**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: Абакиров Тологон 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.

**Код:**

#include <iostream>

#include <vector>

#include <algorithm>

class Aeroflot {

public:

int flightNumber;

std::string departureCity;

std::string destinationCity;

std::string arrivalTime;

std::string departureTime;

std::string registrationSection;

Aeroflot(int number, const std::string& departure, const std::string& destination,

const std::string& arrival, const std::string& departureT, const std::string& registration)

: flightNumber(number), departureCity(departure), destinationCity(destination),

arrivalTime(arrival), departureTime(departureT), registrationSection(registration) {}

// 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;

// Ввод данных с клавиатуры

int numberOfFlights;

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

std::cin >> numberOfFlights;

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

int number;

std::string departure, destination, arrival, departureTime, registration;

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

std::cout << "Номер рейса: ";

std::cin >> number;

std::cout << "Пункт отправления: ";

std::cin >> departure;

std::cout << "Пункт назначения: ";

std::cin >> destination;

std::cout << "Время прибытия: ";

std::cin >> arrival;

std::cout << "Время отправления: ";

std::cin >> departureTime;

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

std::cin >> registration;

flights.emplace\_back(number, departure, destination, arrival, departureTime, registration);

}

// Сортировка по названию пункта назначения

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

// Вывод на консоль

std::cout << "\nТаблица рейсов:\n";

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

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

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

if (flights.empty()) {

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

}

else {

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

const Aeroflot& flight = flights[i];

std::cout << "| " << i + 1 << " | " << flight.departureCity << " | " << flight.destinationCity

<< " | " << flight.arrivalTime << " | " << flight.departureTime << " | "

<< flight.registrationSection << " |\n";

