**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**КЫРГЫЗСКОЙ РЕСПУБЛИКИ**

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

**ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

Кафедра: **Программное обеспечение компьютерных систем**

Курс «Объектно-ориентированное программирование»

**ОТЧЕТ**

**Лабораторная работа №2**

Выполнил: студент группы ПИ-3-21

Алтынбеков Даурен

Проверил: Мусабаев Э.Б.

**Бишкек 2024**

# **Задание №1**

**Постановка задачи:**

Вам предоставляется код, реализующий работу с информацией о рейсах авиакомпании с использованием класса **Aeroflot**. Задача состоит в разработке программы для управления данными о рейсах, их вводе, сортировке и выводе.

**Функциональность программы:**

1. **Класс Aeroflot:**
   * Класс предназначен для хранения информации о рейсе.
   * В классе присутствуют приватные свойства, такие как номер рейса, пункт отправления, пункт назначения, время прибытия, время отправления и номер секции для регистрации.
   * Для каждого свойства предусмотрены геттеры для доступа к данным.
2. **Функция сравнения compareFlights:**
   * Реализована функция сравнения двух рейсов по пункту назначения.
3. **Основная программа:**
   * Программа запрашивает количество рейсов.
   * Для каждого рейса пользователю предлагается ввести информацию, такую как номер рейса, пункт отправления, пункт назначения, время прибытия, время отправления и номер секции для регистрации.
   * Введенные данные сохраняются в объекты класса **Aeroflot** и помещаются в вектор **flights**.
   * Рейсы в векторе сортируются по пункту назначения с использованием функции **compareFlights**.
   * Отсортированные рейсы выводятся в табличной форме с использованием форматирования.

**Исходный код программы на C++:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <iomanip>

using namespace std;

class Aeroflot {

private:

int flightNumber;

string departureCity;

string destinationCity;

string arrivalTime;

string departureTime;

string registrationSection;

public:

Aeroflot(int num, const string& depCity, const string& destCity, const string& arrTime, const string& depTime, const string& regSection)

: flightNumber(num), departureCity(depCity), destinationCity(destCity), arrivalTime(arrTime), departureTime(depTime), registrationSection(regSection) {}

// Геттеры для доступа к приватным свойствам

int getFlightNumber() const { return flightNumber; }

const string& getDepartureCity() const { return departureCity; }

const string& getDestinationCity() const { return destinationCity; }

const string& getArrivalTime() const { return arrivalTime; }

const string& getDepartureTime() const { return departureTime; }

const string& getRegistrationSection() const { return registrationSection; }

// Функция вывода информации о рейсе

void displayFlightInfo() const {

cout << setw(6) << getFlightNumber() << setw(15) << getDepartureCity() << setw(15) << getDestinationCity()

<< setw(15) << getArrivalTime() << setw(15) << getDepartureTime() << setw(15) << getRegistrationSection() << endl;

}

};

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

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

}

int main() {

setlocale(LC\_ALL, "Russian");

vector<Aeroflot> flights;

int numFlights;

cout << "Введите количество рейсов: ";

cin >> numFlights;

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

int num;

string depCity, destCity, arrTime, depTime, regSection;

cout << "Введите информацию о рейсе " << i + 1 << ":" << endl;

cout << "Номер рейса: ";

cin >> num;

cout << "Пункт отправления: ";

cin >> depCity;

cout << "Пункт назначения: ";

cin >> destCity;

cout << "Время прибытия: ";

cin >> arrTime;

cout << "Время отправления: ";

cin >> depTime;

cout << "Место регистрации: ";

cin >> regSection;

flights.emplace\_back(num, depCity, destCity, arrTime, depTime, regSection);

}

sort(flights.begin(), flights.end(), compareFlights);

cout << setw(6) << "Номер" << setw(15) << "Отправление" << setw(15) << "Назначение"

<< setw(15) << "Прибытие" << setw(15) << "Отправление" << setw(15) << "Регистрация" << endl;

for (const auto& flight : flights) {

flight.displayFlightInfo();

