**Ministry of education and science of the Kyrgyz Republic**

**Kyrgyz State Technical University named after I.Razzakov**

**Faculty of Information Technologies**

**Department of Software of Computer Systems**

**Major: 710400 «Software Engineering»**

Report

Discipline: «**Object-Oriented Design**»

Software requirements

Done by**:** Idirisov Aziret SE(eng)- 1- 21

Checked by: Musabaev E.B.

Bishkek 2024

Task №1

Функциональные возможности программы управления информацией о рейсах:

Ввод информации о рейсах:

Пользователь может ввести количество рейсов.

Для каждого рейса пользователь может ввести следующие данные:

Номер рейса.

Пункт отправления.

Пункт назначения.

Время прибытия.

Время отправления.

Место регистрации.

Проверка уникальности информации о рейсах:

Программа проверяет, что номер рейса и время отправления являются уникальными для каждого рейса.

Если введены неуникальные номер рейса или время отправления, программа запрашивает повторный ввод этих значений.

Отображение информации о рейсах:

После ввода всех данных программа сортирует рейсы по пункту назначения и выводит таблицу с информацией о рейсах.

Таблица содержит следующие колонки: номер рейса, пункт отправления, пункт назначения, время прибытия, время отправления, место регистрации.

Если не было введено ни одного рейса, программа выводит сообщение о том, что рейсы недоступны.

Управление информацией о рейсах:

Пользователь может вводить информацию о нескольких рейсах одновременно.

Интерактивный интерфейс:

Программа взаимодействует с пользователем через консольный интерфейс, запрашивая ввод информации о рейсах и выводя результаты на экран.

#include <iostream>

#include <iomanip>

#include <vector>

#include <algorithm>

using namespace std;

class Aeroflot {

private:

int flightNumber;

string departurePoint;

string destinationPoint;

string arrivalTime;

string departureTime;

string checkInLocation;

public:

Aeroflot(int num, const string& dep, const string& dest,

const string& arrTime, const string& depTime,

const string& checkIn)

: flightNumber(num), departurePoint(dep), destinationPoint(dest),

arrivalTime(arrTime), departureTime(depTime), checkInLocation(checkIn) {}

void displayFlightInfo() const {

cout << setw(8) << flightNumber

<< setw(15) << departurePoint

<< setw(15) << destinationPoint

<< setw(15) << arrivalTime

<< setw(15) << departureTime

<< setw(20) << checkInLocation << endl;

}

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; }

string getCheckInLocation() const { return checkInLocation; }

};

bool compareFlights(const Aeroflot& flight1, const Aeroflot& flight2) {

return flight1.getDestinationPoint() < flight2.getDestinationPoint();

}

bool isFlightNumberUnique(const vector<Aeroflot>& flights, int num) {

for (const auto& flight : flights) {

if (flight.getFlightNumber() == num) {

return false; // Найден дубликат номера рейса

}

}

return true;

}

bool isTimeUnique(const vector<Aeroflot>& flights, const string& depTime) {

for (const auto& flight : flights) {

if (flight.getDepartureTime() == depTime) {

return false; // Найден дубликат времени отправления

}

}

return true;

}

int main() {

vector<Aeroflot> flights;

int numFlights;

cout << "Enter the number of flights: ";

cin >> numFlights;

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

int num;

string dep, dest, arrTime, depTime, checkIn;

// Проверка уникальности номера рейса

do {

cout << "Flight Number: ";

cin >> num;

} while (!isFlightNumberUnique(flights, num));

// Проверка уникальности времени отправления

do {

cout << "Departure Time: ";

cin >> depTime;

} while (!isTimeUnique(flights, depTime));

cout << "Departure Point: ";

cin >> dep;

cout << "Destination Point: ";

cin >> dest;

cout << "Arrival Time: ";

cin >> arrTime;

cout << "Check-In Location: ";

cin >> checkIn;

flights.emplace\_back(num, dep, dest, arrTime, depTime, checkIn);

}

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

cout << "\nFlight Information Table:\n";

cout << setw(8) << "Flight#"

<< setw(15) << "Departure"

<< setw(15) << "Destination"

<< setw(15) << "Arrival Time"

<< setw(15) << "Departure Time"

<< setw(20) << "Check-In Location\n";

if (flights.empty()) {

cout << "No flights available.\n";

} else {

for (const auto& flight : flights) {

flight.displayFlightInfo();

}

}

return 0;

}

Task №2

Основные функциональные возможности программы:

Добавление студента: Пользователь может добавить нового студента, вводя его фамилию, имя, дату рождения и номер телефона. Программа проверяет уникальность номера телефона и добавляет студента в группу.

Поиск студента по фамилии: Пользователь может найти студента, введя его фамилию. Программа выводит информацию о всех студентах с заданной фамилией.

