**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

**КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. И. РАЗЗАКОВА**

**Кафедра «Программное обеспечение компьютерных систем»**

**Отчет**

**ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ С++**

**Лабораторная работа №2**

**Выполнила: Эмилбекова Элиза ПИ-2-21**

**Проверил: Мусабаев Э.Б.**

**Бишкек 2024**

**Задания**

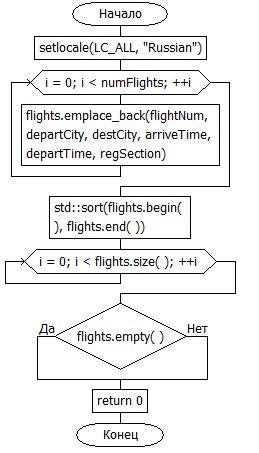
1. В здании аэропорта на экранах отображается информация о самолетах, а именно: информация о пункте отправления, пункте назначения, номере рейса, времени прибытия, времени отправления, номере секции для регистрации. Экраны – это средство, которое помогает своевременно зарегистрировать и отправить пассажиров.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

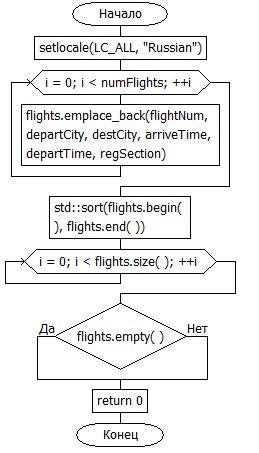
Создайте необходимую информацию в виде таблицы для такого экрана, с помощью класса Aeroflot, содержащего в описании следующие поля: номер рейса; название пункта отправления; название пункта назначения рейса; время прибытия; время отправления; место регистрации.

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.

**Решение:**

**Блок -схема:**

****

****

**Код:**

#include <iostream>

#include <vector>

#include <algorithm>

class Aeroflot {

public:

std::string flightNumber;

std::string departureCity;

std::string destinationCity;

std::string arrivalTime;

std::string departureTime;

int registrationSection;

// Конструктор класса

Aeroflot(std::string flightNum, std::string departCity, std::string destCity,

std::string arriveTime, std::string departTime, int regSection)

: flightNumber(flightNum), departureCity(departCity), destinationCity(destCity),

arrivalTime(arriveTime), departureTime(departTime), registrationSection(regSection) {}

// Перегрузка оператора < для сортировки по названию пункта назначения

bool operator<(const Aeroflot& other) const {

return destinationCity < other.destinationCity;

}

};

int main() {

setlocale(LC\_ALL, "Russian");

std::vector<Aeroflot> flights; // Вектор для хранения объектов класса Aeroflot

// Ввод данных с клавиатуры

int numFlights;

std::cout << "Введите количество рейсов: ";

std::cin >> numFlights;

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

std::string flightNum, departCity, destCity, arriveTime, departTime;

int regSection;

std::cout << "Введите информацию о рейсе #" << i + 1 << ":\n";

std::cout << "Номер рейса: ";

std::cin >> flightNum;

std::cout << "Пункт отправления: ";

std::cin >> departCity;

std::cout << "Пункт назначения: ";

std::cin >> destCity;

std::cout << "Время прибытия: ";

std::cin >> arriveTime;

std::cout << "Время отправления: ";

std::cin >> departTime;

std::cout << "Номер секции для регистрации: ";

std::cin >> regSection;

flights.emplace\_back(flightNum, departCity, destCity, arriveTime, departTime, regSection);

}

// Сортировка по названию пункта назначения

std::sort(flights.begin(), flights.end());

// Вывод на консоль в виде таблицы

std::cout << "\nТаблица рейсов:\n";

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

std::cout << "| № | Номер рейса | Пункт отправления | Пункт назначения | Время прибытия | Время отправления | Секция регистрации |\n";

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

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

std::cout << "| " << i + 1 << " | " << flights[i].flightNumber << " | " << flights[i].departureCity

