**USMAN ASHFAQ**

**23F-0706**

**OOP Lab# 04**

**Q#01:**

#include <iostream>

#include <string>

using namespace std;

struct Time {

int hours;

int minutes;

};

struct Flight {

string flight\_num;

string orig;

string destination;

Time departure\_time;

Time arrival\_time;

bool seats\_ava[10][6];

int flight\_duration;

float price;

};

struct Airline {

string entername;

string pass;

int fleet\_size;

Flight flights[100];

int numFlights;

};

struct Customer {

string user\_name;

string id\_Number;

string mobile\_num;

string email\_address;

string booking\_reference;

Flight\* booked\_Flight;

int seat\_row, seat\_col;

string address;

string passport\_number;

};

const int MAX\_AIRLINES = 10;

const int MAX\_CUSTOMERS = 100;

Airline airlines[MAX\_AIRLINES];

int totalAirlines = 0;

Customer customers[MAX\_CUSTOMERS];

int totalCustomers = 0;

void initializeSeats(bool seats[10][6]) {

for (int i = 0; i < 10; ++i) {

for (int j = 0; j < 6; ++j) {

seats[i][j] = true;

}

}

}

// Add a new airline

void addNewAirline() {

if (totalAirlines >= MAX\_AIRLINES) {

cout << "Cannot add more airlines. Maximum limit reached.\n";

return;

}

Airline& airline = airlines[totalAirlines];

cout << "Enter the name of the airline: ";

cin.ignore();

getline(cin, airline.entername);

cout << "Enter the airline code (2-4 characters): ";

cin >> airline.pass;

cout << "Enter the fleet size: ";

cin >> airline.fleet\_size;

airline.numFlights = 0; // No flights initially

totalAirlines++;

cout << "Airline added successfully!\n";

}

// Schedule a flight for an airline

void scheduleFlightForAirline() {

cout << "Enter airline code to schedule a flight for: ";

string airlineCode;

cin >> airlineCode;

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

if (airlines[i].pass == airlineCode) {

Airline& airline = airlines[i];

if (airline.numFlights >= 100) {

cout << "Cannot add more flights. Maximum limit reached.\n";

return;

}

Flight& flight = airline.flights[airline.numFlights];

cout << "Enter flight number: ";

cin >> flight.flight\_num;

cout << "Enter origin (3-letter airport code): ";

cin >> flight.orig;

cout << "Enter destination (3-letter airport code): ";

cin >> flight.destination;

cout << "Enter departure time (hours and minutes): ";

cin >> flight.departure\_time.hours >> flight.departure\_time.minutes;

cout << "Enter arrival time (hours and minutes): ";

cin >> flight.arrival\_time.hours >> flight.arrival\_time.minutes;

cout << "Enter flight duration (in minutes): ";

cin >> flight.flight\_duration;

cout << "Enter ticket price: ";

cin >> flight.price;

initializeSeats(flight.seats\_ava); // Initialize seat availability

airline.numFlights++;

cout << "Flight scheduled successfully!\n";

return;

}

}

cout << "Airline not found.\n";

}

// Book a flight

void bookFlight() {

cout << "Enter origin (3-letter code): ";

string origin;

cin >> origin;

cout << "Enter destination (3-letter code): ";

string destination;

cin >> destination;

// Find available flights

Flight\* availableFlight = nullptr;

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

for (int j = 0; j < airlines[i].numFlights; ++j) {

if (airlines[i].flights[j].orig == origin &&

airlines[i].flights[j].destination == destination) {

availableFlight = &airlines[i].flights[j];

break;

}

}

if (availableFlight) break;

}

if (!availableFlight) {

cout << "No flights available for the given route.\n";

return;

}

if (totalCustomers >= MAX\_CUSTOMERS) {

cout << "Cannot add more customers. Maximum limit reached.\n";

return;

}

// Book a seat

Customer& customer = customers[totalCustomers];

cout << "Enter customer name: ";

cin.ignore();

getline(cin, customer.user\_name);

cout << "Enter customer ID (passport or other): ";

cin >> customer.id\_Number;

cout << "Enter mobile number: ";

cin >> customer.mobile\_num;

cout << "Enter email address: ";

cin >> customer.email\_address;

