**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

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

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Дисциплина «**Объектно-Ориентированное Программирование**»

**Отчет**

**Лабораторная работа №2**

Выполнил: студент группы ПИ-2-21

Газиев Давид

Проверил: Мусабаев Э. Б.

**Бишкек 2024**

**Задание 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 программы**

#include <iostream>

#include <string>

#include <vector>

#include <algorithm>

#include <windows.h>

using namespace std;

class Aeroflot {

private:

int flightNumber;

string departurePoint;

string destinationPoint;

string arrivalTime;

string departureTime;

int registrationSection;

public:

Aeroflot(int flightNum, const string& departurePt, const string& destinationPt, const string& arrivalT, const string& departureT, int registrationSec)

: flightNumber(flightNum), departurePoint(departurePt), destinationPoint(destinationPt),

arrivalTime(arrivalT), departureTime(departureT), registrationSection(registrationSec) {}

int getFlightNumber() const { return flightNumber; }

string getDeparturePoint() const { return departurePoint; }

string getDestinationPoint() const { return destinationPoint; }

string getArrivalTime() const { return arrivalTime; }

string getDepartureTime() const { return departureTime; }

int getRegistrationSection() const { return registrationSection; }

};

void bubbleSort(vector<Aeroflot>& flights) {

int n = flights.size();

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

for (int j = 0; j < n - i - 1; ++j) {

if (flights[j].getDestinationPoint() > flights[j + 1].getDestinationPoint()) {

swap(flights[j], flights[j + 1]);

}

}

}

}

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

setlocale(LC\_ALL, "ru");

vector<Aeroflot> flights;

int numFlights;

cout << "Введите количество рейсов: ";

cin >> numFlights;

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

int flightNumber;

string departurePoint, destinationPoint, arrivalTime, departureTime;

int registrationSection;

cout << "Введите информацию про " << i + 1 << "-й рейс\n";

cout << "Введите номер рейса: ";

cin >> flightNumber;

cout << "Введите пункт отправления: ";

cin >> departurePoint;

cout << "Введите пункт назначения: ";

cin >> destinationPoint;

cout << "Введите время прибытия: ";

cin >> arrivalTime;

cout << "Введите время отправления: ";

cin >> departureTime;

cout << "Введите номер секции для регистрации: ";

cin >> registrationSection;

flights.emplace\_back(flightNumber, departurePoint, destinationPoint, arrivalTime, departureTime, registrationSection);

}

bubbleSort(flights);

cout << "------------------------------------------------------------------------------------------------------------" << endl;

cout << "| # Рейса | Пункт отправления | Пункт назначения | Время прибытия | Время отправления | Секция регистрации |" << endl;

cout << "------------------------------------------------------------------------------------------------------------" << endl;

bool flightsFound = false;

for (const auto& flight : flights) {

cout << "| " << flight.getFlightNumber() << " | " << flight.getDeparturePoint() << " | " << flight.getDestinationPoint() << " | "

<< flight.getArrivalTime() << " | " << flight.getDepartureTime() << " | " << flight.getRegistrationSection() << " |" << endl;

flightsFound = true;

}

if (!flightsFound) {

cout << "Нет рейсов для отображения." << endl;