}

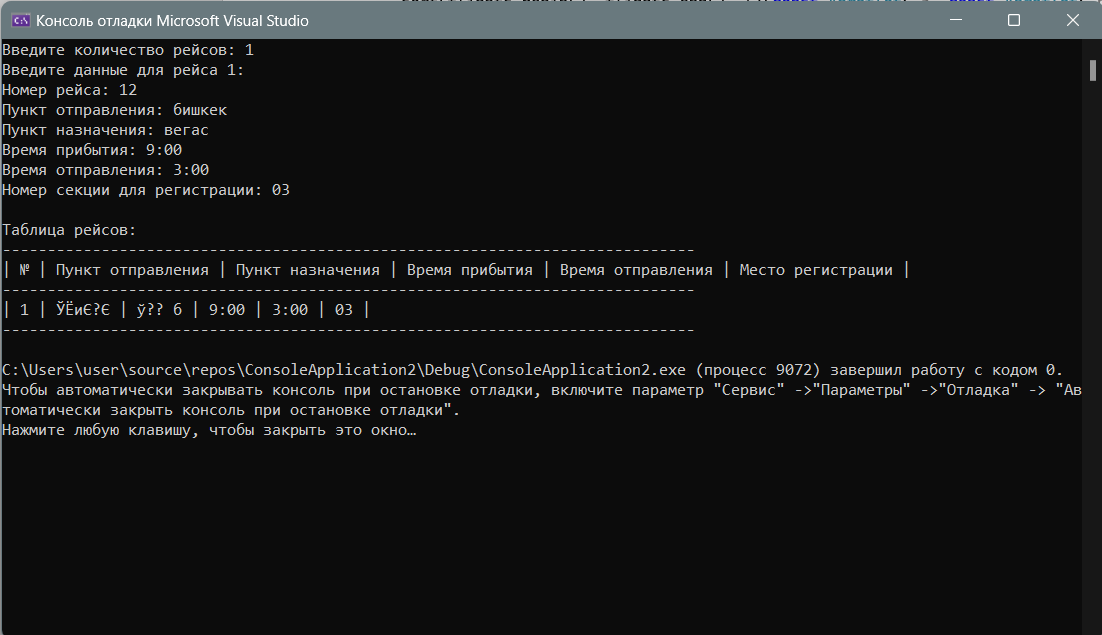
}

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

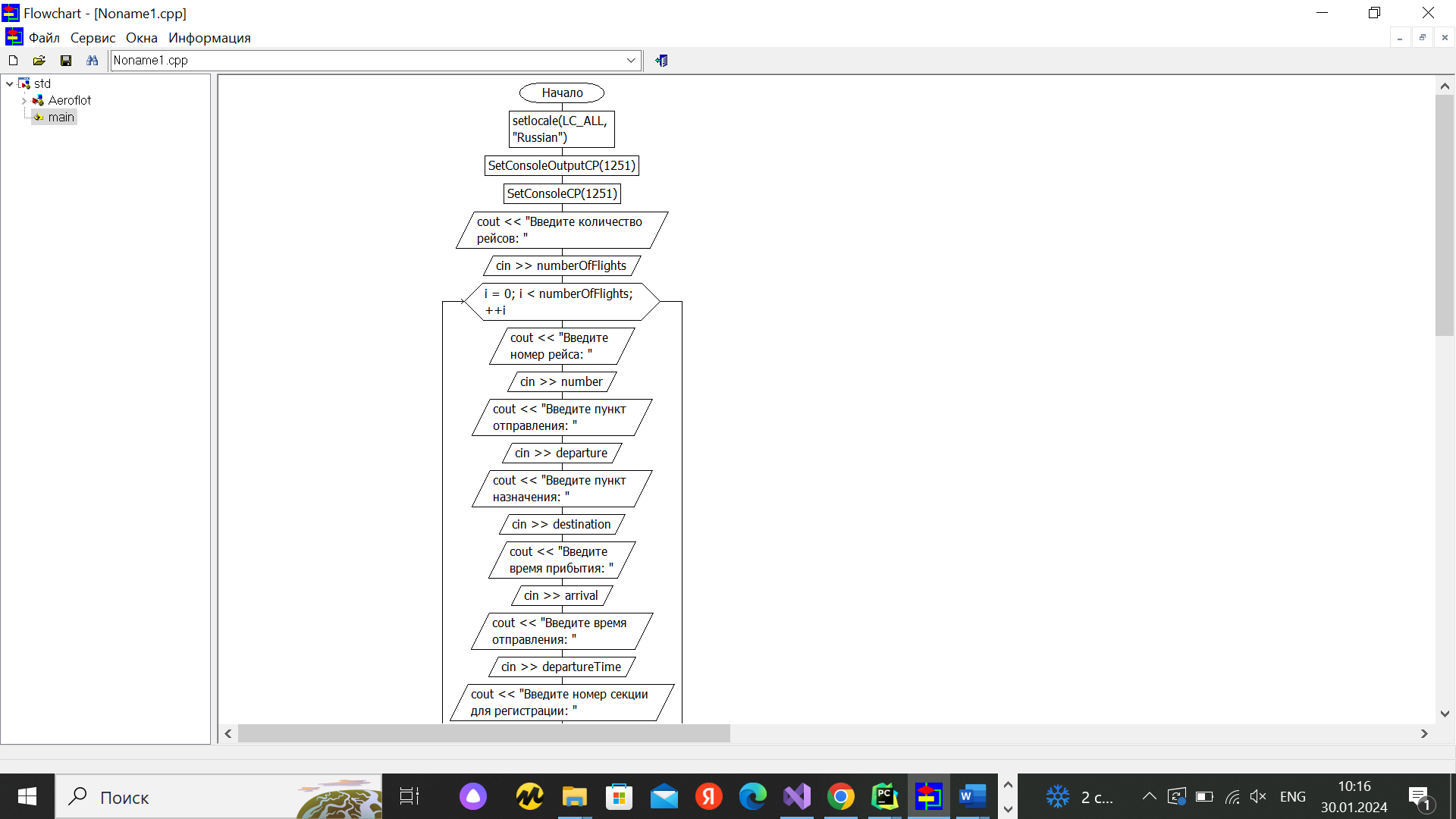
return 0;

