Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Факультет информационных технологий

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно – ориентированное программирование»

**ОТЧЕТ**

По лабораторной работе №2.

Тема: «Классы и объекты»

                                                                       Выполнила: студент группы

ПИ(б)-2-21 Абдылдаева Аэлита

Проверил: Мусабаев Э. Б.

**Бишкек – 2024**

**Тема:** Классы и объекты.

**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:***

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Aeroflot {

public:

int flightNumber;

string departureCity;

string destinationCity;

string arrivalTime;

string departureTime;

int registrationSection;

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

const string& arrival, const string& departureTime, int section)

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

arrivalTime(arrival), departureTime(departureTime), registrationSection(section) {}

void displayFlightInfo() const {

cout << "| " << flightNumber << " | " << departureCity << " | " << destinationCity

<< " | " << arrivalTime << " | " << departureTime << " | " << registrationSection << " |"

<< endl;

}

};

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

return a.destinationCity < b.destinationCity;

}

void showAllFlights(const vector<Aeroflot>& flights) {

// Вывод информации о всех рейсах

cout << "| Номер | Отправление | Назначение | Прибытие | Отправление | Регистрация |" << endl;

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

for (const auto& flight : flights) {

flight.displayFlightInfo();

}

if (flights.empty()) {

cout << "Нет доступных рейсов." << endl;

}

}

Aeroflot chooseFlight(const vector<Aeroflot>& flights) {

int flightNumber;

cout << "Введите номер рейса: ";

cin >> flightNumber;

auto it = find\_if(flights.begin(), flights.end(), [flightNumber](const Aeroflot& flight) {

return flight.flightNumber == flightNumber;

});

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

return \*it;

}

else {

cerr << "Рейс с указанным номером не найден." << endl;

return Aeroflot(0, "", "", "", "", 0); // Возвращаем пустой рейс

}

}

void addFlight(vector<Aeroflot>& flights) {

int flightNumber;

string departureCity, destinationCity, arrivalTime, departureTime;

int registrationSection;

cout << "Введите информацию о новом рейсе:" << endl;

cout << "Номер рейса: ";

cin >> flightNumber;

cout << "Пункт отправления: ";

cin >> departureCity;

cout << "Пункт назначения: ";

cin >> destinationCity;

cout << "Время прибытия: ";

cin >> arrivalTime;

cout << "Время отправления: ";

cin >> departureTime;

cout << "Номер секции для регистрации: ";

cin >> registrationSection;

flights.emplace\_back(flightNumber, departureCity, destinationCity,

arrivalTime, departureTime, registrationSection);

cout << "Рейс успешно добавлен." << endl;

}

int main() {

setlocale(LC\_ALL, "rus");

vector<Aeroflot> flights;

int numberOfFlights;

cout << "Введите количество рейсов: ";

cin >> numberOfFlights;

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

int flightNumber;

string departureCity, destinationCity, arrivalTime, departureTime;

int registrationSection;

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

cout << "Номер рейса: ";

cin >> flightNumber;

cout << "Пункт отправления: ";

cin >> departureCity;

cout << "Пункт назначения: ";

cin >> destinationCity;

cout << "Время прибытия: ";

cin >> arrivalTime;

cout << "Время отправления: ";

cin >> departureTime;

cout << "Номер секции для регистрации: ";

cin >> registrationSection;

flights.emplace\_back(flightNumber, departureCity, destinationCity,

arrivalTime, departureTime, registrationSection);

}

int choice;

do {

// Вывод меню

cout << "Меню:" << endl;

cout << "1. Показать все рейсы" << endl;

cout << "2. Выбрать рейс" << endl;

cout << "3. Добавить рейс" << endl;

cout << "0. Выйти" << endl;

cout << "Выберите опцию: ";

cin >> choice;

switch (choice) {

case 1:

showAllFlights(flights);

break;

case 2:

{

Aeroflot selectedFlight = chooseFlight(flights);

if (selectedFlight.flightNumber != 0) {

cout << "Выбранный рейс:" << endl;

selectedFlight.displayFlightInfo();

}

}

break;

case 3:

addFlight(flights);

break;

case 0:

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

default:

cout << "Некорректный ввод. Повторите попытку." << endl;

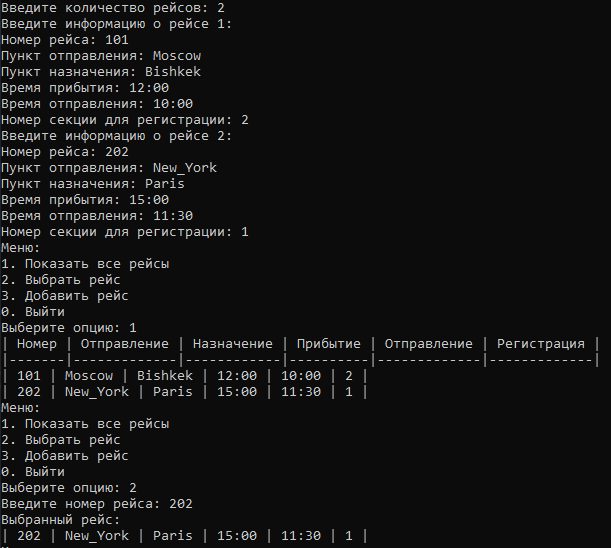
}

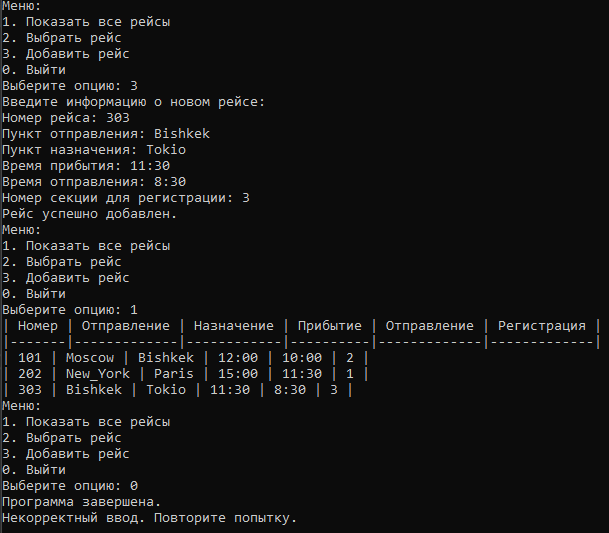
} while (choice != 0);

return 0;

}

***Результат:***





**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:***

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

class Student {

public:

string firstName;

string lastName;

string birthDate;

string phoneNumber;

Student(const string& first, const string& last, const string& birth, const string& phone)

: firstName(first), lastName(last), birthDate(birth), phoneNumber(phone) {}

void displayStudentInfo() const {

cout << "| " << firstName << " | " << lastName << " | " << birthDate

<< " | " << phoneNumber << " |" << endl;

}

};

class StudentGroup {

private:

vector<Student> students;

public:

void addStudent(const string& first, const string& last, const string& birth, const string& phone) {

students.emplace\_back(first, last, birth, phone);

cout << "Студент добавлен." << endl;

}

void removeStudent(const string& lastName) {

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

return student.lastName == lastName;

});

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

students.erase(it);

cout << "Студент удален." << endl;

}

else {

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

}

}

void searchStudentByLastName(const string& lastName) {

searchAndDisplayResults([lastName](const Student& student) {

return student.lastName == lastName;

}, "Студент с фамилией " + lastName + " не найден.");

}

void searchStudentByBirthDate(const string& birthDate) {

searchAndDisplayResults([birthDate](const Student& student) {

return student.birthDate == birthDate;

}, "Студент с датой рождения " + birthDate + " не найден.");

