 0;

}

break;

case 99:

return;

}

}

}

KeeperT::~KeeperT()

{

clear();

/\*cout << "111" << endl;\*/

}

void KeeperT::pop\_front()

{

int si = this->Size;

if (si)//Фикс исключения 1

{

Senten\* temp = head;

head = head->pNext;//Исключение 1: попытка чтения поля пустого элемента

delete temp;

Size--;

}

else

{

cout << "Bad Input: cannot delete an element of an empty container" << endl;

Sleep(1000);

}

}

void KeeperT::push\_back(int data)

{

if (head == nullptr)

{

head = new Senten(data);

tail = this->head;

}

else

{

Senten\* current = this->head;

while (current->pNext != nullptr)

{

current = current->pNext;

}

current->pNext = new Senten(data, current->pNext, current);

tail = current->pNext;

}

Size++;

}

void KeeperT::clear()

{

while (Size)

{

pop\_front();

}

}

int KeeperT::GetSize()

{

return Size;

}

int KeeperT::request\_strt()

{

int strt;

cout << "Input container length: ";

cin >> strt;

return strt;

}

void KeeperT::Spawn()

{

int strt = request\_strt();

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

{

push\_back(rand() % 10);

}

}

void KeeperT::Spawn(int siz)

{

int strt = siz;

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

{

push\_back(rand() % 10);

}

}

string KeeperT::operator[](const int index)

{

int counter = 0;

string ret;

Senten\* current = this->head;

while (current != nullptr)

{

if (counter == index)

{

ret = current->gettext();

return ret;

}

current = current->pNext;

counter++;

}

}

void KeeperT::Show()

{

Senten\* current = this->tail;

cout << endl;

if (current->gettext()!="b")

cout <<current->gettext() << endl;//3

current = current->pPrev;

if (current->gettext() != "b")

cout << current->gettext() << endl;//2

current = current->pPrev;

if (current->gettext() != "b")

cout << current->gettext() << endl;//1

else

cout << "No sentences were found";

}

Код 4 – Листинг файла KeeperT.cpp

* 1. **Senten.cpp**

#include "Senten.h"

void Senten::setdat(int data)

{

this->data = data;

}

void Senten::settext(string text)

{

this->text = text;

}

string Senten::gettext()

{

return this->text;

}

Senten::Senten(int data, Senten\* pNext, Senten\* pPrev)

{

this->data = data;

this->pNext = pNext;

this->pPrev = pPrev;

this->text = "b";

}

Код 5 – Листинг файла Senten.cpp

* 1. **KeeperA.h**

#pragma once

#include "Aeroflot.h"

#include <iostream>

#include <string>

using namespace std;

class KeeperA

{

public:

KeeperA();

explicit KeeperA(int siz);

~KeeperA();

KeeperA(const KeeperA& other);

void swap(int fir, int sec);

void sort();

void pop\_front();

void pop\_back();

void pop(int d);

void push\_back(int data = rand() % 1000);

//void push\_front(int data = rand() % 1000);

//void push(int i = 0, int d = rand() % 1000);

void clear();

void setfilt(string filter = " ");

int GetSize();

int request\_strt();

void Spawn();

void Spawn(int siz);

string operator[](int index);

void Show();

private:

int Size;

string filter;

Aeroflot\* head;

Aeroflot\* tail;

};

Код 6 – Листинг файла KeeperA.h

* 1. **Aeroflot.h**

#pragma once

#include <iostream>

#include <string>

using namespace std;

class Aeroflot

{

public:

Aeroflot\* pNext;

Aeroflot\* pPrev;

Aeroflot(int data = int(), Aeroflot\* pNext = nullptr, Aeroflot\* pPrev = nullptr);

Aeroflot(string temp);

void setdest(string dest="a - b");

void setdat(int dat = 0);

void settype(string typ="plane");

string getdest();

int getdat();

string gettype();

void display();

friend std::istream& operator>>(std::istream& input, Aeroflot& aero);

friend std::ostream& operator<<(std::ostream& output, const Aeroflot& aero);

private:

string destin;

int data;

string type;

};