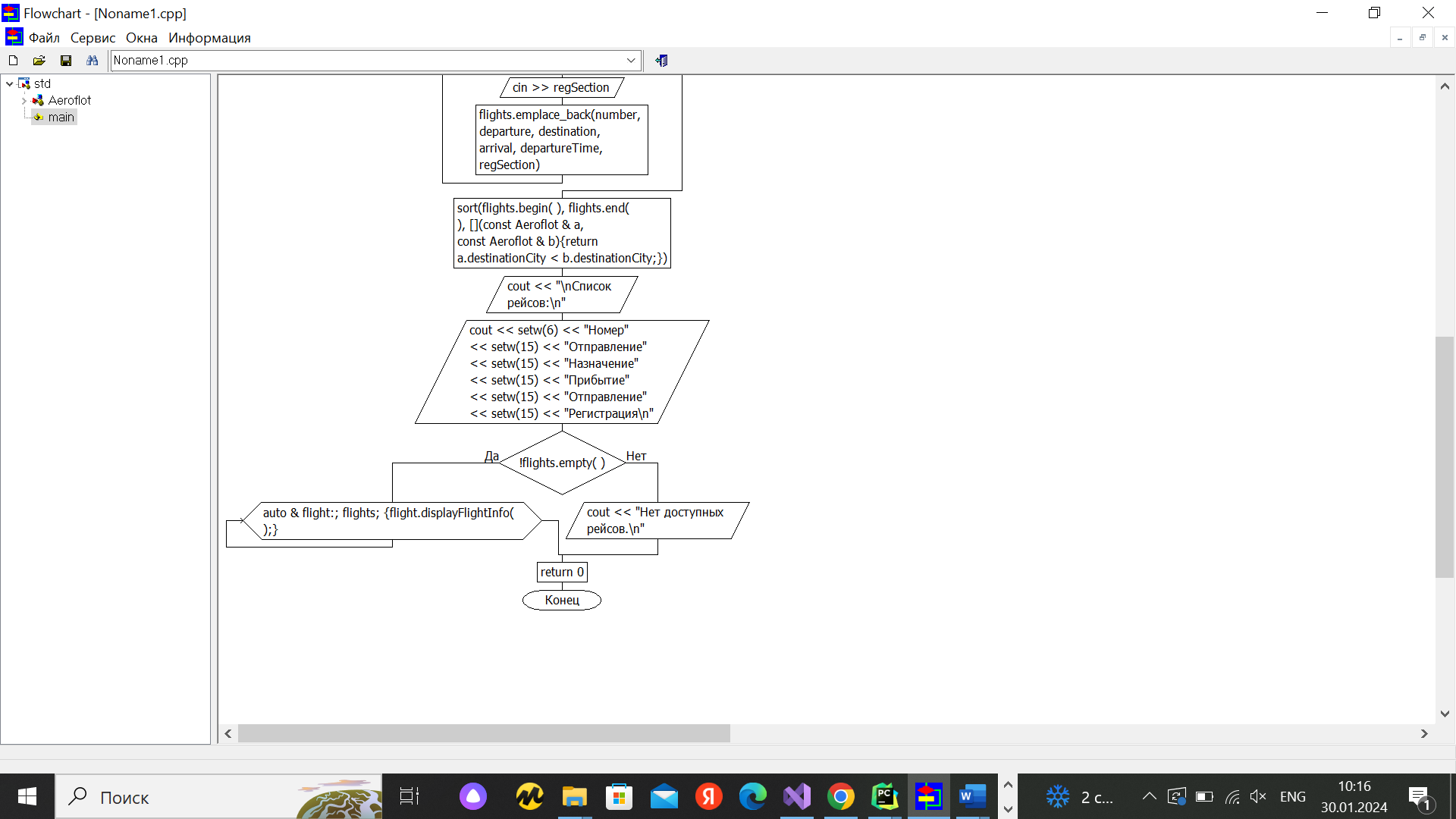
}

**Результат:**

****

**Блок-схема:**

****

****

**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:**

2 задание

#include <iostream>

#include <vector>

#include <algorithm>

class Student {

public:

std::string lastName;

std::string firstName;

std::string dateOfBirth;

std::string phoneNumber;

Student(const std::string& last, const std::string& first, const std::string& dob, const std::string& phone)

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

// Перегрузка оператора < для сортировки по фамилии

bool operator<(const Student& other) const {

return lastName < other.lastName;

}

// Перегрузка оператора == для поиска по фамилии

bool operator==(const std::string& targetLastName) const {

return lastName == targetLastName;

}

};

class StudentDatabase {

private:

std::vector<Student> students;

public:

void addStudent(const Student& student) {

students.push\_back(student);

}

void removeStudent(const std::string& targetLastName) {

auto it = std::remove\_if(students.begin(), students.end(),

[targetLastName](const Student& student) {

return student == targetLastName;

});

students.erase(it, students.end());