}

void searchStudentByPhoneNumber(const string& phoneNumber) {

searchAndDisplayResults([phoneNumber](const Student& student) {

return student.phoneNumber == phoneNumber;

}, "Студент с номером телефона " + phoneNumber + " не найден.");

}

void displayAllStudents() const {

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

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

for (const auto& student : students) {

student.displayStudentInfo();

}

if (students.empty()) {

cout << "Группа студентов пуста." << endl;

}

}

void sortByLastName() {

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

return a.lastName < b.lastName;

});

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

}

void sortByBirthDate() {

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

return a.birthDate < b.birthDate;

});

cout << "Студенты отсортированы по дате рождения." << endl;

}

void sortByPhoneNumber() {

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

return a.phoneNumber < b.phoneNumber;

});

cout << "Студенты отсортированы по номеру телефона." << endl;

}

private:

template <typename Predicate>

void searchAndDisplayResults(Predicate predicate, const string& notFoundMessage) {

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

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

cout << "Студент найден:" << endl;

it->displayStudentInfo();

}

else {

cerr << notFoundMessage << endl;

}

}

};

int main() {

setlocale(LC\_ALL, "rus");

StudentGroup group;

int choice;

do {

// Вывод меню

cout << "Меню:" << endl;

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

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

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

cout << "4. Найти студента по дате рождения" << endl;

cout << "5. Найти студента по номеру телефона" << endl;

cout << "6. Показать всех студентов" << endl;

cout << "7. Сортировать студентов по фамилии" << endl;

cout << "8. Сортировать студентов по дате рождения" << endl;

cout << "9. Сортировать студентов по номеру телефона" << endl;

cout << "0. Выйти" << endl;

cout << "Выберите опцию: ";

cin >> choice;

switch (choice) {

case 1: {

string firstName, lastName, birthDate, phoneNumber;

cout << "Введите данные студента:" << endl;

cout << "Имя: ";

cin >> firstName;

cout << "Фамилия: ";

cin >> lastName;

cout << "Дата рождения: ";

cin >> birthDate;

cout << "Номер телефона: ";

cin >> phoneNumber;

group.addStudent(firstName, lastName, birthDate, phoneNumber);

break;

}

case 2: {

string lastName;

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

cin >> lastName;

group.removeStudent(lastName);

break;

}

case 3: {

string lastName;

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

cin >> lastName;

group.searchStudentByLastName(lastName);

break;

}

case 4: {

string birthDate;

cout << "Введите дату рождения студента для поиска: ";

cin >> birthDate;

group.searchStudentByBirthDate(birthDate);

break;

}

case 5: {

string phoneNumber;

cout << "Введите номер телефона студента для поиска: ";

cin >> phoneNumber;

group.searchStudentByPhoneNumber(phoneNumber);

break;

}

case 6:

group.displayAllStudents();

break;

case 7:

group.sortByLastName();

break;

case 8:

group.sortByBirthDate();

break;

case 9:

group.sortByPhoneNumber();

break;

case 0:

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

break;

default:

cout << "Некорректный ввод. Повторите попытку." << endl;

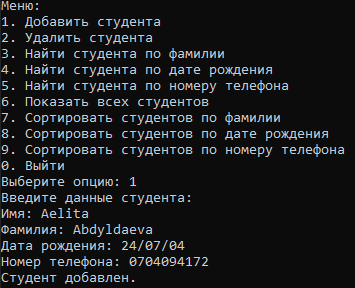
}

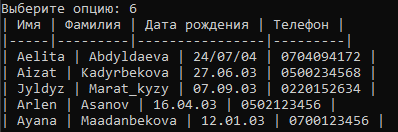
} while (choice != 0);

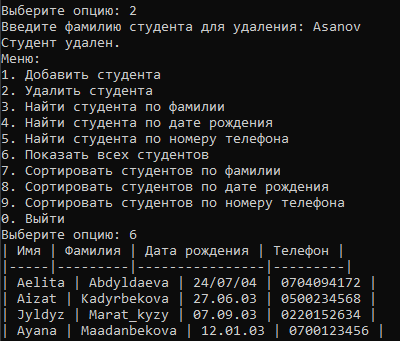
return 0;