Удаление студента: Пользователь может удалить студента из группы, введя его фамилию. Программа удаляет всех студентов с заданной фамилией.

Сортировка студентов по фамилии: Пользователь может отсортировать всех студентов в группе по фамилии в алфавитном порядке.

Отображение всех студентов: Пользователь может просмотреть информацию о всех студентах в группе, включая фамилию, имя, дату рождения и номер телефона.

Редактирование информации о студенте: Пользователь может изменить информацию о студенте, введя его фамилию. Программа позволяет изменить фамилию, имя, дату рождения и номер телефона выбранного студента.

Выход из программы: Пользователь может завершить выполнение программы.

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using namespace std;

class Student {

private:

string lastName;

string firstName;

string dateOfBirth;

string phoneNumber;

public:

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

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

const string& getLastName() const {

return lastName;

}

void setLastName(const string& last) {

lastName = last;

}

const string& getFirstName() const {

return firstName;

}

void setFirstName(const string& first) {

firstName = first;

}

const string& getDateOfBirth() const {

return dateOfBirth;

}

void setDateOfBirth(const string& dob) {

dateOfBirth = dob;

}

const string& getPhoneNumber() const {

return phoneNumber;

}

void setPhoneNumber(const string& phone) {

phoneNumber = phone;

}

};

class StudentGroup {

private:

vector<Student> students;

public:

void addStudent(const Student& student) {

for (const auto& s : students) {

if (s.getPhoneNumber() == student.getPhoneNumber()) {

cout << "Error: Student with the same phone number already exists." << endl;

return;

}

}

students.push\_back(student);

}

void removeStudent(const string& lastName) {

students.erase(

remove\_if(students.begin(), students.end(),

[&lastName](const Student& student) {

return student.getLastName() == lastName;

}),

students.end());

}

vector<Student> findStudentByLastName(const string& lastName) {

vector<Student> foundStudents;

for (const auto& student : students) {

if (student.getLastName() == lastName) {

foundStudents.push\_back(student);

}

}

return foundStudents;

}

void sortByLastName() {

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

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

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

});

}

void displayStudents() {

for (const auto& student : students) {

cout << "Last Name: " << student.getLastName() << ", "

<< "First Name: " << student.getFirstName() << ", "

<< "Date of Birth: " << student.getDateOfBirth() << ", "

<< "Phone Number: " << student.getPhoneNumber() << endl;

}

}

void editStudentInfo(const string& lastName) {

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

[&lastName](const Student& student) {

return student.getLastName() == lastName;

});

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

cout << "Editing student: " << it->getLastName() << endl;

string newLastName, newFirstName, newDateOfBirth, newPhoneNumber;

cout << "Enter new Last Name: ";

cin >> newLastName;

cout << "Enter new First Name: ";

cin >> newFirstName;

cout << "Enter new Date of Birth: ";

cin >> newDateOfBirth;

cout << "Enter new Phone Number: ";

cin >> newPhoneNumber;

it->setLastName(newLastName);

it->setFirstName(newFirstName);

it->setDateOfBirth(newDateOfBirth);

it->setPhoneNumber(newPhoneNumber);

cout << "Student information updated." << endl;

} else {

cout << "Student with last name " << lastName << " not found." << endl;

}

}

};

Student inputStudentData() {

string lastName, firstName, dateOfBirth, phoneNumber;

cout << "Enter Last Name: ";

cin >> lastName;

cout << "Enter First Name: ";

cin >> firstName;

cout << "Enter Date of Birth: ";

cin >> dateOfBirth;

cout << "Enter Phone Number: ";

cin >> phoneNumber;

return Student(lastName, firstName, dateOfBirth, phoneNumber);

}

int main() {

StudentGroup group;

char choice;

do {

cout << "\nMain Menu:" << endl;

cout << "1. Add Student" << endl;

cout << "2. Find Student by Last Name" << endl;

cout << "3. Remove Student" << endl;

cout << "4. Sort Students by Last Name" << endl;

cout << "5. Display All Students" << endl;

cout << "6. Edit Student Information" << endl;

cout << "7. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case '1': {

group.addStudent(inputStudentData());

break;

}

case '2': {

string lastName;

cout << "Enter Last Name to search: ";

cin >> lastName;

vector<Student> foundStudents = group.findStudentByLastName(lastName);

if (!foundStudents.empty()) {

cout << "Found students:" << endl;

for (const auto& student : foundStudents) {

cout << "Last Name: " << student.getLastName() << ", "

<< "First Name: " << student.getFirstName() << ", "

<< "Date of Birth: " << student.getDateOfBirth() << ", "

<< "Phone Number: " << student.getPhoneNumber() << endl;

}

} else {

cout << "No students found with last name " << lastName << endl;

}

break;

