

PF Assignment#4

Tour management system(c++)

BY:

Syed Ghayur hussian (231570) and Huzaifa bin asaad(231654)

TO:

Dr. Ashfaq Hussain Farooqi

LinkedIn:

<https://www.linkedin.com/in/ghayur-hussian-1155172a6/>

Date: 30 December 2023

Description:

* There are 3 projects in this file the first 2 are the file handling and 2darray one but I have updated them with structures and the data of sturctures is stored in 2 seprate file for 2 seprate projects.
* The 3rd project is the last one I which I made a link between the first 2 projects.and made an entity that requires data from previous 2 files and add some new data and stores into a new file.
* I also have implemented modular programing such that each code is divided into 3 parts:

1. Main file.
2. Header file
3. Implementation of header file.

The entity is tours and the new 4 attributes are:

**First code:**

* Name
* Destination
* By road/air
* Days at destination

**Second code:**

1. Name

2. Country visited

3. Destination airport name

4. Favourate landmark

**Third code:**

* Name
* Detination
* Arrival day
* time of arrival

**The first code consists of following menus**:

1. Add tour data

2. View tours data

3. search/find record

4. Delete a record

5. Edit arecord

6. Exit out of program

Code first project:

Main:

#include <iostream>

#include "defination\_of\_h1.cpp"//user defined header file included

using namespace std;

int main() {

    // Allocate memory for the array of tours

    Tour\* tours;//poiter made for Tour datatype

    int tourCount = 0;

    tours = new Tour[MAX\_TOURS];//dynamic memory allocation

    string filename = "tour\_data.txt";//file name to store data of structure

    loadFromFile(tours, tourCount, filename);//to load previous data from file when program starts

    while (true) {

        menu(tours, tourCount,filename);//menu function call

        cout << "=============================================";

    }

    // Deallocate memory for the array of tours

    delete[] tours;

    return 0;

}

Header file:

#pragma once//header guard helps to prevent header files from being included multiple times.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int MAX\_TOURS = 10; // Maximum number of tours

struct Tour {//struct made for Tours

    string name;

    string location;

    string mode;

    int days;

};

//function prototype decleration

void saveToFile(Tour\* tours, int& tourCount, const string filename);

void loadFromFile(Tour\* tours, int& tourCount, const string filename);

void add\_tour(Tour\* tours, int& tourCount, const string filename);

void view\_tours(Tour\* tours, int& tourCount);

void find\_Tour(Tour\* tours, int& tourCount);

void delete\_tour(Tour\* tours, int& tourCount, const string filename);

void replace\_tour(Tour\* tours, int& tourCount, const string filename);

void menu(Tour\* tours, int& tourCount, const string filename);

Implementation file:

#include "header1.h"

void saveToFile(Tour\* tours, int& tourCount, const string filename) {// Function to save tour data to a file

    ofstream fout;

    fout.open(filename);

    if (fout.fail()) {

        cout << "Error opening file for writing." << endl;

        return;

    }

    for (int i = 0; i <= tourCount; ++i) {//saves data to file

        fout << tours[i].name << " " << tours[i].location << " " << tours[i].mode << " " << tours[i].days << endl;

    }

    fout.close();

}

// Function to load tour data from a file

void loadFromFile(Tour\* tours, int& tourCount, const string filename) {

    ifstream fin;

    fin.open(filename);

    if (fin.fail()) {

        cout << "Error opening file for reading. Starting with an empty data set." << endl;

        return;

    }

    tourCount = 0; // Reset tour count to pick data from the start of the file

    while (tourCount < MAX\_TOURS && fin >> tours[tourCount].name >> tours[tourCount].location >> tours[tourCount].mode >> tours[tourCount].days) {

        tourCount++; // Loading data into the struct of tour

    }

    fin.close(); // File closing

}

void add\_tour(Tour\* tours, int& tourCount, const string filename) {//function to add data

    if (tourCount < MAX\_TOURS) {

        // Taking input from the user

        cout << "\nBy which name would you like to book this trip: ";

        cin.ignore();//clear anything in buffer

        getline(cin, tours[tourCount].name);

        cout << "\nWhat is your Destination: ";

        getline(cin, tours[tourCount].location);

        cout << "\nMode of transport (air/road): ";

        getline(cin, tours[tourCount].mode);

        cout << "\nHow many days would you like to spend in the destination (in days): ";

        getline(cin,tours[tourCount].days);

        cout << "Tour added successfully.\n";