}

Student\* findStudent(const std::string& targetLastName) {

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

[targetLastName](const Student& student) {

return student == targetLastName;

});

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

return &(\*it);

}

else {

return nullptr;

}

}

void sortByLastName() {

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

}

void displayStudents() const {

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

std::cout << "| Фамилия | Имя | Дата рождения | Телефон |\n";

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

for (const auto& student : students) {

std::cout << "| " << student.lastName << " | " << student.firstName

<< " | " << student.dateOfBirth << " | " << student.phoneNumber << " |\n";

}

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

}

};

int main() {

setlocale(LC\_ALL, "Russian");

StudentDatabase database;

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 фамилии\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 last, first, dob, phone;

std::cout << "Введите фамилию: ";

std::cin >> last;

std::cout << "Введите имя: ";

std::cin >> first;

std::cout << "Введите дату рождения: ";

std::cin >> dob;

std::cout << "Введите номер телефона: ";

std::cin >> phone;

database.addStudent(Student(last, first, dob, phone));

break;

}

case 2: {

std::string targetLastName;

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

std::cin >> targetLastName;

database.removeStudent(targetLastName);

break;

}

case 3: {

std::string targetLastName;

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

std::cin >> targetLastName;

Student\* foundStudent = database.findStudent(targetLastName);

if (foundStudent) {

std::cout << "Студент найден:\n";

std::cout << "Фамилия: " << foundStudent->lastName << "\n";

std::cout << "Имя: " << foundStudent->firstName << "\n";

std::cout << "Дата рождения: " << foundStudent->dateOfBirth << "\n";

std::cout << "Телефон: " << foundStudent->phoneNumber << "\n";

}

else {

std::cout << "Студент не найден.\n";

}

break;

}

case 4: {

database.sortByLastName();

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

break;

}

case 5: {

database.displayStudents();

break;

}

case 0:

std::cout << "Программа завершена.\n";

break;

default:

std::cout << "Некорректный выбор. Пожалуйста, выберите снова.\n";

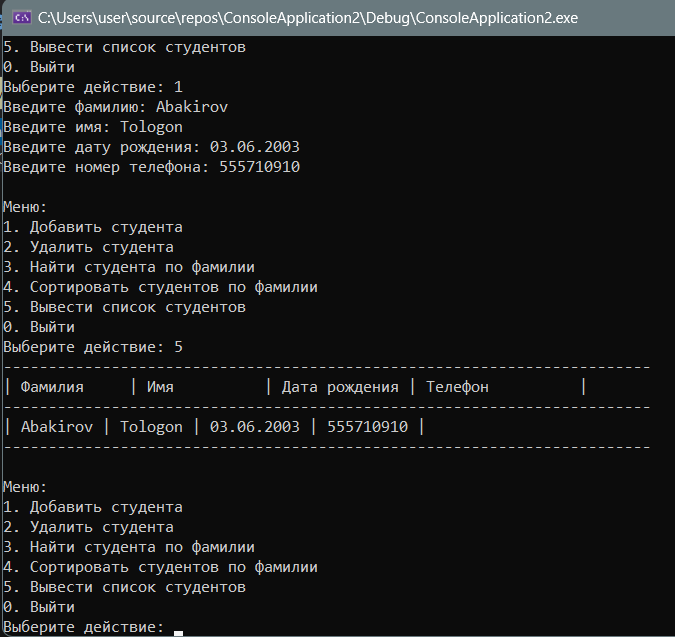
}

} while (choice != 0);

return 0;

}

**Результат:**



**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;

**Код:**

#include <iostream>

#include <fstream>

#include <stack>

#include <string>

class TrainSortingNode {

private:

std::stack<std::string> leftDirection; // Стек для левого направления

std::stack<std::string> rightDirection; // Стек для правого направления

public:

// Добавление вагона в левое направление

void addLeftWagon(const std::string& wagon) {

leftDirection.push(wagon);

}

// Добавление вагона в правое направление

void addRightWagon(const std::string& wagon) {

rightDirection.push(wagon);

}

// Удаление вагона из вершины левого направления

void removeLeftWagon() {

if (!leftDirection.empty()) {

leftDirection.pop();

}

else {

std::cout << "Левое направление пусто.\n";

