Министерство образования и науки Кыргызской Республики

Кыргызский государственный технический университет

им. И.Раззакова

Институт Совместных Образовательных Программ

Кафедра «Программное обеспечение компьютерных систем»

Направление: 710400 «Программная инженерия»

Дисциплина: «Объектно-ориентированное проектирование и программирование ПО»

ОТЧЕТ

По лабораторной работе №2

Выполнил: студент группы

ПИ(б)-ИСОП 1-21 Именов Алишер

Проверила: ст. пр. 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.

#include <iostream>

#include <vector>

#include <algorithm>

class Aeroflot {

public:

int flightNumber;

std::string departureCity;

std::string destinationCity;

std::string arrivalTime;

std::string departureTime;

int registrationSection;

// Конструктор класса

Aeroflot(int flightNumber, const std::string& departureCity, const std::string& destinationCity,

const std::string& arrivalTime, const std::string& departureTime, int registrationSection)

: flightNumber(flightNumber), departureCity(departureCity), destinationCity(destinationCity),

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

// Функция вывода информации о рейсе

void displayFlightInfo() const {

std::cout << flightNumber << "\t" << departureCity << "\t" << destinationCity << "\t"

<< arrivalTime << "\t" << departureTime << "\t" << registrationSection << std::endl;

}

};

// Функция сравнения для сортировки в алфавитном порядке по пункту назначения

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

return a.destinationCity < b.destinationCity;

}

int main() {

setlocale(LC\_ALL, "rus");

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

// Ввод данных о рейсах с клавиатуры

int numFlights;

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

std::cin >> numFlights;

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

int flightNumber;

std::string departureCity, destinationCity, arrivalTime, departureTime;

int registrationSection;

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

std::cout << "Номер рейса: ";

std::cin >> flightNumber;

std::cout << "Пункт отправления: ";

std::cin >> departureCity;

std::cout << "Пункт назначения: ";

std::cin >> destinationCity;

std::cout << "Время прибытия: ";

std::cin >> arrivalTime;

std::cout << "Время отправления: ";

std::cin >> departureTime;

std::cout << "Место регистрации: ";

std::cin >> registrationSection;

flights.emplace\_back(flightNumber, departureCity, destinationCity, arrivalTime, departureTime, registrationSection);

}

// Сортировка рейсов по пункту назначения

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

// Вывод таблицы рейсов

std::cout << "Номер\tОтправление\tНазначение\tПрибытие\tОтправление\tРегистрация" << std::endl;

for (const auto& flight : flights) {

flight.displayFlightInfo();

}

// Проверка наличия рейсов

if (flights.empty()) {

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

}

return 0;

}

**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.

#include <iostream>

#include <vector>

#include <algorithm>

#include <iomanip>

class Student {

public:

std::string name;

std::string surname;

std::string dateOfBirth;

std::string phoneNumber;

// Конструктор класса

Student(const std::string& name, const std::string& surname,

const std::string& dateOfBirth, const std::string& phoneNumber)

: name(name), surname(surname), dateOfBirth(dateOfBirth), phoneNumber(phoneNumber) {}

};

class StudentDatabase {

private:

std::vector<Student> students;

public:

// Добавление студента в базу данных

void addStudent(const Student& student) {

students.push\_back(student);

}

// Удаление студента из базы данных по фамилии

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

students.erase(std::remove\_if(students.begin(), students.end(),

[surname](const Student& student) {

return student.surname == surname;

}),

students.end());

}

// Поиск студента по фамилии

Student\* findStudentBySurname(const std::string& surname) {

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

[surname](const Student& student) {

return student.surname == surname;

});

return (it != students.end()) ? &(\*it) : nullptr;

}

// Сортировка студентов по фамилии

void sortStudentsBySurname() {

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

[](const Student& a, const Student& b) {

return a.surname < b.surname;

});

}

// Вывод всех студентов

void displayAllStudents() const {

if (students.empty()) {

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

return;

}

std::cout << std::setw(15) << "Фамилия" << std::setw(15) << "Имя"

<< std::setw(15) << "Дата рождения" << std::setw(15) << "Телефон" << std::endl;

for (const auto& student : students) {

std::cout << std::setw(15) << student.surname << std::setw(15) << student.name

<< std::setw(15) << student.dateOfBirth << std::setw(15) << student.phoneNumber << std::endl;

}

}

};

int main() {

setlocale(LC\_ALL, "rus");

StudentDatabase database;

int choice;

std::string surname;

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. Отсортировать студентов по фамилии\n";

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

std::cout << "0. Выход\n";

std::cout << "Выберите действие: ";

std::cin >> choice;

switch (choice) {

case 1: {

std::string name, dateOfBirth, phoneNumber;

std::cout << "Введите имя: ";

std::cin >> name;

std::cout << "Введите фамилию: ";

std::cin >> surname;

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

std::cin >> dateOfBirth;

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

std::cin >> phoneNumber;

database.addStudent(Student(name, surname, dateOfBirth, phoneNumber));

break;

}

case 2: {

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

std::cin >> surname;

database.removeStudent(surname);

break;

}

case 3: {

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

std::cin >> surname;

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

if (foundStudent) {

std::cout << "Студент найден:\n";

std::cout << "Имя: " << foundStudent->name << "\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.sortStudentsBySurname();

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

break;

}

case 5: {

database.displayAllStudents();

break;

}

case 0:

std::cout << "Выход.\n";

break;

default:

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

}

} while (choice != 0);

return 0;

}

**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;

#include <iostream>

#include <fstream>

#include <stack>

#include <vector>

class TrainSorter {

private:

std::stack<int> leftDirection; // Стек для вагонов первого типа

std::stack<int> rightDirection; // Стек для вагонов второго типа

public:

// Добавление вагона в узел

void addWagon(int wagonType) {

if (wagonType == 1) {

leftDirection.push(wagonType);

}

else if (wagonType == 2) {

rightDirection.push(wagonType);

}

}

// Вывод вагонов в обоих направлениях

void displayTrains() const {

std::cout << "Вагоны в левом направлении: ";

displayStack(leftDirection);

std::cout << "Вагоны в правом направлении: ";

displayStack(rightDirection);

}

private:

// Вспомогательная функция для вывода стека

void displayStack(const std::stack<int>& s) const {

std::stack<int> temp = s;

while (!temp.empty()) {

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

temp.pop();

}

std::cout << std::endl;

}

};

int main() {

setlocale(LC\_ALL, "rus");

TrainSorter sorter;

// Пример ввода из файла

/\*std::ifstream inputFile("train\_input.txt");

if (inputFile.is\_open()) {

int wagonType;

while (inputFile >> wagonType) {

sorter.addWagon(wagonType);

}

inputFile.close();

}

else {

std::cerr << "Не удалось открыть файл для чтения." << std::endl;

return 1;

}\*/

//Пример ввода с клавиатуры

std::cout << "Введите типы вагонов (1 или 2), завершите ввод 0: ";

int wagonType;

while (std::cin >> wagonType && wagonType != 0) {

sorter.addWagon(wagonType);

}

// Вывод состава по направлениям

sorter.displayTrains();

return 0;

}