        // Now saving data to file we just call the function

        saveToFile(tours, tourCount, filename);

        tourCount++;

    } else {

        cout << "Maximum number of tours reached.\n";

    }

}

void view\_tours(Tour\* tours, int& tourCount) {//this function views tour data

    cout << "NAMES\t\tDESTINATION\t\tMODE\t\tDAYS\n" << endl;

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

        cout << tours[i].name << "\t\t" << tours[i].location << "\t\t" << tours[i].mode << "\t\t" << tours[i].days << endl;

    }

}

void find\_Tour(Tour\* tours, int& tourCount) {//this function finds tour data

    cout << "Enter name to find\n";

    string name;

    cin.ignore(); // Ignore any characters in the buffer

    getline(cin, name);

    bool found = false;

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

        if (tours[i].name == name) {

            cout << "Match found:\n";

            cout << tours[i].name << "\t" << tours[i].location << "\t" << tours[i].mode << "\t" << tours[i].days << endl;

            found = true;

            break;

        }

    }

    if (!found) {//if match not found

        cout << "Match not found\n";

    }

}

void delete\_tour(Tour\* tours, int& tourCount, const string filename) {//this function delet tour data

    cout << "Enter Name to Delete:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;//random value assing to index

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

        if (tours[i].name == name) {

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

        cout << "Record deleted\n";

        if (foundIndex < tourCount - 1) {//check if the foundindex is not the last index in the array. If it is the last index, there's no need to perform the replacement,just decremnt it

            tours[foundIndex] = tours[tourCount - 1];//assign teh last index data to the foundindex and the decrement the array size.

        }

        tourCount--;//decremantation

        // Save data to file after deleting a tour

        saveToFile(tours, tourCount, filename);

    } else {

        cout << "Match not found\n";

    }

}

void replace\_tour(Tour\* tours, int& tourCount, const string filename) {//this function repalce tour data

    cout << "Enter Name to Replace:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;

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

        if (tours[i].name == name) {//cheking wether the name exist in file

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

         // Taking new input to replace the old data

        cout << "Enter new Name\n";

        getline(cin, tours[foundIndex].name);

        cout << "Enter location\n";

        getline(cin, tours[foundIndex].location);

        cout << "By Road or Air\n";

        getline(cin, tours[foundIndex].mode);

        cout << "Enter Days\n";

        getline(cin,tours[foundIndex].days);

        cout << "Record replaced successfully\n";

        // Save data to file after replacing a tour

        saveToFile(tours, tourCount, filename);

    } else {

        cout << "Match not found\n";

    }

}

void menu(Tour\* tours, int& tourCount, const string filename) {//this function is menu for tour data

    cout << endl

         << endl;

    cout << "\t\tTour Management System\t\t\n";

    cout << " 1. Add Tour Data.\n 2. View Tour Data. \n 3. Find Tour. \n 4. Delete Tour Data.\n 5. Replace Tour Data\n 6. Return back\n";

    int option;

    cin >> option;

    switch (option) {

    case 1:

        add\_tour(tours, tourCount,filename);

        break;

    case 2:

        view\_tours(tours, tourCount);

        break;

    case 3:

        find\_Tour(tours, tourCount);

        break;

    case 4:

        delete\_tour(tours, tourCount,filename);

        break;

    case 5:

        replace\_tour(tours, tourCount,filename);

        break;

    case 6:

        exit(0);

        break;

    default:

        cout << "Invalid entry\n";

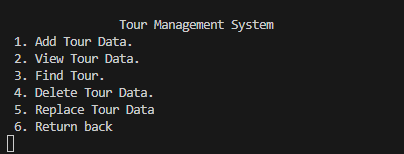
        break;

    }

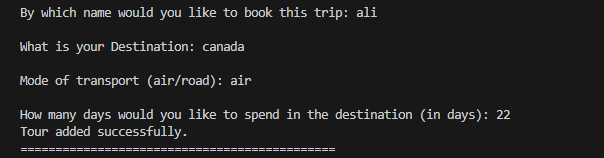
}

Code Output:

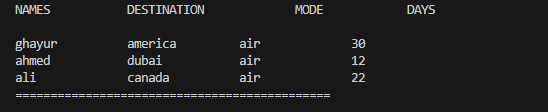
Menu:



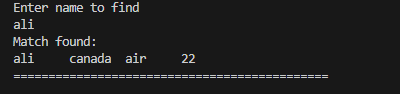
Option 1:



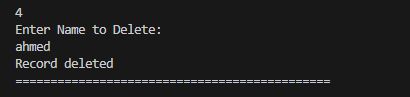
Option 2:



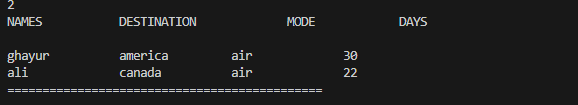
Option 3:



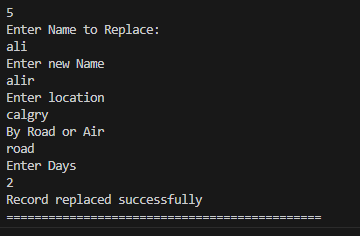
Option 4:



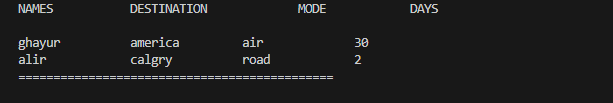
Name data deleted:



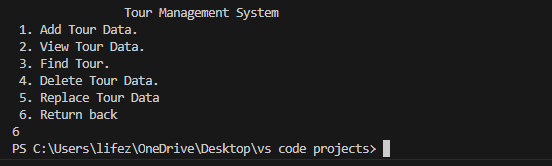
Option 5:



Name data replaced:



Option 6: (program exits)



**The second code consists of following menus:**

1. Add tour data

2. View tours data

3. search/find record

4. Delete a record

5. Edit arecord

6. Exit out of program

Code second project:

Main:

#include <iostream>

#include "defination\_of\_h2.cpp"//user defined header file included

using namespace std;

int main() {

    // Allocate memory for the array of tours

    Tour\* tours;//poiter made for Tour datatype

    int tourCount = 0;

    tours = new Tour[MAX\_TOURS];//dynamic memory allocation

    string filename = "tour\_data2.txt";//file name to store data of structure

    // Load tour data from the file at the beginning

    loadFromFile(tours, tourCount, filename);

    while (true) {

        menu(tours, tourCount,filename);//menu function call

        cout << "=============================================";

    }

    // Deallocate memory for the array of tours

    delete[] tours;

    return 0;

}

Header file:

#pragma once//header guard helps to prevent header files from being included multiple times.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

const int MAX\_TOURS = 10; // Maximum number of tours

struct Tour {//struct made for Tours

    string name;

    string destination;

    string airport;

    string landmark;

};

//function prototype decleration

void saveToFile(Tour\* tours, int& tourCount, const string filename);

void loadFromFile(Tour\* tours, int& tourCount, const string filename);

void add\_tour(Tour\* tours, int& tourCount, const string filename);

void view\_tours(Tour\* tours, int& tourCount);

void find\_Tour(Tour\* tours, int& tourCount);

void delete\_tour(Tour\* tours, int& tourCount, const string filename);

void replace\_tour(Tour\* tours, int& tourCount, const string filename);

void menu(Tour\* tours, int& tourCount, const string filename);

Implementation file:

#include "header2.h"

void saveToFile(Tour\* tours, int& tourCount, const string filename) {// Function to save tour data to a file

    ofstream fout;

    fout.open(filename);

    if (fout.fail()) {

        cout << "Error opening file for writing." << endl;

        return;

    }

    for (int i = 0; i <= tourCount; ++i) {//saves data to file

        fout << tours[i].name << " " << tours[i].destination << " " << tours[i].airport << " " << tours[i].landmark << endl;

    }

    fout.close();

}

// Function to load tour data from a file

void loadFromFile(Tour\* tours, int& tourCount, const string filename) {

    ifstream fin;

    fin.open(filename);

    if (fin.fail()) {

        cout << "Error opening file for reading. Starting with an empty data set." << endl;

        return;