// Additional information

cout << "Enter customer address: ";

cin.ignore();

getline(cin, customer.address);

cout << "Enter passport number: ";

cin >> customer.passport\_number;

// Find an available seat

bool seatFound = false;

for (int row = 0; row < 10; ++row) {

for (int col = 0; col < 6; ++col) {

if (availableFlight->seats\_ava[row][col]) {

availableFlight->seats\_ava[row][col] = false; // Book the seat

customer.booked\_Flight = availableFlight;

customer.seat\_row = row;

customer.seat\_col = col;

customer.booking\_reference = customer.id\_Number + "-FLIGHT";

seatFound = true;

break;

}

}

if (seatFound) break;

}

if (seatFound) {

totalCustomers++;

cout << "Flight booked successfully! Your booking reference is: "

<< customer.booking\_reference << endl;

}

else {

cout << "No seats available on the selected flight.\n";

}

}

// View seating arrangement

void viewSeatingArrangement() {

cout << "Enter flight number to view seating arrangement: ";

string flightNum;

cin >> flightNum;

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

for (int j = 0; j < airlines[i].numFlights; ++j) {

if (airlines[i].flights[j].flight\_num == flightNum) {

Flight& flight = airlines[i].flights[j];

cout << "Seating arrangement for flight " << flight.flight\_num

<< ":\n";

for (int row = 0; row < 10; ++row) {

for (int col = 0; col < 6; ++col) {

cout << (flight.seats\_ava[row][col] ? 'O' : 'X')

<< " ";

}

cout << endl;

}

return;

}

}

}

cout << "Flight not found.\n";

}

// Calculate total sales

void calculateTotalSales() {

cout << "Enter flight number to calculate total sales: ";

string flightNum;

cin >> flightNum;

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

for (int j = 0; j < airlines[i].numFlights; ++j) {

if (airlines[i].flights[j].flight\_num == flightNum) {

Flight& flight = airlines[i].flights[j];

float totalSales = 0;

for (int row = 0; row < 10; ++row) {

for (int col = 0; col < 6; ++col) {

if (!flight.seats\_ava[row][col]) {

totalSales += flight.price;

}

}

}

cout << "Total sales for flight " << flight.flight\_num << ":$" << totalSales << endl;

return;

}

}

}

cout << "Flight not found.\n";

}

int main() {

int choice;

do {

cout << "\n Airline Reservation System \n";

cout << "1. Add New Airline\n";

cout << "2. Schedule Flight for an Airline\n";

cout << "3. Book a Flight\n";

cout << "4. View Seating Arrangement\n";

cout << "5. Cancel a Booking\n";

cout << "6. Search Flights\n";

cout << "7. Calculate Total Sales\n";

cout << "8. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

addNewAirline();

break;

case 2:

scheduleFlightForAirline();

break;

case 3:

bookFlight();

break;

case 4:

viewSeatingArrangement();

break;

case 5:

break;

case 6:

break;

case 7:

calculateTotalSales();

break;

case 8:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

break;

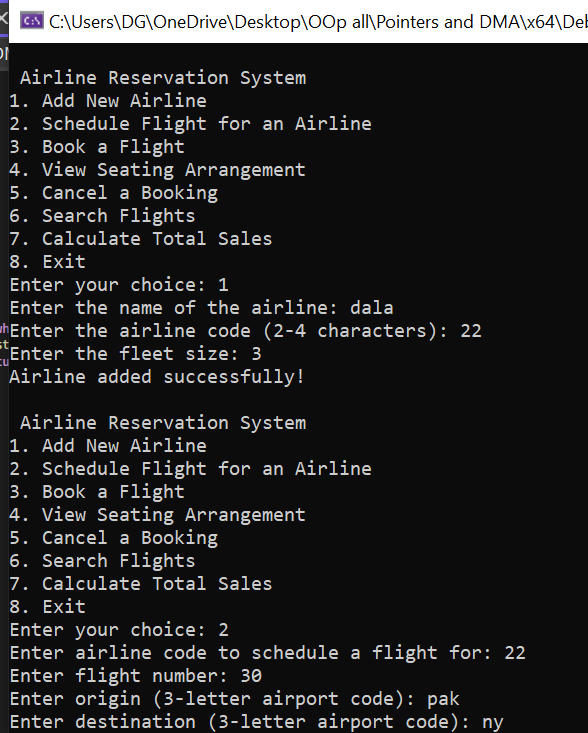
}

} while (choice != 8);

system("pause");