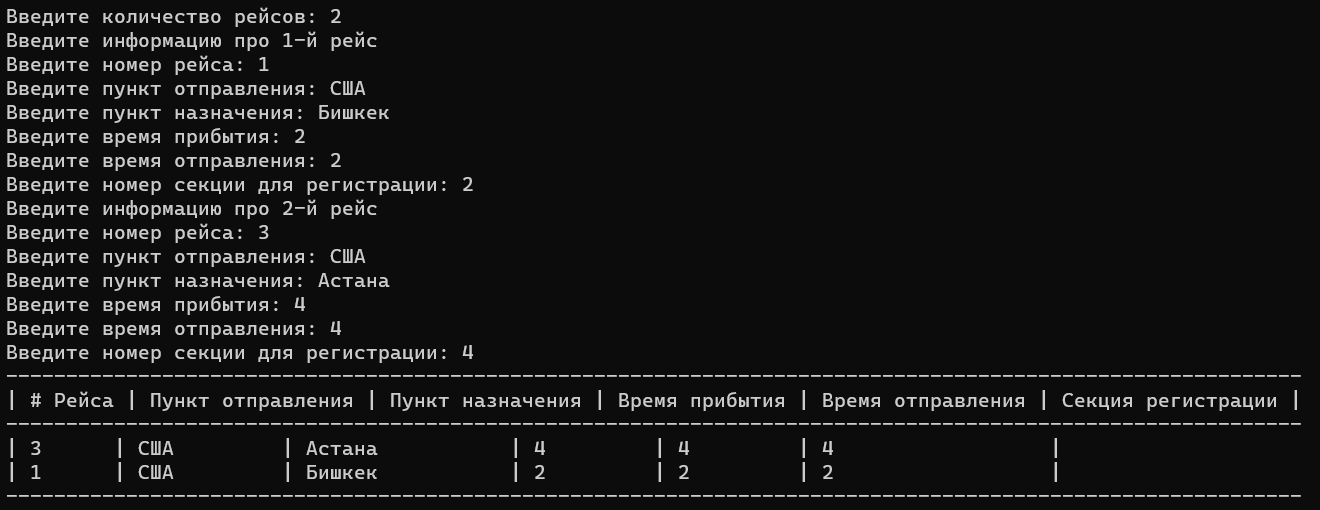
}

cout << "------------------------------------------------------------------------------------------------------------" << endl;

return 0;

}

**Проверка выполнения**

****

**Блок схема**



**Задание 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 программы**

#include <iostream>

#include <Windows.h>

#include <string>

#include <vector>

#include <algorithm>

using namespace std;

class Student {

private:

string surname;

string birthDate;

string phoneNumber;

public:

Student(const string& sur, const string& bDate, const string& phone) : surname(sur), birthDate(bDate), phoneNumber(phone) {}

string getSurname() const { return surname; }

string getBirthDate() const { return birthDate; }

string getPhoneNumber() const { return phoneNumber; }

};

class StudentDatabase {

private:

vector<Student> students;

public:

void addStudent(const string& surname, const string& birthDate, const string& phoneNumber) {

students.emplace\_back(surname, birthDate, phoneNumber);

}

bool removeStudent(const string& surname) {

auto it = remove\_if(students.begin(), students.end(), [&](const Student& student) {

return student.getSurname() == surname;

});

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

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

return true;

}

return false;

}

Student\* findStudentBySurname(const string& surname) {

auto it = find\_if(students.begin(), students.end(), [&](const Student& student) {

return student.getSurname() == surname;

});

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

return &(\*it);

}

return nullptr;

}

void sortStudentsBySurname() {

sort(students.begin(), students.end(), [](const Student& a, const Student& b) {

return a.getSurname() < b.getSurname();

});

}

void displayAllStudents() const {

if (students.empty()) {

cout << "База данных студентов пуста." << endl;

return;

}

cout << "-----------------------------------------" << endl;

cout << "| Фамилия | Дата рождения | Телефон |" << endl;

cout << "-----------------------------------------" << endl;

for (const auto& student : students) {

cout << "| " << student.getSurname() << " | " << student.getBirthDate() << " | " << student.getPhoneNumber() << " |" << endl;

}

cout << "-----------------------------------------" << endl;

}

bool hasStudents() const {

return !students.empty();

}

};

int main()

{

setlocale(LC\_ALL, "ru");

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

StudentDatabase database;

while (true) {

cout << "Меню:" << endl;

cout << "1. Добавить студента" << endl;

cout << "2. Удалить студента" << endl;

cout << "3. Найти студента по фамилии" << endl;

cout << "4. Отсортировать студентов по фамилии" << endl;

cout << "5. Вывести всех студентов" << endl;

cout << "0. Выйти" << endl;

int choice;

cout << "Выберите действие: ";

cin >> choice;

if (!database.hasStudents() && choice != 1 && choice != 6) {

cout << "База данных студентов пуста. Добавьте студента или выберите пункт 'Выйти'." << endl;

continue;

}

switch (choice) {

case 1: {

string surname, birthDate, phoneNumber;

cout << "Введите фамилию: ";

cin >> surname;

cout << "Введите дату рождения (дд.мм.гггг): ";

cin >> birthDate;

cout << "Введите номер телефона: ";

cin >> phoneNumber;

database.addStudent(surname, birthDate, phoneNumber);

break;

}

case 2: {

string surname;

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

cin >> surname;

if (database.removeStudent(surname)) {

cout << "Студент " << surname << " успешно удален\n";

database.displayAllStudents();

}

else {

cout << "Студента " << surname << " нет в БД\n";

}

break;

}

case 3: {

string surname;

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

cin >> surname;

Student\* foundStudent = database.findStudentBySurname(surname);

if (foundStudent != nullptr) {

cout << "Студент найден: " << foundStudent->getSurname() << " " << foundStudent->getBirthDate() << " " << foundStudent->getPhoneNumber() << endl;