Код 7 – Листинг файла Aeroflot.h

* 1. **KeeperT.h**

#pragma once

#include <iostream>

#include <string>

#include "Senten.h"

using namespace std;

class KeeperT

{

public:

KeeperT();

explicit KeeperT(int siz);

~KeeperT();

void pop\_front();

void push\_back(int data = rand() % 10);

void clear();

int GetSize();

int request\_strt();

void Spawn();

void Spawn(int siz);

string operator[](const int index);

void Show();

private:

int Size;

Senten\* head;

Senten\* tail;

};

Код 8 – Листинг файла KeeperT.h

* 1. **Senten.h**

#pragma once

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

class Senten

{

public:

Senten\* pNext;

Senten\* pPrev;

void setdat(int data = 0);

void settext(string text);

string gettext();

Senten(int data = int(), Senten\* pNext = nullptr, Senten\* pPrev = nullptr);

private:

int data;

string text;

};

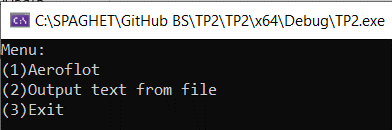
Код 9 – Листинг файла Senten.h

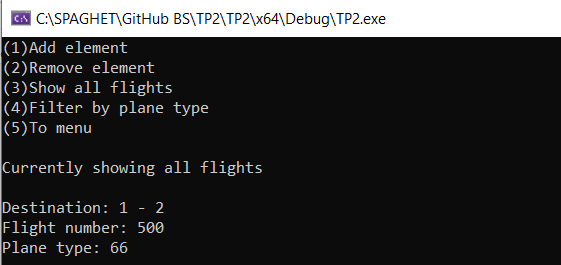
* 1. **GitHub**

Ссылка на репозиторий GitHub: https://github.com/Look0Me/TP2

1. **Результаты работы программы**

Ниже приведены скриншоты с примерами работы программы. Чтобы проверить все функциональный возможности программы, пройдемся по каждой возможной ветке работы программы.



Рисунок 2 – Главное меню

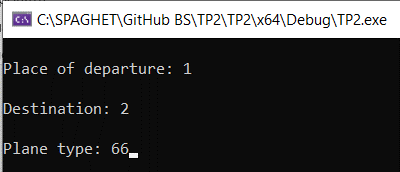


Рисунок 3 – Открытие меню Aeroflot просит пользователя ввести данный первого рейса

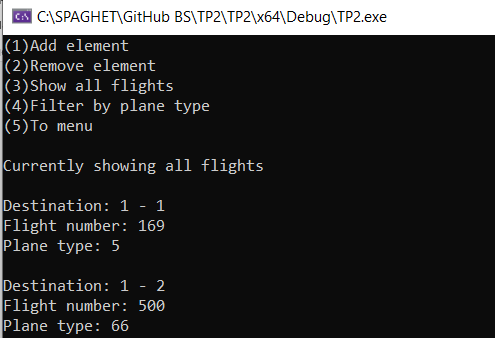


Рисунок 4 – Добавление нового рейса. Причём новая запись выводится перед старой, потому что идёт сортировка по полю Destination.

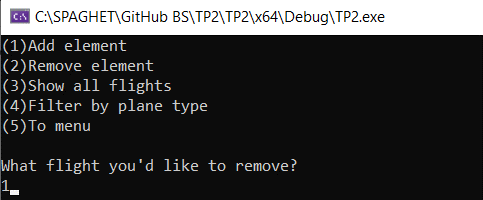
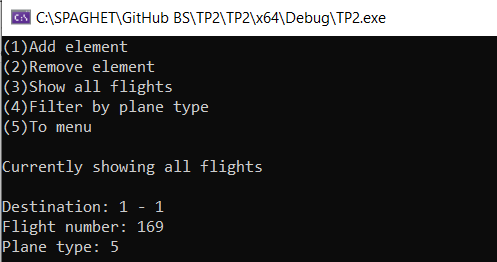