    }

    tourCount = 0; // Reset tour count to pick data from the start of the file

    while (tourCount < MAX\_TOURS && fin >> tours[tourCount].name >> tours[tourCount].destination >> tours[tourCount].airport >> tours[tourCount].landmark) {

        tourCount++; // Loading data into the struct of tour

    }

    fin.close(); // File closing

}

void add\_tour(Tour\* tours, int& tourCount, const string filename) {//function to add data

    if (tourCount < MAX\_TOURS) {

        // Taking input from the user

        cout << "\nBy which name would you like to book this trip: ";

        cin.ignore();//clear anything in buffer

        getline(cin, tours[tourCount].name);

        cout << "\nWhat is your Destination: ";

        getline(cin, tours[tourCount].destination);

        cout << "\nDestination airport Name: ";

        getline(cin, tours[tourCount].airport);

        cout << "\nFavourate Landmark: ";

        getline(cin,tours[tourCount].landmark);

        cout << "Tour added successfully.\n";

        // Now saving data to file we just call the function

        saveToFile(tours, tourCount, filename);

        tourCount++;

    } else {

        cout << "Maximum number of tours reached.\n";

    }

}

void view\_tours(Tour\* tours, int& tourCount) {//this function views tour data

    cout << "NAMES\t\tDESTINATION\t\tAIRPORT\t\tLANDMARK\n" << endl;

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

        cout << tours[i].name << "\t\t" << tours[i].destination << "\t\t" << tours[i].airport << "\t\t" << tours[i].landmark << endl;

    }

}

void find\_Tour(Tour\* tours, int& tourCount) {//this function finds tour data

    cout << "Enter name to find\n";

    string name;

    cin.ignore(); // Ignore any characters in the buffer

    getline(cin, name);

    bool found = false;

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

        if (tours[i].name == name) {

            cout << "Match found:\n";

            cout << tours[i].name << "\t" << tours[i].destination << "\t" << tours[i].airport << "\t" << tours[i].landmark << endl;

            found = true;

            break;

        }

    }

    if (!found) {//if match not found

        cout << "Match not found\n";

    }

}

void delete\_tour(Tour\* tours, int& tourCount, const string filename) {//this function delet tour data

    cout << "Enter Name to Delete:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;//random value assing to index

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

        if (tours[i].name == name) {

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

        cout << "Record deleted\n";

        if (foundIndex < tourCount - 1) {//check if the foundindex is not the last index in the array. If it is the last index, there's no need to perform the replacement,just decremnt it

            tours[foundIndex] = tours[tourCount - 1];//assign teh last index data to the foundindex and the decrement the array size.

        }

        tourCount--;//decremantation

        // Save data to file after deleting a tour

        saveToFile(tours, tourCount, filename);

    } else {

        cout << "Match not found\n";

    }

}

void replace\_tour(Tour\* tours, int& tourCount, const string filename) {//this function repalce tour data

    cout << "Enter Name to Replace:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;

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

        if (tours[i].name == name) {//cheking wether the name exist in file

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

         // Taking new input to replace the old data

        cout << "Enter new Name\n";

        getline(cin, tours[foundIndex].name);

        cout << "Enter Destination\n";

        getline(cin, tours[foundIndex].destination);

        cout << "Enter Airport\n";

        getline(cin, tours[foundIndex].airport);

        cout << "Enter Landmark\n";

        getline(cin,tours[foundIndex].landmark);

        cout << "Record replaced successfully\n";

        // Save data to file after replacing a tour

        saveToFile(tours, tourCount, filename);

    } else {

        cout << "Match not found\n";

    }

}

void menu(Tour\* tours, int& tourCount, const string filename) {//this function is menu for tour data

    cout << endl

         << endl;

    cout << "\t\tTour Management System\t\t\n";

    cout << " 1. Add Tour Data.\n 2. View Tour Data. \n 3. Find Tour. \n 4. Delete Tour Data.\n 5. Replace Tour Data\n 6. Return back\n";

    int option;

    cin >> option;

    switch (option) {

    case 1:

        add\_tour(tours, tourCount,filename);

        break;

    case 2:

        view\_tours(tours, tourCount);

        break;

    case 3:

        find\_Tour(tours, tourCount);

        break;

    case 4:

        delete\_tour(tours, tourCount,filename);

        break;

    case 5:

        replace\_tour(tours, tourCount,filename);

        break;

    case 6:

        exit(0);

        break;

    default:

        cout << "Invalid entry\n";

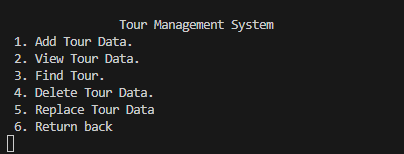
        break;

