**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И.Раззакова**

**ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №2**

Выполнил: студент группы ПИ-3-21

Жайлообек уулу Улукбек

Проверил: Мусабаев Э.Б.

**Бишкек 2023**

# **Лабораторная работа №2**

**Постановка задачи:**

**Тема: Классы и объекты**

**Задания:**

1. В здании аэропорта на экранах отображается информация о самолетах, а именно: информация о пункте отправления, пункте назначения, номере рейса, времени прибытия, времени отправления, номере секции для регистрации. Экраны – это средство, которое помогает своевременно зарегистрировать и отправить пассажиров. Важно, чтобы информация на экранах была понятной и верной.

Создайте необходимую информацию в виде таблицы для такого экрана, с помощью класса Aeroflot, содержащего в описании следующие поля: номер рейса; название пункта отправления; название пункта назначения рейса; время прибытия; время отправления; место регистрации.

Напишите код программы, выполняющей следующие действия: ввод с клавиатуры значений полей объектов; сортировку записей в таблице в алфавитном порядке по названию пунктов назначения; вывод на консоль значений полей объектов класса в виде таблицы рейсов; если таких рейсов нет, выдать соответствующее сообщение.

**Исходный код программы на C++:**

#include <iostream>

#include <string>

#include <algorithm>

#include <iomanip>

using namespace std;

class Aeroflot {

public:

int flightNumber; //nomer reisa

string departure\_point;// mesto otpravleniya

string destination\_point;//mesto naznacheniya

string arrival\_time;

string departure\_time;

int registration\_desk;

Aeroflot() {}

Aeroflot(int flightNumber, string departure\_point, string destination\_point, string arrival\_time, string departure\_time, int registration\_desk) {

this->flightNumber = flightNumber;

this->departure\_point = departure\_point;

this->destination\_point = destination\_point;

this->arrival\_time = arrival\_time;

this->departure\_time = departure\_time;

this->registration\_desk = registration\_desk;

}

//В общем, эти конструкторы позволяют создавать экземпляры класса "Aeroflot" либо без начальных значений, либо с конкретными значениями для атрибутов полета.

};\

bool compareByDestination(const Aeroflot& a, const Aeroflot& b) {

return a.destination\_point < b.destination\_point;

}

//а использована, для сортировки списка объектов класса Aeroflot по их пунктам назначения.В таком случае, она будет передана в функцию сортировки(например, std::sort), которая будет использовать её для сравнения объектов и упорядочивания их по заданному критерию.

int main() {

int n;

cout << "Введите количество рейсов: ";

cin >> n;

Aeroflot\* flights = new Aeroflot[n];

// Ввод данных с клавиатуры

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

cout << "Введите данные для рейса #" << i + 1 << ":\n";

cout << "Номер рейса: ";

cin >> flights[i].flightNumber;

cout << "Пункт отправления: ";

cin >> flights[i].departure\_point;

cout << "Пункт назначения: ";

cin >> flights[i].destination\_point;

cout << "Время прибытия: ";

cin >> flights[i].arrival\_time;

cout << "Время отправления: ";

cin >> flights[i].departure\_time;

cout << "Номер секции для регистрации: ";

cin >> flights[i].registration\_desk;

}

// Сортировка рейсов по пункту назначения

sort(flights, flights + n, compareByDestination);

// Вывод данных

cout << "\nТаблица рейсов:\n";

cout << "------------------------------------------------------------------------\n";

cout << "| Номер | Отправление | Назначение | Прибытие | Отправление | Секция |\n";

cout << "------------------------------------------------------------------------\n";

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

cout << "| " << flights[i].flightNumber << " | " << flights[i].departure\_point << " | " << flights[i].destination\_point << " | " << flights[i].arrival\_time << " | " << flights[i].departure\_time << " | " << flights[i].registration\_desk << " |\n";