Рисунок 5 – Удаление рейса

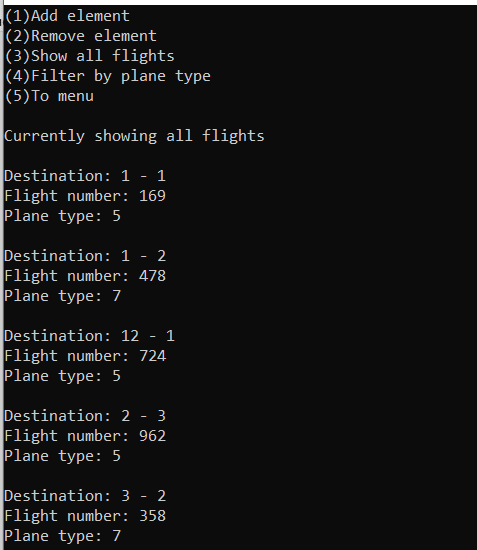
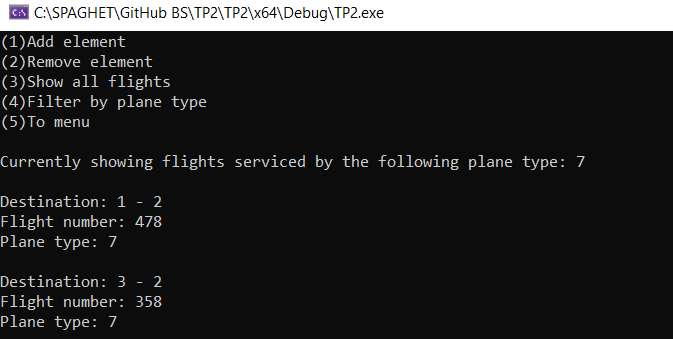


Рисунок 6 – Фильтрация рейсов по типу самолёта

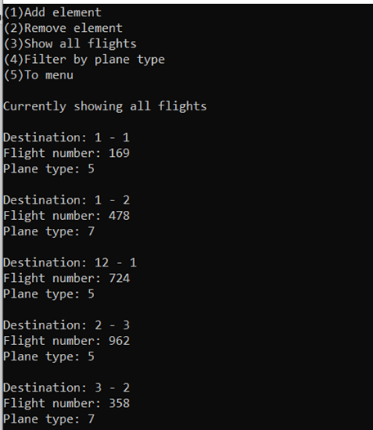
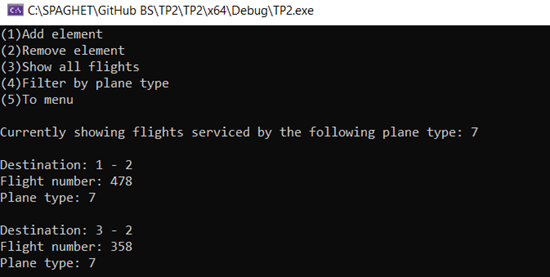
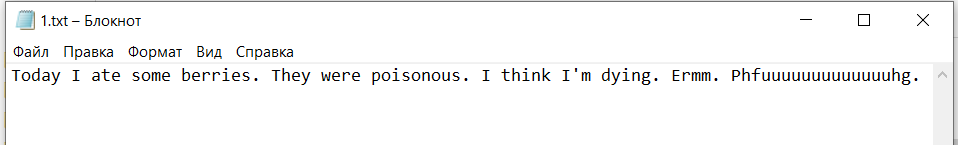


Рисунок 7 – Отключение фильтрации



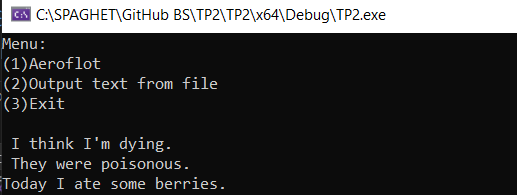


Рисунок 8 – Удаление элемента дека

1. **Вывод**

В ходе выполнения лабораторной работы был создан Aeroflot, описывающий структуру для хранения информации о рейсе;

Были перегружены операторы извлечения и вставки.

Было реализовано пользовательское меню, код программы разделен на заголовочные файлы и файлы, содержащие код.

В итоге тестирования программы выяснилось, что все функции работают исправно, что говорит о корректном написании кода.

По итогу выполнению лабораторной работы были закреплены полученные знания о стандартных, строковых и файловых потоках в языке С++.