}

}

// Удаление вагона из вершины правого направления

void removeRightWagon() {

if (!rightDirection.empty()) {

rightDirection.pop();

}

else {

std::cout << "Правое направление пусто.\n";

}

}

// Вывод информации о состоянии узла

void displayNodeState() const {

std::cout << "Состояние узла:\n";

std::cout << "Левое направление: ";

displayStackContents(leftDirection);

std::cout << "Правое направление: ";

displayStackContents(rightDirection);

}

private:

// Вспомогательная функция для вывода содержимого стека

void displayStackContents(const std::stack<std::string>& stack) const {

if (stack.empty()) {

std::cout << "Пусто\n";

}

else {

std::stack<std::string> temp = stack;

while (!temp.empty()) {

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

temp.pop();

}

std::cout << "\n";

}

}

};

int main() {

setlocale(LC\_ALL, "Russian");

TrainSortingNode sortingNode;

int choice;

do {

std::cout << "\nМеню:\n";

std::cout << "1. Добавить вагон в левое направление\n";

std::cout << "2. Удалить вагон из левого направления\n";

std::cout << "3. Добавить вагон в правое направление\n";

std::cout << "4. Удалить вагон из правого направления\n";

std::cout << "5. Вывести состояние узла\n";

std::cout << "0. Выйти\n";

std::cout << "Выберите действие: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string wagon;

std::cout << "Введите тип вагона для левого направления: ";

std::cin >> wagon;

sortingNode.addLeftWagon(wagon);

break;

}

case 2: {

sortingNode.removeLeftWagon();

break;

}

case 3: {

std::string wagon;

std::cout << "Введите тип вагона для правого направления: ";

std::cin >> wagon;

sortingNode.addRightWagon(wagon);

break;

}

case 4: {

sortingNode.removeRightWagon();

break;

}

case 5: {

sortingNode.displayNodeState();

break;

}

case 0:

std::cout << "Программа завершена.\n";

break;

default:

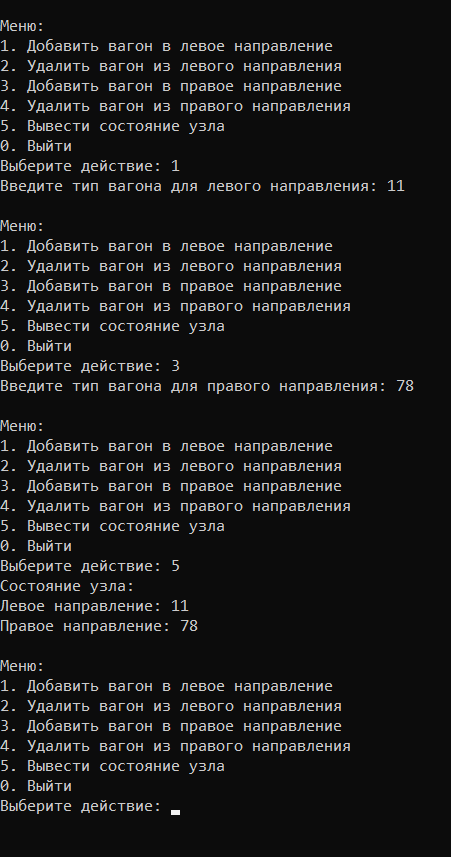
std::cout << "Некорректный выбор. Пожалуйста, выберите снова.\n";