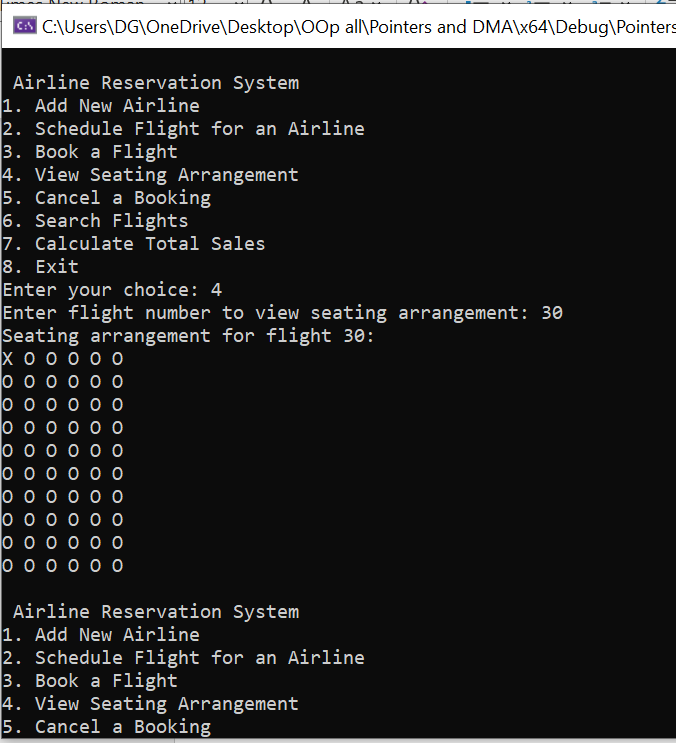
return 0;

}



A screenshot of a computer program

Description automatically generated



A screen shot of a computer

Description automatically generated

**TASK#02:**

#include <iostream>

using namespace std;

enum SensorType {

Temperature,

Pressure,

Humidity

};

union SensorData {

float temperature;

long pressure;

double humidity;

};

struct Sensor {

int sensorID;

SensorType type;

SensorData data;

};

void displaySensorData(const Sensor& sensor) {

cout << "Sensor ID: " << sensor.sensorID << endl;

switch (sensor.type) {

case Temperature:

cout << "Temperature: " << sensor.data.temperature << "°C" << endl;

break;

case Pressure:

cout << "Pressure: " << sensor.data.pressure << " Pascals" << endl;

break;

case Humidity:

cout << "Humidity: " << sensor.data.humidity << "%" << endl;

break;

default:

cout << "Invalid sensor type." << endl;

break;

}

}

int main() {

Sensor sensor1 = { 1, Temperature, { 23.5f } };

Sensor sensor2 = { 2, Pressure, { 101325 } };

Sensor sensor3 = { 3, Humidity, { 45.0 } };

cout << "Sensor 1 Data:" << endl;

displaySensorData(sensor1);

cout << endl;

cout << "Sensor 2 Data:" << endl;

displaySensorData(sensor2);

cout << endl;

cout << "Sensor 3 Data:" << endl;

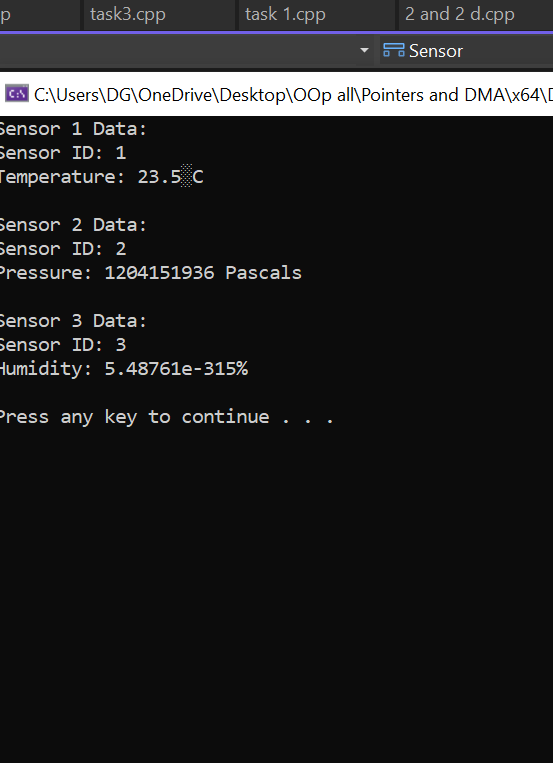
displaySensorData(sensor3);

cout << endl;

system("pause");

return 0;