}

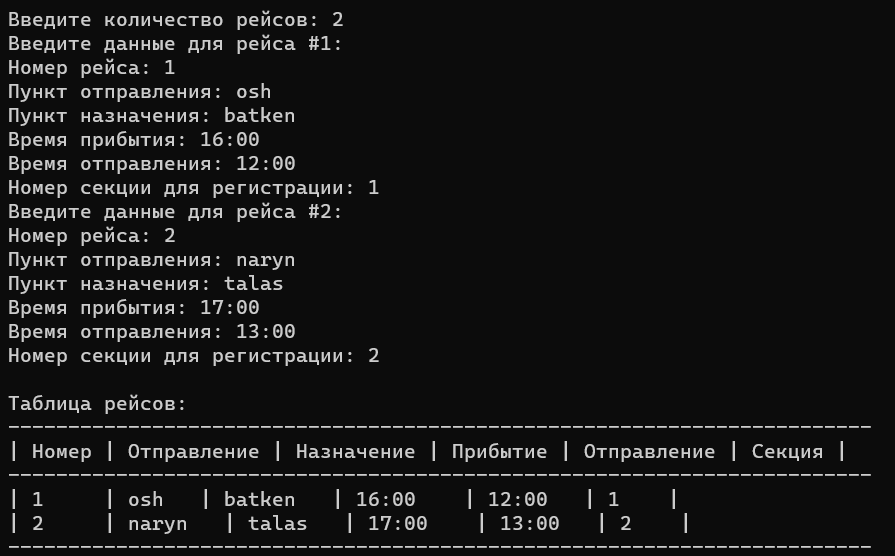
cout << "------------------------------------------------------------------------\n";

delete[] flights;

return 0;

}

Тест:



**2. Опишите класс «студенческая группа».**

Предусмотрите возможность: работы с переменным числом студентов; поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона); добавления или удаления записей; сортировки по разным полям.

Разработайте программу, демонстрирующую работу с этим классом.

Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

**Исходный код программы на C++:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

class Student {

public:

string lastName;

string firstName;

string dateOfBirth;

string phoneNumber;

Student(string last, string first, string dob, string phone)

: lastName(last), firstName(first), dateOfBirth(dob), phoneNumber(phone) {}

};

class StudentGroup {

private:

vector<Student> students;

public:

void addStudent(const Student& student) {

students.push\_back(student);

}

void removeStudent(const string& lastName) {

students.erase(

remove\_if(students.begin(), students.end(),

[&lastName](const Student& student) { return student.lastName == lastName; }),

students.end());

}

Student\* findStudentByLastName(const string& lastName) {

auto it = find\_if(students.begin(), students.end(),

[&lastName](const Student& student) { return student.lastName == lastName; });

if (it != students.end()) {

return &(\*it);

}

return nullptr;

}

void sortStudentsByLastName() {

sort(students.begin(), students.end(),

[](const Student& a, const Student& b) { return a.lastName < b.lastName; });

}

void sortStudentsByFirstName() {

sort(students.begin(), students.end(),

[](const Student& a, const Student& b) { return a.firstName < b.firstName; });

}

void printAllStudents() const {

cout << "Список студентов:\n";

for (const auto& student : students) {

cout << "Фамилия: " << student.lastName << ", Имя: " << student.firstName

<< ", Дата рождения: " << student.dateOfBirth << ", Телефон: " << student.phoneNumber << endl;

}

}

};

int main() {

StudentGroup group;

// Добавление студентов

group.addStudent(Student("Жайлообекуулу", "Улукбек", "21.03.2003", "0500566230"));

group.addStudent(Student("Алтынбеков", "Даурен", "27.11.2003", "0500566240"));

group.addStudent(Student("Саркобенов", "Арстан", "12.03.2003", "0500566250"));

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Добавить студента\n";

cout << "2. Поиск студента по фамилии\n";

cout << "3. Удаление студента\n";

cout << "4. Сортировка студентов по фамилии\n";

cout << "5. Сортировка студентов по имени\n";

cout << "6. Вывод всех студентов\n";

cout << "0. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

string lastName, firstName, dob, phone;

cout << "Введите данные нового студента:\n";

cout << "Фамилия: ";

cin >> lastName;

cout << "Имя: ";

cin >> firstName;

cout << "Дата рождения (ДД.ММ.ГГГГ): ";

cin >> dob;

cout << "Телефон: ";

cin >> phone;

group.addStudent(Student(lastName, firstName, dob, phone));

cout << "Студент добавлен.\n";

break;

}

case 2: {

string lastName;

cout << "Введите фамилию студента: ";

cin >> lastName;

Student\* foundStudent = group.findStudentByLastName(lastName);

if (foundStudent) {

cout << "Студент найден:\n";

cout << "Фамилия: " << foundStudent->lastName << ", Имя: " << foundStudent->firstName