    }

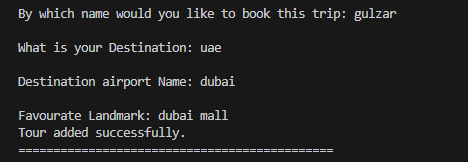
}

Code output:

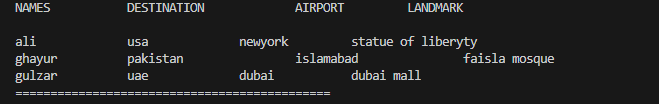
Menu:



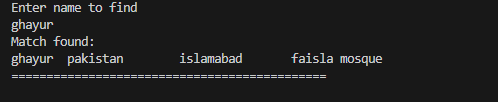
Option 1:



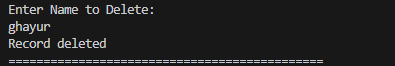
Option 2:



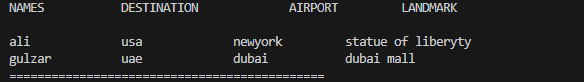
Option 3:



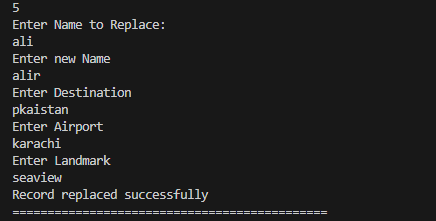
Option 4:



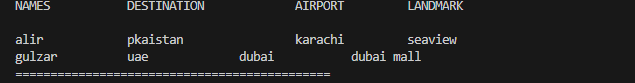
Name data deleted:



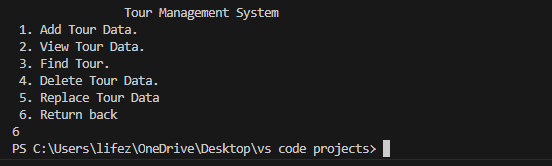
Option 5:



Name data replaced:



Option 6: (program exits)



**The Third code consists of following menus**:

1. Add tour data

2. View tours data

3. search/find record

4. Delete a record

5. Edit arecord

6. Exit out of program

Code third project:

Main:

#include "defination\_of\_3.cpp"//user defined header file included

#include <iostream>

using namespace std;

int main() {

    Tour\* tours; // Pointer to store tour data

    string filename = "tour\_data3.txt";

    int tourCount=0;

    tours = new Tour[MAX\_TOURS];//dynamic memory allocation

    // Load tour data from the file at the beginning

    loadFromFile(tours, tourCount, filename);//file name to store data of structure

    while (true) {

        menu(tours, tourCount, filename);//menu function call

        cout << "=============================================";

    }

    // Deallocate memory for the array of tours

    delete[] tours;

    return 0;

}

Header file:

#pragma once//header guard helps to prevent header files from being included multiple times.

#include<iostream>

#include<fstream>

#include<string>

using namespace std;

const int MAX\_TOURS = 20; // Maximum number of tours

struct Tour {//struct made for Tours

    string name;

    string destination;

    string day;

    string time;

};

// Function prototypes

void menu(Tour\* tours, int& tourCount, string filename);

void loadFromFile(Tour\* tours, int& tourCount, string filename);

void file\_merge(Tour\* tours, int& tourCount, ifstream& nameFile, ifstream& destinationFile);

void saveToFile(Tour\* tours, int& tourCount, string filename);

void add\_tour(Tour\* tours, int& tourCount, string filename);

void view\_tours(Tour\* tours, int& tourCount);

void find\_tour(Tour\* tours, int& tourCount);

void replace\_tour(Tour\* tours, int& tourCount, string filename);

void delete\_tour(Tour\* tours, int& tourCount, string filename);

Implementation file:

#include "header3.h"

void menu(Tour\* tours, int& tourCount, string filename) {//this function is menu for tour data

    cout << endl << endl;

    cout << "\t\tTour Management System\t\t\n";

    cout << " 1. Add Tour Data.\n 2. View Tour Data. \n 3. Find Tour. \n 4. Delete Tour Data.\n 5. Replace Tour Data\n 6. Return back\n";

    int option;

    cin >> option;

    switch (option) {

        case 1:

            add\_tour(tours, tourCount, filename);

            break;

        case 2:

            view\_tours(tours, tourCount);

            break;

        case 3:

            find\_tour(tours, tourCount);

            break;

        case 4:

            delete\_tour(tours, tourCount, filename);

            break;

        case 5:

            replace\_tour(tours, tourCount, filename);

            break;

        case 6:

            exit(0);

            break;

        default:

            cout << "Invalid entry\n";

            break;