}

else {

cout << "Студент не найден." << endl;

}

break;

}

case 4: {

database.sortStudentsBySurname();

cout << "Студенты отсортированы по фамилии." << endl;

database.displayAllStudents();

break;

}

case 5: {

database.displayAllStudents();

break;

}

case 0: {

return 0;

}

default: {

cout << "Некорректный выбор. Попробуйте снова." << endl;

break;

}

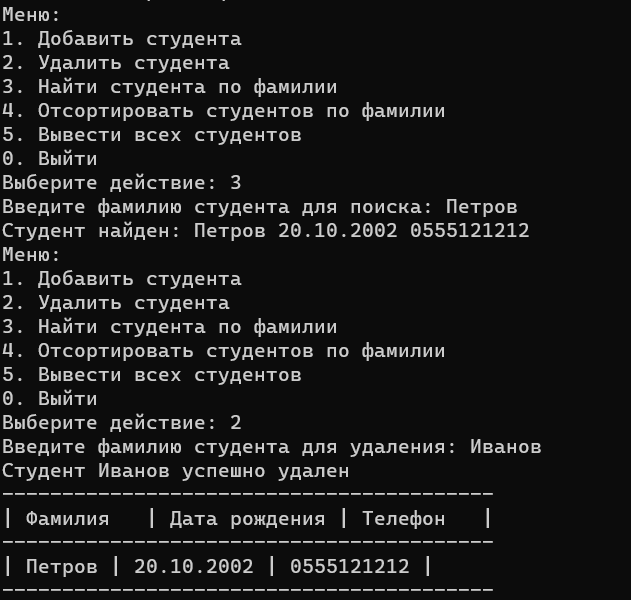
}

}

return 0;

}

**Проверка выполнения**

****

**Блок схема**



**Задание 3**

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 программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Node {

char data;

Node\* next;

Node(char d) : data(d), next(nullptr) {}

};

class Stack {

private:

Node\* topNode;

public:

Stack() : topNode(nullptr) {}

void push(char item) {

Node\* newNode = new Node(item);

newNode->next = topNode;

topNode = newNode;

}

void pop() {

if (!isEmpty()) {

Node\* temp = topNode;

topNode = topNode->next;

delete temp;

}

}

char top() const {

if (!isEmpty()) {

return topNode->data;

}

return '\0';

}

bool isEmpty() const {

return topNode == nullptr;

}

};

void separateTrain(const Stack& train, Stack& directionA, Stack& directionB) {

Stack tempTrain = train;

while (!tempTrain.isEmpty()) {

char wagon = tempTrain.top();

tempTrain.pop();

if (wagon == 'A') {

directionA.push(wagon);

}

else if (wagon == 'B') {

directionB.push(wagon);

}

}

}

void printStack(const Stack& stack) {

Stack tempStack = stack;

while (!tempStack.isEmpty()) {

cout << tempStack.top() << " ";

tempStack.pop();

}

cout << endl;

}

int main() {

setlocale(LC\_ALL, "ru");

Stack train;

Stack directionA;

Stack directionB;

char choice;

cout << "Хотите ввести состав с клавиатуры? (y/n): ";

cin >> choice;

if (choice == 'y' || choice == 'Y') {

cout << "Введите состав: (A - вагон типа A, B - вагон типа B, X - конец ввода)" << endl;

char wagon;

while (true) {

cin >> wagon;

if (wagon == 'X' || wagon == 'x') {

break;

}

train.push(wagon);

}

}

else {

ifstream file("train.txt");

if (!file) {

cerr << "Ошибка открытия файла." << endl;

return 1;

}

char wagon;

while (file >> wagon) {

train.push(wagon);

}

file.close();

}

separateTrain(train, directionA, directionB);

cout << "Направление A:" << endl;

printStack(directionA);

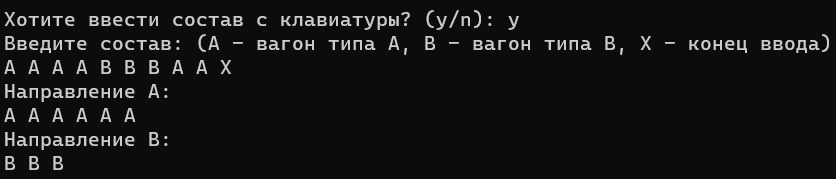
cout << "Направление B:" << endl;

printStack(directionB);

return 0;

}

**Проверка выполнения**

****

**Блок схема**