<< ", Дата рождения: " << foundStudent->dateOfBirth << ", Телефон: " << foundStudent->phoneNumber << endl;

}

else {

cout << "Студент с фамилией " << lastName << " не найден.\n";

}

break;

}

case 3: {

string lastName;

cout << "Введите фамилию студента для удаления: ";

cin >> lastName;

group.removeStudent(lastName);

cout << "Студент с фамилией " << lastName << " удален.\n";

break;

}

case 4:

group.sortStudentsByLastName();

cout << "Студенты отсортированы по фамилии.\n";

break;

case 5:

group.sortStudentsByFirstName();

cout << "Студенты отсортированы по имени.\n";

break;

case 6:

group.printAllStudents();

break;

case 0:

cout << "Выход из программы.\n";

break;

default:

cout << "Некорректный выбор.\n";

}

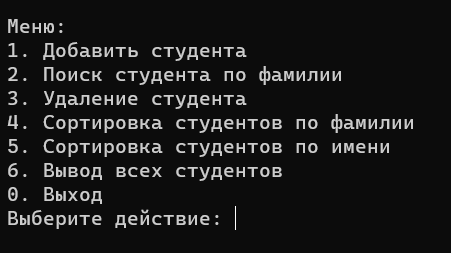
} while (choice != 0);

return 0;

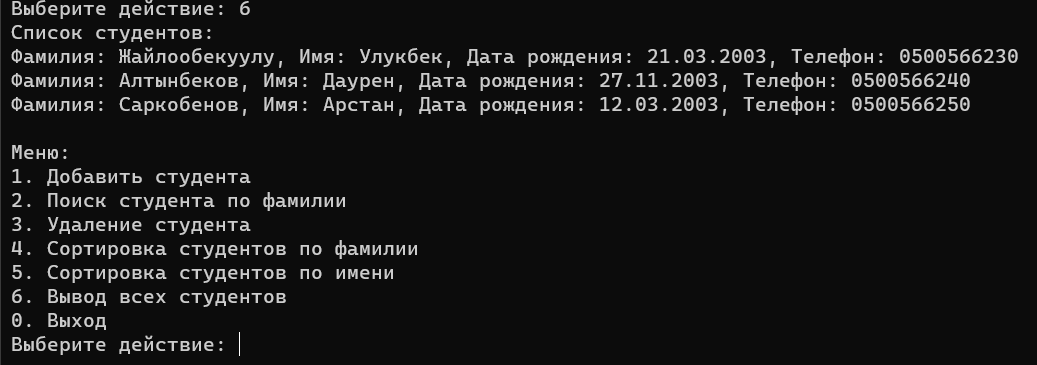
}

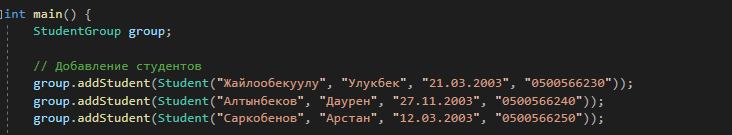
Тесты:

Меню:

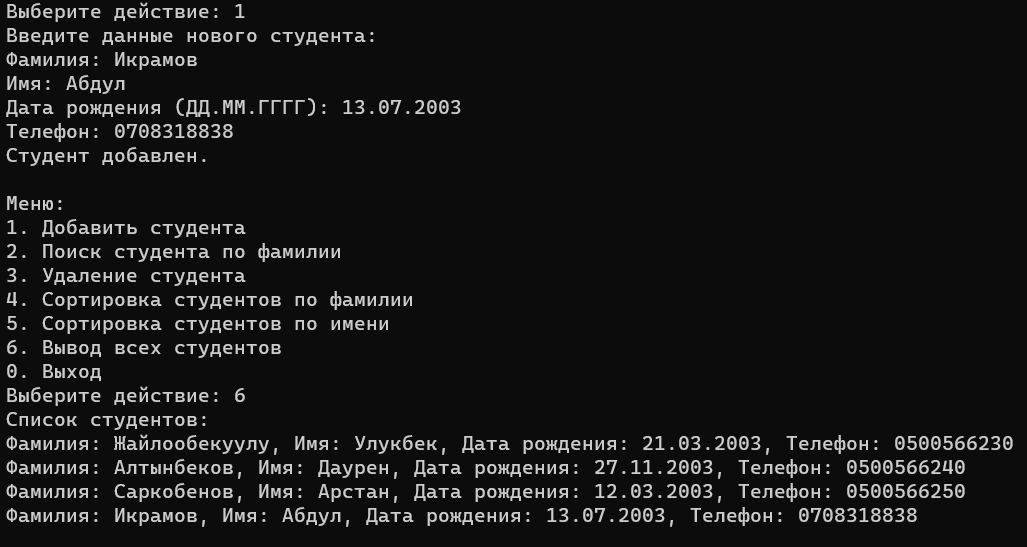


**Вывод студентов(6):**

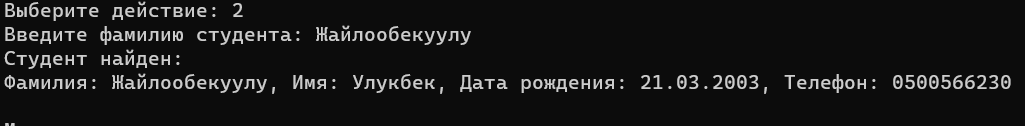




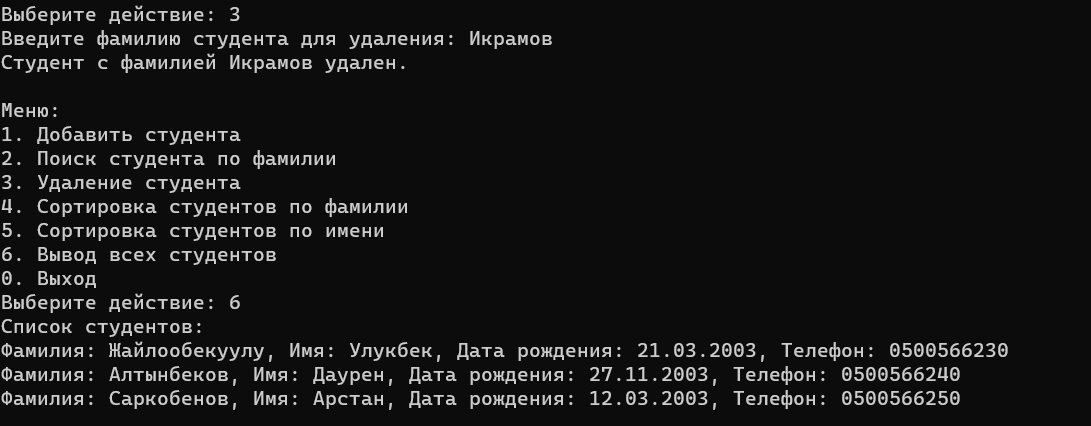
**Добавление студента(1)**



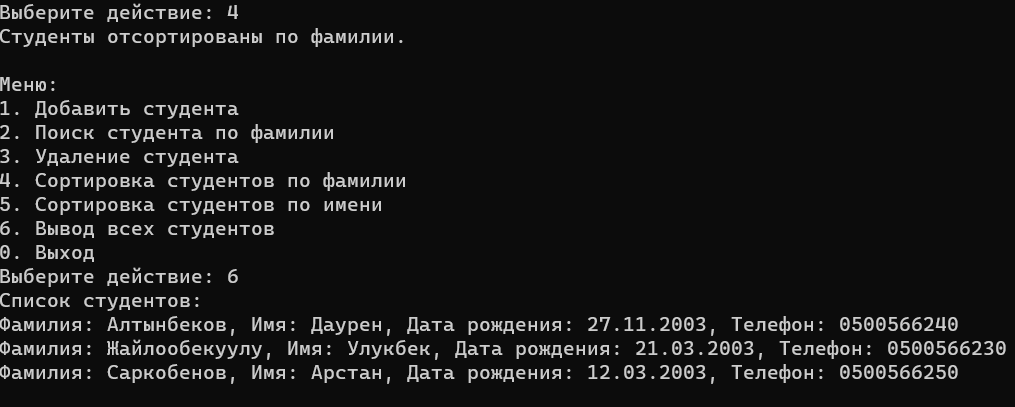
**Поиск студента по фамилии(2)**



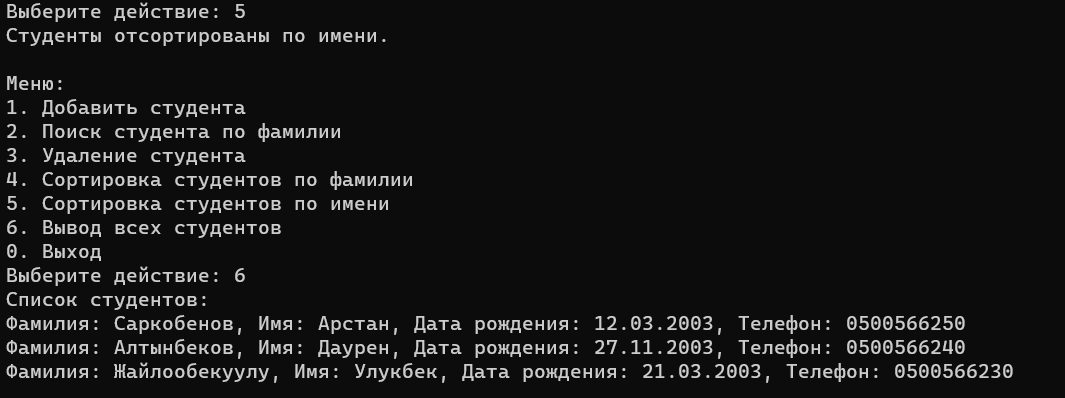
**Удаление студента(3)**



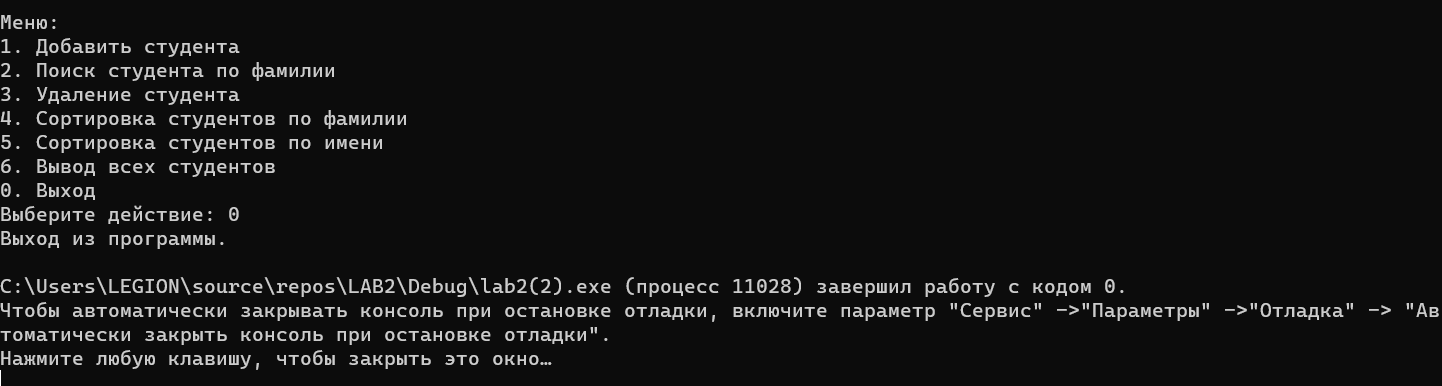
**Сортировка по фамилии(4)**



Сортировка по имени(5)



**Выход(0)**



**3. Опишите класс, реализующий стек (Stack).**

Разработать программу, использующую этот класс для моделирования Т-образного сортировочного узла на железной дороге. Программа должна разделять на два направления состав, состоящий из вагонов двух типов (на каждое направление формируется состав из вагонов одного типа). Предусмотреть возможность формирования состава из файла или с клавиатуры. Возможно использование стандартных функций при работе со стеком в виде контейнера из библиотеки STL:

**push()** - поместить элемент в вершину стека;

**pop()** - удалить элемент из вершины стека;

**Исходный код программы на C++:**

# #include <iostream>

#include <fstream>

#include <stack>

#include <string>

#include <vector>

using namespace std;

class Student {

public:

string type; // тип вагона

int number; // номер вагона

Student(string t, int n) : type(t), number(n) {}

};

class TrainSortingNode {

private:

stack<Student> leftDirection; // Вагоны направляющиеся влево

stack<Student> rightDirection; // Вагоны направляющиеся вправо

public:

void addStudentToDirection(const Student& student, const string& direction) {

if (direction == "left") {

leftDirection.push(student);

}

else if (direction == "right") {

rightDirection.push(student);

}

else {

cout << "Некорректное направление.\n";