<< " | " << flights[i].destinationCity << " | " << flights[i].arrivalTime << " | "

<< flights[i].departureTime << " | " << flights[i].registrationSection << " |\n";

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

}

// Вывод сообщения, если нет рейсов

if (flights.empty()) {

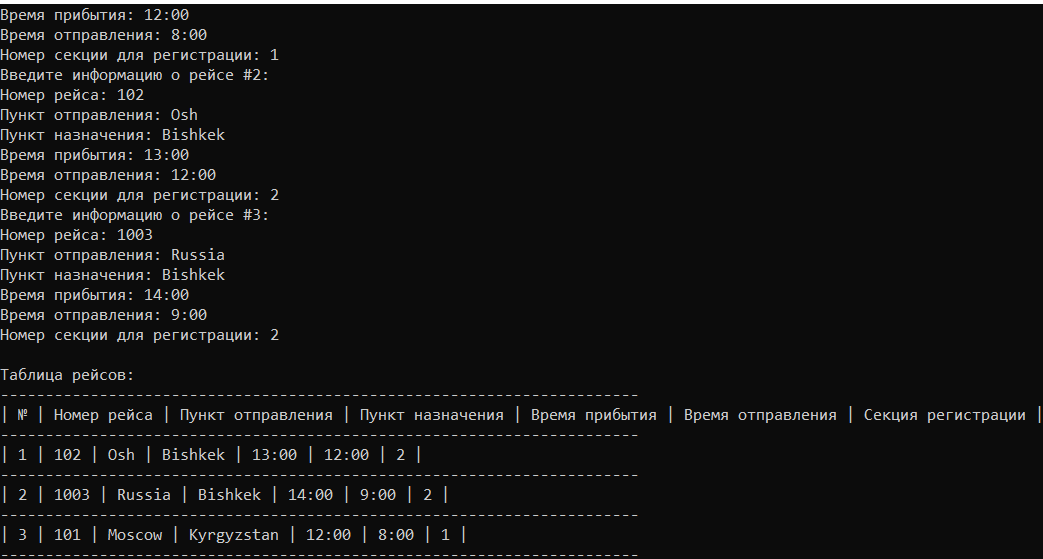
std::cout << "Нет доступных рейсов.\n";

}

return 0;

}

**Работа программы :**



**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.

**Код:**

int main() {

setlocale(LC\_ALL, "Russian");

StudentGroup group;

int choice;

do {

std::cout << "\nМеню:\n";

std::cout << "1. Добавить студента\n";

std::cout << "2. Удалить студента\n";

std::cout << "3. Найти студента по фамилии\n";

std::cout << "4. Сортировать студентов по имени\n";

std::cout << "5. Вывести список студентов\n";

std::cout << "0. Выйти\n";

std::cout << "Выберите действие: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string firstName, lastName, dateOfBirth, phoneNumber;

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

std::cout << "Имя: ";

std::cin >> firstName;

std::cout << "Фамилия: ";

std::cin >> lastName;

std::cout << "Дата рождения: ";

std::cin >> dateOfBirth;

std::cout << "Номер телефона: ";

std::cin >> phoneNumber;

group.addStudent(Student(firstName, lastName, dateOfBirth, phoneNumber));

break;

}

case 2: {

std::string lastName;

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

std::cin >> lastName;

group.removeStudent(lastName);

break;

}

case 3: {

std::string lastName;

std::cout << "Введите фамилию студента для поиска: ";

std::cin >> lastName;

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

if (foundStudent) {

std::cout << "Найден студент:\n";

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

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

}

else {

std::cout << "Студент не найден.\n";

}

break;

}

case 4:

group.sortStudentsByName();

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

break;

case 5:

group.displayStudents();

break;

case 0:

std::cout << "Выход из программы.\n";

break;

default:

std::cout << "Некорректный ввод. Попробуйте еще раз.\n";

break;