}

case '3': {

string lastName;

cout << "Enter Last Name to remove: ";

cin >> lastName;

group.removeStudent(lastName);

break;

}

case '4': {

group.sortByLastName();

cout << "Students sorted by last name." << endl;

break;

}

case '5': {

cout << "All students in the group:" << endl;

group.displayStudents();

break;

}

case '6': {

string lastName;

cout << "Enter Last Name of student to edit: ";

cin >> lastName;

group.editStudentInfo(lastName);

break;

}

case '7': {

cout << "Exiting program." << endl;

break;

}

default: {

cout << "Invalid choice. Please enter a number from 1 to 7." << endl;

}

}

} while (choice != '7');

return 0;

}

Task №3

Цель программы - моделирование Т-образного сортировочного узла на железной дороге с использованием структуры данных стек. Программа разделяет вагоны на два направления (A и B) и выводит их в порядке, соответствующем вводу пользователя или содержимому файла.

Логика программы:

Создание класса Stack для реализации стека.

Чтение вагонов из файла "train.txt", если файл доступен, и заполнение стеков для каждого направления в соответствии с типом вагона.

Запрос ввода дополнительных вагонов с клавиатуры, если чтение из файла не удалось или пользователь хочет добавить вагоны.

Вывод вагонов для каждого направления по очереди согласно стеку, начиная с верхнего вагона.

Таким образом, программа позволяет моделировать сортировочный узел на железной дороге, сортируя вагоны по двум направлениям и выводя их в соответствующем порядке.

#include <iostream>

#include <stack>

#include <fstream>

#include <string>

using namespace std;

class Stack {

private:

stack<int> data;

public:

void push(int value) {

data.push(value);

}

int pop() {

int top\_value = data.top();

data.pop();

return top\_value;

}

bool isEmpty() {

return data.empty();

}

};

int main() {

Stack stack1, stack2;

ifstream file("train.txt");

if (file.is\_open()) {

int wagon;

while (file >> wagon) {

if (wagon == 1) {

stack1.push(wagon);

} else if (wagon == 2) {

stack2.push(wagon);

} else {

cout << "Invalid wagon type: " << wagon << endl;

}

}

file.close();

} else {

cout << "Failed to open file." << endl;

}

cout << "Enter wagon numbers (1 for type A, 2 for type B) or -1 to finish input:" << endl;

int wagon;

while (true) {

cin >> wagon;

if (wagon == -1) {

break;

} else if (wagon == 1) {

stack1.push(wagon);

} else if (wagon == 2) {

stack2.push(wagon);

} else {

cout << "Invalid wagon type: " << wagon << endl;

}

}

cout << "Wagons in direction A:" << endl;

while (!stack1.isEmpty()) {

cout << stack1.pop() << endl;

}

cout << "Wagons in direction B:" << endl;

while (!stack2.isEmpty()) {

cout << stack2.pop() << endl;

}

return 0;

}

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

Класс - это шаблон для создания объектов. Объект - экземпляр класса. Пример: класс "Автомобиль" и объекты "Машина1", "Машина2".

2) Каков синтаксис и смысл определения класса? Как называются составные части класса?

Синтаксис: class Название\_класса:. Смысл: определение структуры данных с методами и свойствами. Составные части: атрибуты и методы.

3) Что такое идентификаторы доступа к элементам класса? Дайте определения.

Идентификаторы доступа - это специальные ключевые слова, определяющие уровень доступа к атрибутам и методам класса. Например, public, private, protected.

4) Что такое инкапсуляция?

Инкапсуляция - это принцип ООП, заключающийся в объединении данных и методов, работающих с этими данными, в единый объект.

5) Где в программе с классами можно создавать объекты?

Объекты классов можно создавать в любом месте программы, где доступен код класса.

6) Каков смысл объявления объектов?

Объявление объектов предоставляет возможность использовать функционал класса для работы с данными.

7) В какой момент метод готов для вызова объектом?

Метод готов для вызова объектом, когда объект создан и готов принимать команды.

8) Как называются методы, встроенные в структуру класса?

Методы, встроенные в структуру класса, называются методами класса или статическими методами.

9) Как определяется метод, если внутри класса записан только его заголовок, сам метод определен вне класса?

Метод определяется вне класса как обычная функция, указывая имя класса как префикс. Например: Класс.метод().

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

Каждому объекту принадлежит копия каждого поля данных. Имена и значения могут отличаться для каждого объекта.

11) Тиражируются ли методы класса?

Методы класса могут быть тиражированы, то есть существовать в единственном экземпляре для всех объектов класса.

12) Как в программе написать доступ к открытым и закрытым полям класса?

Доступ к открытым полям класса осуществляется напрямую по имени поля. Доступ к закрытым полям осуществляется через методы класса.