    }

}

void loadFromFile(Tour\* tours, int& tourCount, string filename) {//load data from file to struct

    ifstream fin;

    ifstream nameFile("tour\_data.txt"); // File of project 1

    ifstream destinationFile("tours\_data2.txt"); // File of project 2

    if (!nameFile.is\_open() || !destinationFile.is\_open()) {

        cout << "Error opening files\n";

        return; // Error handling

    }

    file\_merge(tours,tourCount,nameFile, destinationFile); // Function call

    fin.open(filename);

    if (fin.fail()) {

        cout << "Error opening file for reading. Starting with an empty data set." << endl;

        return; // Error handling

    }

    tourCount = 0; // Reset tour count to pick data from the start of the file

    while (fin >> tours[tourCount].name >> tours[tourCount].destination >> tours[tourCount].day >> tours[tourCount].time) {

        tourCount++; // Loading data into the struct of tour

    }

    fin.close(); // File closing

}

void saveToFile(Tour\* tours, int& tourCount, string filename) {// Function to save tour data to a file

    ofstream fout;

    fout.open(filename); // Opening new file

    if (fout.fail()) {

        cout << "Error opening file for writing." << endl; // Error handling

        return;

    }

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

        fout << tours[i].name << " " << tours[i].destination << " " << tours[i].day << " " << tours[i].time << endl; // Writing in file

    }

    fout.close(); // Good practice to close file

}

void file\_merge(Tour \*tours, int &tourCount, ifstream &nameFile, ifstream &destinationFile)// Function to load tour data from a past 2 project

{

    tourCount = 0; // Reset tour count to pick data from the start of the file

    struct Temp1 {//making temp stuff to hold data

        string name;

        string location;

        string mode;

        int days;

    };

    struct Temp2 {

        string name;

        string destination;

        string airport;

        string landmark;

    };

    Temp1 temp1[MAX\_TOURS]; // Variable declaration

    Temp2 temp2[MAX\_TOURS];

    // Reading data from both files

    while (tourCount < MAX\_TOURS &&

           destinationFile >> temp2[tourCount].name >> temp2[tourCount].destination >> temp2[tourCount].airport >> temp2[tourCount].landmark &&

           nameFile >> temp1[tourCount].name >> temp1[tourCount].location >> temp1[tourCount].mode >> temp1[tourCount].days) {

        //taking the data we need

        tours[tourCount].name = temp1[tourCount].name;

        tours[tourCount].destination = temp2[tourCount].destination;

        tourCount++; // Loading data into the struct of tour

    }

    // Close the files after reading

    nameFile.close();

    destinationFile.close();

}

void add\_tour(Tour\* tours, int& tourCount, string filename) {

    if (tourCount < MAX\_TOURS) {

        // Taking input from the user

        cout << "\nFor name: " << tours[tourCount].name << " and Destination: " << tours[tourCount].destination << " enter day of arrival and time\n";

        //the uper line picks name from project 1 file and destination from project 2 file

        cout << "Arrived at what Day\n";

        cin.ignore();

        getline(cin, tours[tourCount].day);

        cout << "Arrived at what time (format hh:mm:ss)\n";

        getline(cin, tours[tourCount].time);

        cout << "Updated successfully.\n";

        saveToFile(tours, tourCount,filename);// Now saving data to file we just call the function

        tourCount++;

    } else {

        cout << "Maximum number of tours reached.\n";

    }

}

void view\_tours(Tour\* tours, int& tourCount) {//this function views tour data

    cout << "NAMES\t\tDESTINATION\t\tDAY\t\tTIME\n" << endl;

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

        cout << tours[i].name << "\t\t" << tours[i].destination << "\t\t" << tours[i].day << "\t\t" << tours[i].time << endl;

    }

}

void find\_tour(Tour\* tours, int& tourCount) {//this function finds tour data

    cout << "Enter name to find\n";

    string name;

    cin.ignore();// Ignore any characters in the buffer

    getline(cin, name);

    bool found = false;