}

if (flights.empty()) {

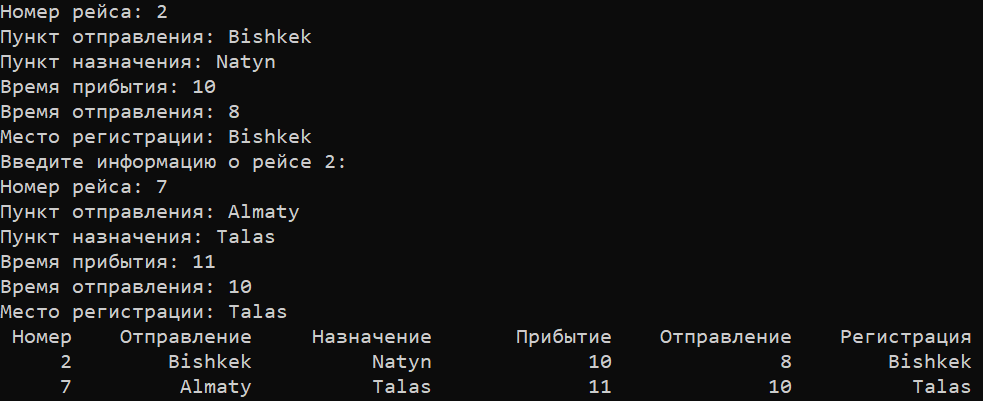
cout << "Нет информации о рейсах." << endl;

}

return 0;

}

**Тесты:**



# **Задание №2**

**Постановка задачи:**

Вам предоставлен код, реализующий работу с информацией о студентах и базе данных студентов с использованием классов Student и StudentDatabase. Задача состоит в разработке программы для управления данными о студентах, их добавлении, удалении, поиске и отображении.

**Функциональность программы:**

1. **Класс Student:**
   * Класс предназначен для хранения информации о студенте.
   * В классе присутствуют приватные свойства, такие как имя, фамилия, дата рождения и номер телефона.
   * Для каждого свойства предусмотрены геттеры для доступа к данным.
2. **Класс StudentDatabase:**
   * Класс предназначен для управления базой данных студентов.
   * В базе данных реализованы функции добавления студента, удаления студента по фамилии, поиска студента по фамилии, дате рождения или номеру телефона, сортировки студентов по фамилии и отображения списка всех студентов.
3. **Основная программа:**
   * Программа запрашивает пользователя вводить данные о студентах или выполнять различные операции с базой данных студентов.
   * Предоставлены следующие опции:
     + Добавление студента.
     + Удаление студента по фамилии.
     + Поиск студента по фамилии, дате рождения или номеру телефона.
     + Сортировка студентов по фамилии.
     + Отображение списка всех студентов.
     + Выход из программы.

**Исходный код программы на C++:**

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

class Student {

private:

string firstName;

string lastName;

string birthDate;

string phoneNumber;

public:

Student(const string& fName, const string& lName, const string& bDate, const string& phone)

: firstName(fName), lastName(lName), birthDate(bDate), phoneNumber(phone) {}

const string& getFirstName() const { return firstName; }

const string& getLastName() const { return lastName; }

const string& getBirthDate() const { return birthDate; }

const string& getPhoneNumber() const { return phoneNumber; }

void displayStudentInfo() const {

cout << "Фамилия: " << getLastName() << "\nИмя: " << getFirstName() << "\nДата рождения: " << getBirthDate() << "\nТелефон: " << getPhoneNumber() << endl;

}

};

class StudentDatabase {

private:

vector<Student> students;

public:

void addStudent(const Student& student) {

students.push\_back(student);

}

void removeStudent(const string& lastName) {

auto it = find\_if(students.begin(), students.end(), [&](const Student& s) { return s.getLastName() == lastName; });

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

students.erase(it);

cout << "Студент с фамилией " << lastName << " удален." << endl;