}

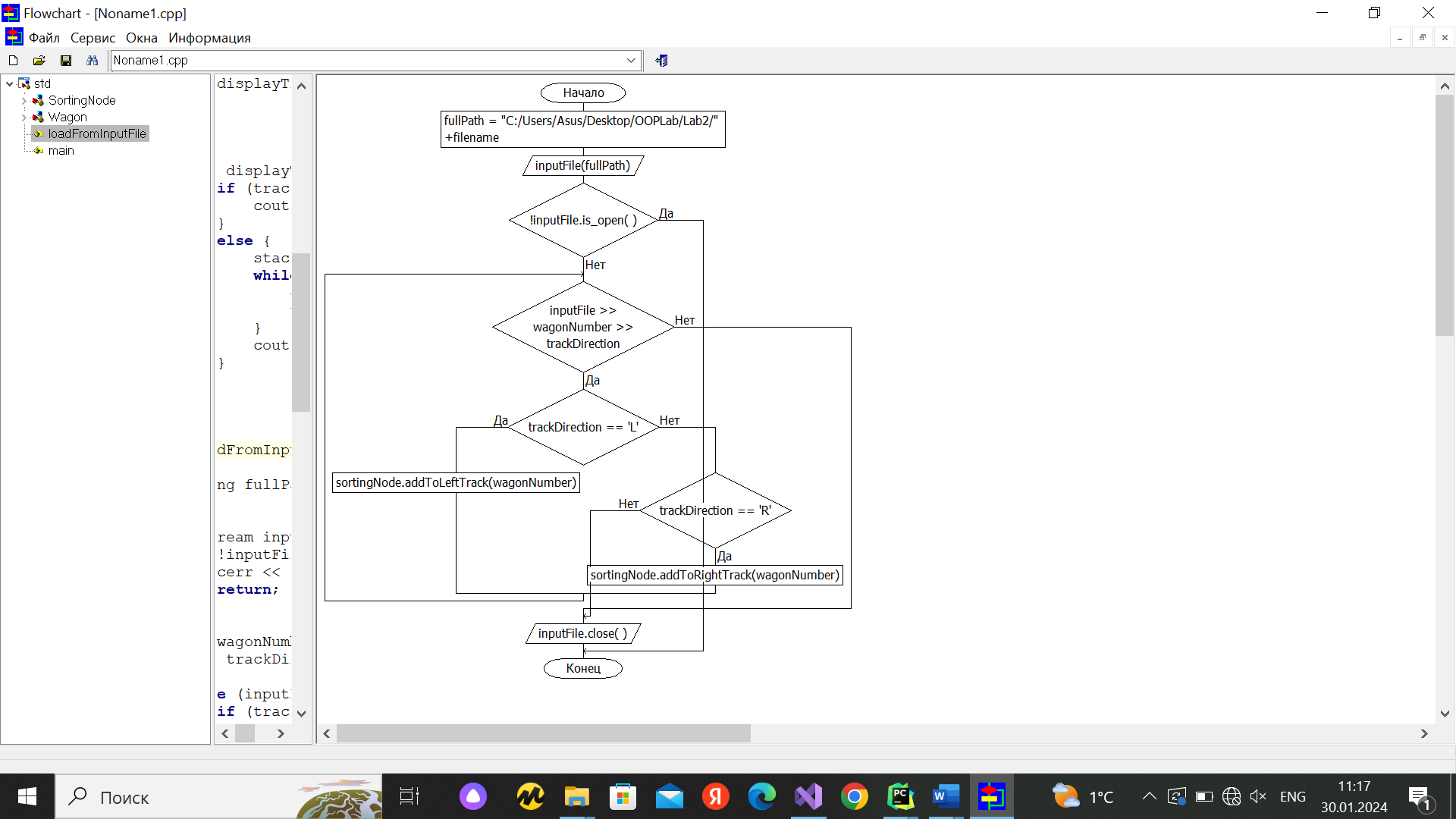
} while (choice != 0);

return 0;

}Результат:



**Блок – схема:**



**ВОПРОСЫ К ЗАЩИТЕ ЛАБОРАТОРНОЙ РАБОТЫ №2**

1. *Как вы понимаете, что такое класс и объект? Приведите примеры.*

Класс - это шаблон или описание для создания объектов. Он определяет, какие данные (переменные) и функции (методы) могут быть связаны вместе в одном объекте. Класс описывает структуру и поведение объектов.

Объект - это экземпляр класса, созданный на основе его описания (класса). Он представляет собой конкретный элемент, который содержит данные и может выполнять функции, определенные в классе.

1. *Каков синтаксис и смысл определения класса? Как называются составные части класса?*

**Синтаксис определения класса:** Класс определяется с использованием ключевого слова **class**, за которым следует имя класса. Тело класса заключается в фигурные скобки **{}** и содержит переменные и функции-члены класса.

**Смысл определения класса** - это как план для создания объектов. Класс говорит, какие данные (переменные) и действия (функции) могут быть у объектов этого класса, но сам по себе класс не занимает память. Память выделяется только когда мы создаем объект на основе этого класса. Класс - это как чертеж, а объект - это уже дом, построенный по этому чертежу.