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

        if (tours[i].name == name) {

            cout << "Match found:\n";

            cout << tours[i].name << "\t" << tours[i].destination << "\t" << tours[i].day << "\t" << tours[i].time << endl;

            found = true;

            break;

        }

    }

    if (!found) {//if match not found

        cout << "Match not found\n";

    }

}

void delete\_tour(Tour\* tours, int& tourCount, string filename) {//this function delet tour data

    cout << "Enter Name to Delete:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;//random value assing to index

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

        if (tours[i].name == name) {

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

        cout << "Record deleted\n";

        if (foundIndex < tourCount - 1) {//check if the foundindex is not the last index in the array. If it is the last index, there's no need to perform the replacement,just decremnt it

            tours[foundIndex] = tours[tourCount - 1];//assign teh last index data to the foundindex and the decrement the array size.

        }

        tourCount--;//decremantation

        saveToFile(tours, tourCount,filename); // Save data to file after deleting a tour

    } else {

        cout << "Match not found\n";

    }

}

void replace\_tour(Tour\* tours, int& tourCount, string filename) {//this function repalce tour data

    cout << "Enter Name to Replace:\n";

    string name;

    cin.ignore();

    getline(cin, name);

    int foundIndex = -1;

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

        if (tours[i].name == name) {//checking whether the name exists in the file

            foundIndex = i;

            break;

        }

    }

    if (foundIndex != -1) {

         // Taking new input to replace the old data

        cout << "Enter new Name\n";

        getline(cin, tours[foundIndex].name);

        cout << "Enter new Destination\n";

        getline(cin, tours[foundIndex].destination);

        cout << "Arrived at what Day\n";

        getline(cin, tours[foundIndex].day);

        cout << "Arrived at what time (format hh:mm:ss)\n";

        getline(cin, tours[foundIndex].time);

        cout << "Record replaced successfully\n";

        // Save data to file after replacing a tou

        saveToFile(tours, tourCount,filename); // Save data to file after replacing a tour

    } else {

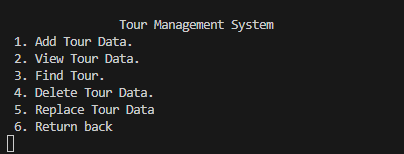
        cout << "Match not found\n";

    }

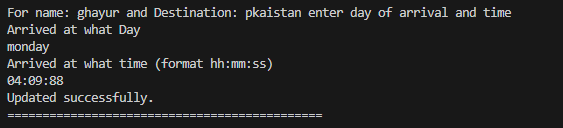
}

Code output:

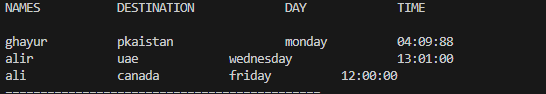
Menu:



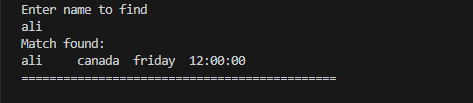
Option 1:



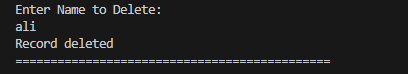
Option 2:



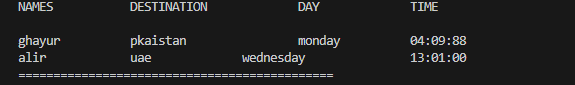
Option 3:



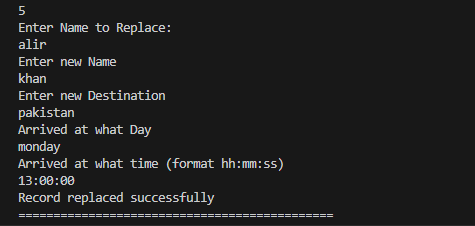
Option 4:



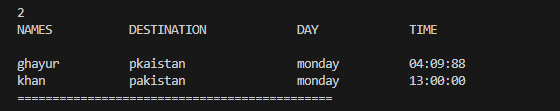
Name data deleted:



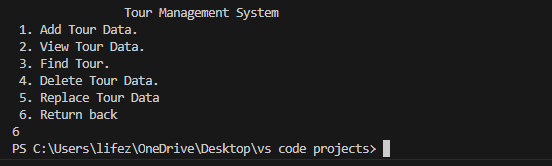
Option 5:



Name data replaced:



Option 6: (program exits)



⭐⭐⭐⭐⭐