}

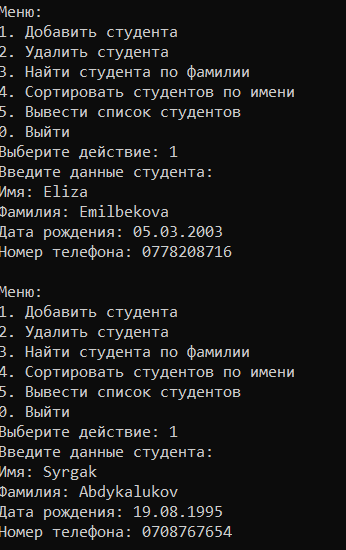
} while (choice != 0);

return 0;

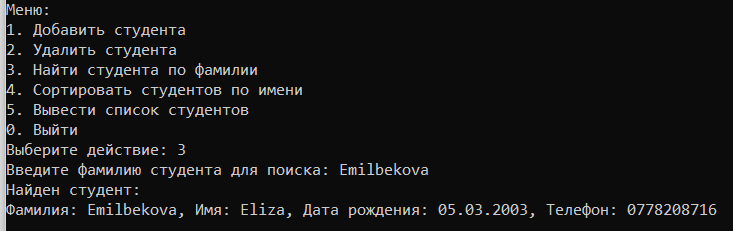
}

**Работа программы :**

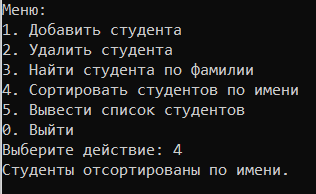
Добавление студента :

****

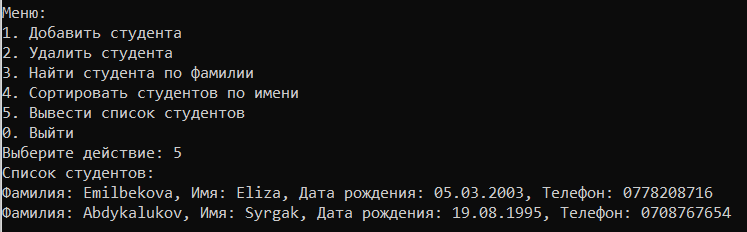
**Поиск студента:**



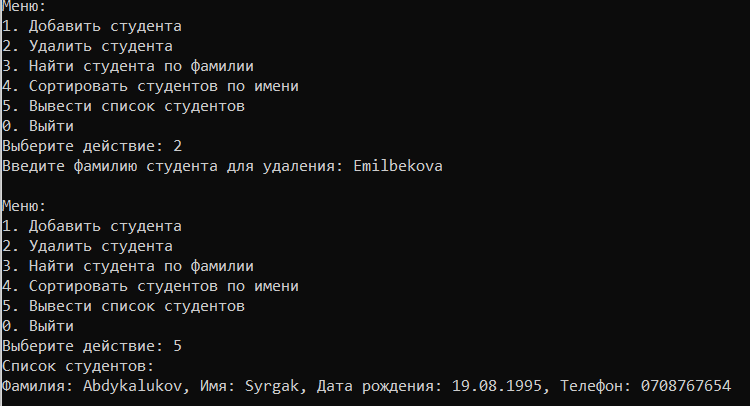
Сортировка студентов по фамилии:



Вывод список:



Удаляем студента и выводим список для проверки



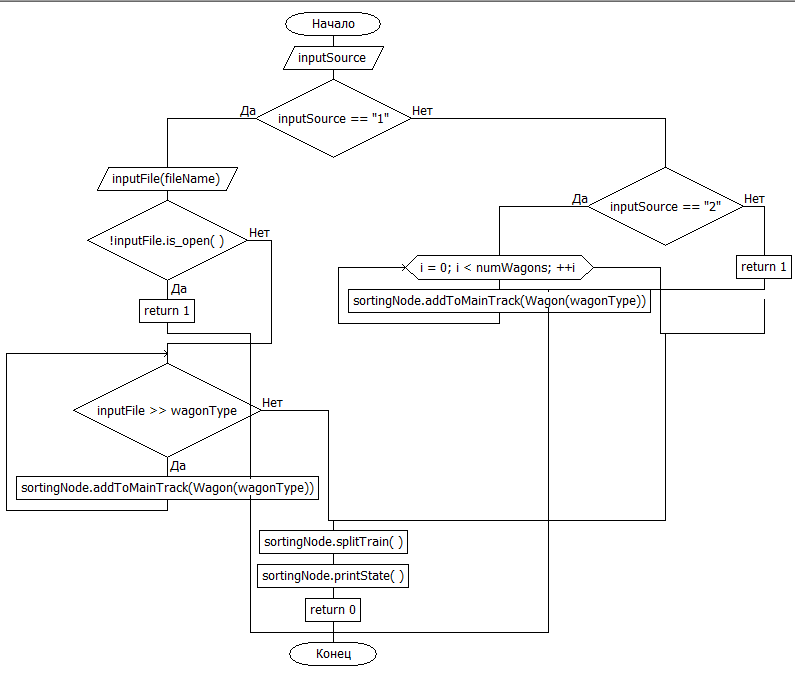
**3. Опишите класс, реализующий стек (Stack).**

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;

**Блок-схема:**

****

**Код:**

#include <iostream>

#include <fstream>

#include <stack>

#include <vector>

class Wagon {

public:

int type; // Тип вагона: 1 или 2

Wagon(int wagonType) : type(wagonType) {}

};

class SortingNode {

private:

std::stack<Wagon> mainTrack;

std::stack<Wagon> sideTrack1;

std::stack<Wagon> sideTrack2;

public:

// Добавить вагон на главный путь

void addToMainTrack(const Wagon& wagon) {

mainTrack.push(wagon);

}

// Разделить состав на два направления

void splitTrain() {

while (!mainTrack.empty()) {

Wagon wagon = mainTrack.top();

mainTrack.pop();

if (wagon.type == 1) {

sideTrack1.push(wagon);

}

else {

sideTrack2.push(wagon);

}

}

}

// Вывести состояние узла

void printState() {

std::cout << "Главный путь: ";

printStack(mainTrack);

std::cout << "Побочный путь 1: ";

printStack(sideTrack1);

std::cout << "Побочный путь 2: ";

printStack(sideTrack2);

}

private:

// Вспомогательная функция для вывода стека

void printStack(const std::stack<Wagon>& track) {

std::stack<Wagon> temp = track;

while (!temp.empty()) {

std::cout << temp.top().type << " ";

temp.pop();

}

std::cout << "\n";

}

};

int main() {

setlocale(LC\_ALL, "Russian");

SortingNode sortingNode;

// Чтение состава из файла или ввод с клавиатуры

std::string inputSource;

std::cout << "Выберите источник данных (файл(1)/клавиатура(2)): ";

std::cin >> inputSource;

if (inputSource == "1") {

std::string fileName;

std::cout << "Введите имя файла: ";

std::cin >> fileName;

std::ifstream inputFile(fileName);

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

std::cerr << "Ошибка открытия файла.\n";

return 1;

}

int wagonType;

while (inputFile >> wagonType) {

sortingNode.addToMainTrack(Wagon(wagonType));

}

}

else if (inputSource == "2") {

int numWagons;

std::cout << "Введите количество вагонов: ";

std::cin >> numWagons;

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

int wagonType;

std::cout << "Введите тип вагона (1 или 2) для вагона #" << i + 1 << ": ";

std::cin >> wagonType;

sortingNode.addToMainTrack(Wagon(wagonType));

}

}

else {

std::cerr << "Некорректный выбор источника данных.\n";

return 1;

}

// Разделение состава на два направления

sortingNode.splitTrain();

// Вывод состояния узла

sortingNode.printState();

return 0;

**Работа программы :**