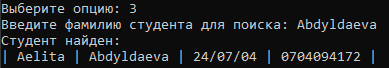
}

***Результат:***

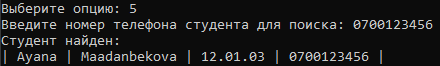




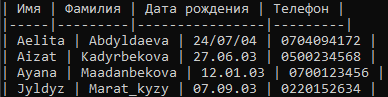


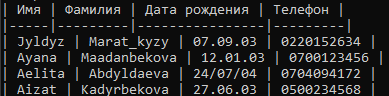




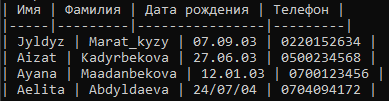












**Задание №3. Опишите класс, реализующий стек (Stack).**

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>

using namespace std;

class TrainSortingNode {

private:

stack<string> leftDirection;

stack<string> rightDirection;

public:

void addWagonToStack(const string& wagon, bool isLeftDirection) {

if (isLeftDirection) {

leftDirection.push(wagon);

}

else {

rightDirection.push(wagon);

}

}

void moveWagons() {

cout << "Перемещение вагонов:" << endl;

while (!leftDirection.empty()) {

cout << "Из левого направления: " << leftDirection.top() << endl;

leftDirection.pop();

}

while (!rightDirection.empty()) {

cout << "Из правого направления: " << rightDirection.top() << endl;

rightDirection.pop();

}

}

// Добавление вагонов из файла

void addWagonsFromFile(const string& filename, bool isLeftDirection) {

ifstream inputFile(filename);

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

cerr << "Ошибка открытия файла: " << filename << endl;

return;

}

string wagon;

while (inputFile >> wagon) {

addWagonToStack(wagon, isLeftDirection);

}

inputFile.close();

cout << "Вагоны добавлены из файла: " << filename << endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

TrainSortingNode sortingNode;

int choice;

do {

// Вывод меню

cout << "Меню:" << endl;

cout << "1. Добавить вагон в левое направление" << endl;

cout << "2. Добавить вагон в правое направление" << endl;

cout << "3. Переместить вагоны" << endl;

cout << "4. Добавить вагоны из файла в левое направление" << endl;

cout << "5. Добавить вагоны из файла в правое направление" << endl;

cout << "0. Выйти" << endl;

cout << "Выберите опцию: ";

cin >> choice;

switch (choice) {

case 1: {

string wagon;

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

cin >> wagon;

sortingNode.addWagonToStack(wagon, true);

break;

}

case 2: {

string wagon;

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

cin >> wagon;

sortingNode.addWagonToStack(wagon, false);

break;

}

case 3:

sortingNode.moveWagons();

break;

case 4:

sortingNode.addWagonsFromFile("wagons.txt", true);

break;

case 5:

sortingNode.addWagonsFromFile("wagons.txt", false);

break;

case 0:

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

break;

default:

cout << "Некорректный ввод. Повторите попытку." << endl;

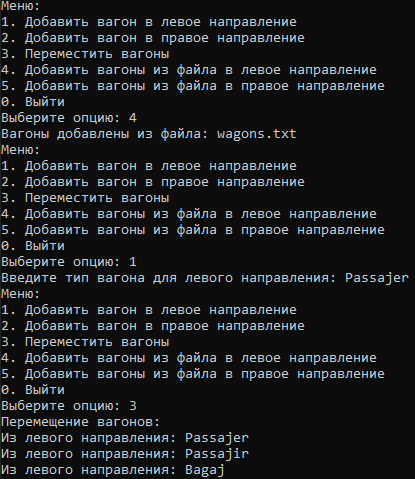
}

} while (choice != 0);

return 0;

}

***Результат:***

******