}

else {

cout << "Студент с фамилией " << lastName << " не найден." << endl;

}

}

void searchStudent(const string& searchKey) const {

for (const auto& student : students) {

if (student.getLastName() == searchKey || student.getBirthDate() == searchKey || student.getPhoneNumber() == searchKey) {

student.displayStudentInfo();

return;

}

}

cout << "Студент с указанным признаком не найден." << endl;

}

void sortStudentsByName() {

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

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

});

}

void displayAllStudents() const {

if (students.empty()) {

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

return;

}

cout << "Список студентов:" << endl;

for (const auto& student : students) {

student.displayStudentInfo();

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

}

}

};

int main() {

setlocale(LC\_ALL, "Russian");

StudentDatabase database;

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Добавить студента\n";

cout << "2. Удалить студента\n";

cout << "3. Поиск студента\n";

cout << "4. Сортировка по фамилии\n";

cout << "5. Показать всех студентов\n";

cout << "0. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

string fName, lName, bDate, phone;

cout << "Введите данные студента:\n";

cout << "Имя: ";

cin >> fName;

cout << "Фамилия: ";

cin >> lName;

cout << "Дата рождения: ";

cin >> bDate;

cout << "Номер телефона: ";

cin >> phone;

Student newStudent(fName, lName, bDate, phone);

database.addStudent(newStudent);

break;

}

case 2: {

string lastName;

cout << "Введите фамилию студента, которого нужно удалить: ";

cin >> lastName;

database.removeStudent(lastName);

break;

}

case 3: {

string searchKey;

cout << "Введите фамилию, дату рождения или номер телефона для поиска студента: ";

cin >> searchKey;

database.searchStudent(searchKey);

break;

}

case 4:

database.sortStudentsByName();

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

break;

case 5:

database.displayAllStudents();

break;

case 0:

cout << "Программа завершена." << endl;

break;

default:

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

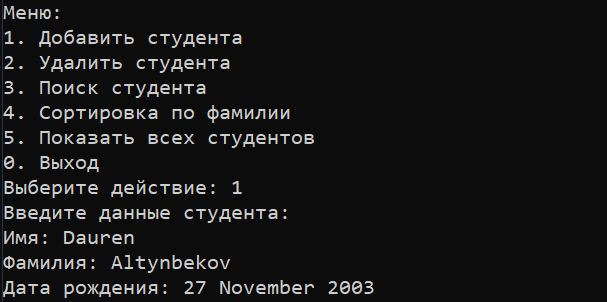
}

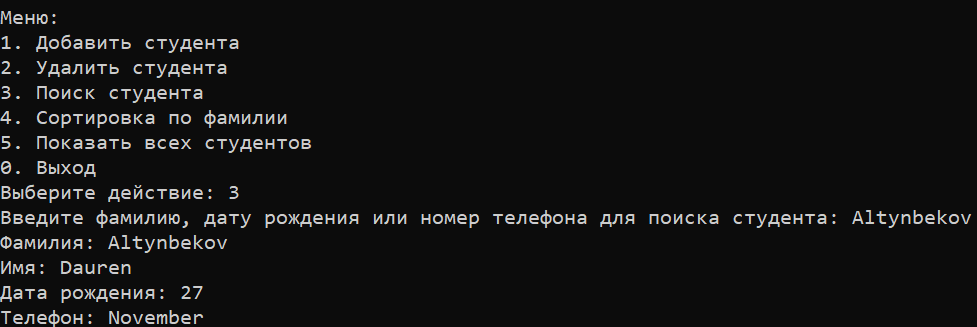
} while (choice != 0);