}

}

void displayTrains() {

cout << "Составы по направлениям:\n";

cout << "Влево:\n";

displayStack(leftDirection);

cout << "Вправо:\n";

displayStack(rightDirection);

}

void displayStack(stack<Student>& s) {

stack<Student> temp = s;

while (!temp.empty()) {

cout << "Тип: " << temp.top().type << ", Номер: " << temp.top().number << endl;

temp.pop();

}

cout << endl;

}

void loadTrainsFromFile(const string& filename) {

ifstream file(filename);

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

cout << "Ошибка открытия файла.\n";

return;

}

string type;

int number;

string direction;

while (file >> type >> number >> direction) {

addStudentToDirection(Student(type, number), direction);

}

file.close();

}

void loadTrainsFromConsole() {

int n;

cout << "Введите количество вагонов: ";

cin >> n;

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

string type, direction;

int number;

cout << "Введите данные для вагона #" << i + 1 << ":\n";

cout << "Тип (грузовой/пассажирский): ";

cin >> type;

cout << "Номер: ";

cin >> number;

cout << "Направление (left/right): ";

cin >> direction;

addStudentToDirection(Student(type, number), direction);

}

}

};

int main() {

TrainSortingNode sortingNode;

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Загрузить составы из файла\n";

cout << "2. Загрузить составы с клавиатуры\n";

cout << "3. Показать составы\n";

cout << "0. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

sortingNode.loadTrainsFromFile("C:/Users/LEGION/OneDrive/Рабочий стол/uluk.txt");

cout << "Составы загружены из файла.\n";

break;

}

case 2:

sortingNode.loadTrainsFromConsole();

cout << "Составы загружены с клавиатуры.\n";

break;

case 3:

sortingNode.displayTrains();

break;

case 0:

cout << "Выход из программы.\n";

break;

default:

cout << "Некорректный выбор.\n";

}

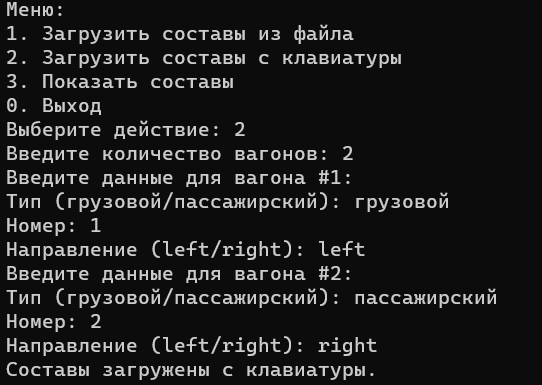
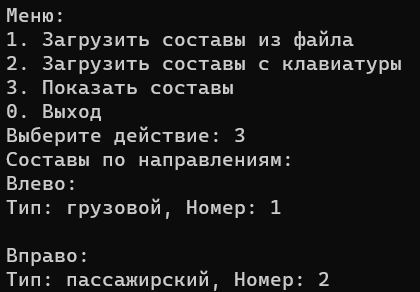
} while (choice != 0);

return 0;

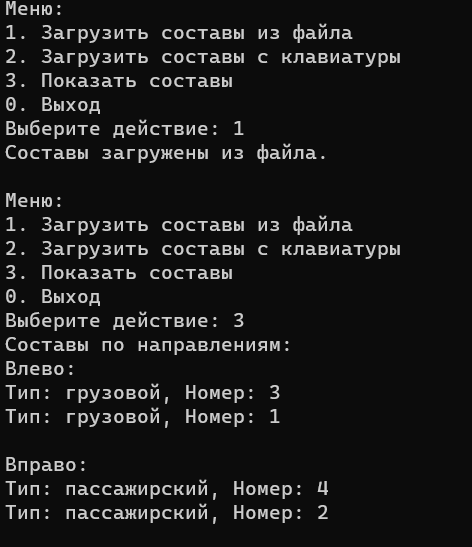
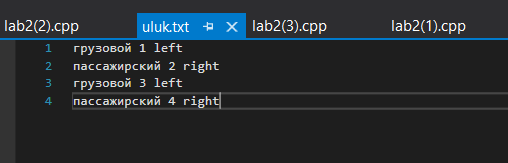
}

Тесты:

***Загрузка составов интерактивным способом(ввод с клавиатуры)***

**Загрузка составов из файла(ввод данных из файла)**

**** ****