**Составные части класса называются полями и методами.**

Поля - это переменные, которые хранят данные объекта класса. Например, в классе "Автомобиль" поля могут быть "марка", "модель", "цвет".

Методы - это функции, которые определяют действия, которые объект класса может выполнять. Например, в классе "Автомобиль" методом может быть "завести двигатель".

1. *Что такое идентификаторы доступа к элементам класса? Дайте определения.*

Существуют **три** основных идентификатора доступа:

* **public:** Элементы класса, объявленные с public, являются общедоступными и могут быть доступны извне класса. Это означает, что данные элементы (поля и методы) могут быть использованы в других частях программы.
* **private:** Элементы класса, объявленные с private, являются закрытыми и недоступными извне класса. Они могут быть использованы только внутри самого класса.
* **protected:** Элементы класса, объявленные с protected, имеют ограниченный доступ и могут быть использованы как внутри класса, так и в производных классах (наследниках). Они не доступны извне класса и его производных классов.

Идентификаторы доступа позволяют регулировать уровень инкапсуляции и защиты данных и методов в классе, обеспечивая контролируемый доступ к ним из других частей программы.

1. *Что такое инкапсуляция?*

**Инкапсуляция** – это механизм, который объединяет данные и код, манипулирующий с этими данными, а также защищает и то и другое от внешнего вмешательства или неправильного использования. Позволяет скрыть конкретную реализацию класса, облегчая отладку и модификацию программ.

1. *Где в программе с классами можно создавать объекты?*

Список объектов можно определить позднее в функции **main().**

1. *Каков смысл объявления объектов?*

Смысл объявления объектов заключается в создании конкретных экземпляров классов, которые могут хранить данные и использовать методы, определенные в классе, для выполнения конкретных задач в программе.

1. *В какой момент метод готов для вызова объектом?*

Метод готов для вызова объектом в момент, когда объект был создан и инициализирован, и метод был объявлен и определен в классе.

1. *Как называются методы, встроенные в структуру класса?*

Методы, встроенные в структуру класса, называются методами-членами класса или методами объекта класса.

1. *Как определяется метод, если внутри класса записан только его заголовок, сам метод определен вне класса?*

Если внутри класса записан только заголовок метода, а его определение находится вне класса, то это называется "разделение метода" (separate method definition). Это позволяет вынести реализацию метода за пределы класса, что может быть полезно для улучшения читаемости кода и разделения интерфейса класса от его реализации.

1. *Если в классе два поля данных и два объекта, сколько полей принадлежит каждому объекту? Совпадет ли имена и значения этих полей для объектов?*

Если в классе есть два поля данных и создано два объекта этого класса, то каждому объекту принадлежит полный набор этих двух полей данных. Имена полей для обоих объектов будут одинаковыми, но значения полей могут различаться, так как каждый объект имеет свои собственные значения для своих полей данных.

Другими словами, имена полей будут общими для всех объектов данного класса, но значения полей будут индивидуальными для каждого объекта.

1. *Тиражируются ли методы класса?*

Да, методы класса тиражируются, что означает, что они существуют только в одном экземпляре в памяти, независимо от количества объектов этого класса, и могут вызываться для каждого объекта этого класса.

1. *Как в программе написать доступ к открытым и закрытым полям класса?*

Для доступа к открытым полям класса (публичным членам) можно использовать их имена напрямую из объекта класса. Для доступа к закрытым полям (приватным членам) класса, обычно используют публичные методы (геттеры и сеттеры) класса.

Пример доступа к открытому полю и закрытому полю класса в C++:

#include <iostream>

using namespace std;

class MyClass {

public:

int publicField; // Открытое поле

private:

int privateField; // Закрытое поле

public:

// Геттер для закрытого поля

int getPrivateField() {

return privateField;

}

// Сеттер для закрытого поля

void setPrivateField(int value) {

privateField = value;

}

};

int main() {

MyClass obj;

obj.publicField = 42; // Доступ к открытому полю напрямую

obj.setPrivateField(10); // Доступ к закрытому полю через сеттер

cout << "Открытое поле: " << obj.publicField << endl;

cout << "Закрытое поле (через геттер): " << obj.getPrivateField() << endl;

return 0;

}

В этом примере, publicField является открытым полем и к нему можно обращаться напрямую, в то время как privateField - закрытым полем, и для доступа к нему используются геттер и сеттер методы.