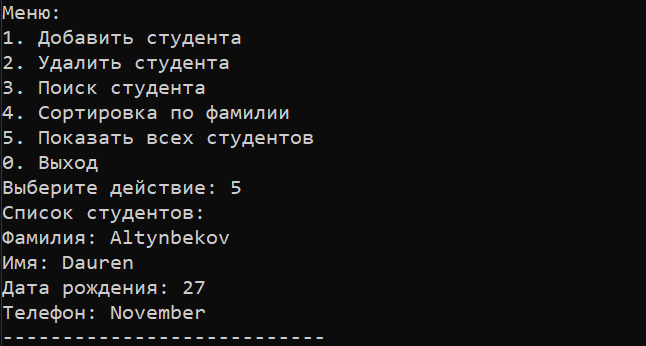
return 0;

}

**Тесты:**







# **Задание №3**

**Постановка задачи:**

Вам предоставлен код, реализующий работу с Т-образным сортировочным узлом на железной дороге с использованием класса **TrainSortingNode**. Задача состоит в разработке программы для управления данными о вагонах, их добавлении, сортировке и отображении в Т-образном сортировочном узле.

**Функциональность программы:**

1. **Класс TrainSortingNode:**
   * Класс предназначен для моделирования Т-образного сортировочного узла на железной дороге.
   * В классе реализованы функции добавления вагона в узел, сортировки вагонов в Т-образный узел и отображения содержимого узла.
   * Узел состоит из трех стеков: **type1Wagons**, **type2Wagons** и **sortingNode**.
2. **Основная программа:**
   * Программа предоставляет пользователю меню с несколькими опциями.
   * Пользователю доступны следующие действия:
     + Добавление вагона в Т-образный узел (вводится тип вагона: 1 или 2).
     + Сортировка вагонов в Т-образном узле.
     + Отображение содержимого Т-образного узла.
     + Загрузка вагонов из файла (вводится имя файла).
     + Выход из программы.

**Исходный код программы на C++:**

#include <iostream>

#include <fstream>

#include <stack>

#include <vector>

#include <algorithm>

using namespace std;

class TrainSortingNode {

private:

stack<int> type1Wagons;

stack<int> type2Wagons;

stack<int> sortingNode;

public:

void addWagon(int wagonType) {

if (wagonType == 1) {

type1Wagons.push(wagonType);

}

else if (wagonType == 2) {

type2Wagons.push(wagonType);

}

}

void sortWagons() {

while (!type1Wagons.empty() || !type2Wagons.empty()) {

if (!type1Wagons.empty()) {

sortingNode.push(type1Wagons.top());

type1Wagons.pop();

}

if (!type2Wagons.empty()) {

sortingNode.push(type2Wagons.top());

type2Wagons.pop();

}

}

}

void displaySortingNode() const {

cout << "Состав в Т-образном сортировочном узле:" << endl;

stack<int> tempStack = sortingNode;

while (!tempStack.empty()) {

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

tempStack.pop();

}

cout << endl;

}

void loadWagonsFromFile(const string& filename) {

ifstream inputFile(filename);

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

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

return;

}

int wagonType;

while (inputFile >> wagonType) {

addWagon(wagonType);

}

inputFile.close();

}

};

int main() {

setlocale(LC\_ALL, "Russian");

TrainSortingNode trainNode;

int choice;

do {

cout << "\nМеню:\n";

cout << "1. Добавить вагон\n";

cout << "2. Сформировать Т-образный сортировочный узел\n";

cout << "3. Отобразить состав в узле\n";

cout << "4. Загрузить вагоны из файла\n";

cout << "0. Выход\n";

cout << "Выберите действие: ";

cin >> choice;

switch (choice) {

case 1: {

int wagonType;

cout << "Введите тип вагона (1 или 2): ";

cin >> wagonType;

trainNode.addWagon(wagonType);

break;

}

case 2:

trainNode.sortWagons();

cout << "Т-образный сортировочный узел сформирован." << endl;

break;

case 3:

trainNode.displaySortingNode();

break;

case 4: {

string filename;

cout << "Введите имя файла: ";

cin >> filename;

trainNode.loadWagonsFromFile(filename);

break;

}

case 0:

cout << "Программа завершена." << endl;

break;

default:

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

}

} while (choice != 0);

return 0;

}

**Тесты:**