}



**TASK #03:**

#include <iostream>

using namespace std;

enum weekdays {

Monday = 1,

Tuesday,

Wednesday,

Thursday,

Friday,

Saturday,

Sunday

};

int main() {

int day;

cout << "Monday\n";

cout << "Tuesday" << endl;

cout << "Wednesday" << endl;

cout << "Thursday" << endl;

cout << "Friday" << endl;

cout << "Saturday" << endl;

cout << "Sunday" << endl;

cout << "Enter the day: ";

cin >> day;

if (day < 1 || day > 7) {

cout << "Invalid input. Please enter a number between 1 and 7." << endl;

}

else {

switch (day) {

case Monday:

case Tuesday:

case Thursday:

cout << "Work day - Time to be productive!" << endl;

break;

case Wednesday:

cout << "Mid-week review - Evaluate your progress!" << endl;

break;

case Friday:

cout << "Casual work day - Dress casual, stay focused!" << endl;

break;

case Saturday:

case Sunday:

cout << "Weekend relaxation - Enjoy your free time!" << endl;

break;

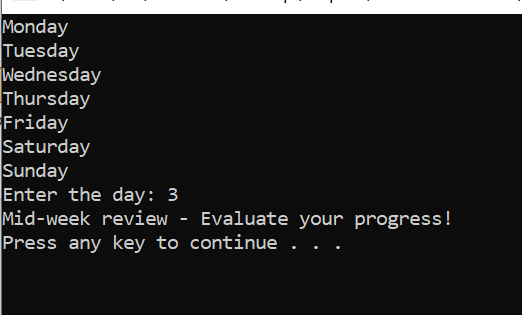
}

}

system("pause");

return 0;

}



**TASK#04:**

#include <iostream>

#include <string>

using namespace std;

enum VehicleType {

TRUCK,

CAR,

MOTORBIKE

};

union MaintenanceData {

float fuelConsumption;

int tirePressure;

float engineTemperature;

};

struct Vehicle {

int id;

int type;

MaintenanceData data;

};

void displayMaintenance(const Vehicle& vehicle) {

cout << "Vehicle ID: " << vehicle.id << endl;

switch (vehicle.type) {

case TRUCK:

cout << "Vehicle Type: Truck" << endl;

cout << "Fuel Consumption: " << vehicle.data.fuelConsumption << " liters per 100 km" << endl;

break;

case CAR:

cout << "Vehicle Type: Car" << endl;

cout << "Tire Pressure: " << vehicle.data.tirePressure << " PSI" << endl;

break;

case MOTORBIKE:

cout << "Vehicle Type: Motorbike" << endl;

cout << "Engine Temperature: " << vehicle.data.engineTemperature << " °C"

<< endl;

break;

default:

cout << "Unknown Vehicle Type" << endl;

break;

}

}

int main() {

Vehicle vehicles[3];

for (int i = 0; i < 3; ++i) {

cout << "Enter details for vehicle " << i + 1 << ":\n";

cout << "Vehicle ID: ";

cin >> vehicles[i].id;

int type;

cout << "Vehicle Type (0 for Truck, 1 for Car, 2 for Motorbike): ";

cin >> type;

vehicles[i].type = type;

switch (vehicles[i].type) {

case TRUCK:

cout << "Enter fuel consumption (liters per 100 km): ";

cin >> vehicles[i].data.fuelConsumption;

break;

case CAR:

cout << "Enter tire pressure (PSI): ";

cin >> vehicles[i].data.tirePressure;

break;

case MOTORBIKE:

cout << "Enter engine temperature (°C): ";

cin >> vehicles[i].data.engineTemperature;

break;

default:

cout << "Invalid vehicle type entered." << endl;

break;

}

}

cout << "\nMaintenance Records:\n";

for (int i = 0; i < 3; ++i) {

displayMaintenance(vehicles[i]);

cout << endl;

}

system("pause");

return 0;

}

