**STRUCTURED AND OBJECT-ORIENTED PROGRAMMING MINI PROJECT USING C++ for PPS-3 Travel Planner**

Slot: C1

Lab: L31 + L32 + L45 + L46

Class Number (Theory): VL2023230501147

Class Number (Lab): VL2023230504707

Members:

Khanak Shah - 23BCB0016

Ashman Sodhi - 23BDS0068

Rupin Gupta - 23BDS0094

**About the program:**

In this C++ program a range of programming techniques are used to enhance its functionality and maintainability:

***1)Operator Overloading:***

By overloading the += operator in the Planner class, users can add locations to their itinerary with ease. This simplifies the code and provides a more intuitive interface for managing trip details.

***2)Function Overloading:***

The Date class showcases function overloading, where the showdata function is overloaded to accept different parameter types. This flexibility allows users to display date information in various formats, enhancing the program's versatility.

***3)Function Overriding:***

With function overriding, each derived class (City, Hotel, Activity) provides its own implementation of the display function inherited from the base class (Location). This ensures that location-specific details are properly displayed in the itinerary.

***4)Friend Function:***

A friend function is used in the Date class to access the private members for the noofdays function. This enables calculation of the trip duration by accessing date information securely.

***5)Inheritance:***

Through inheritance, common attributes are inherited from the base class Location, therefore reducing redundancy. Derived classes then build upon this foundation, adding specialized features for cities, hotels, and activities, thus promoting code reusability and enhancing modularity.

By using these object oriented programming concepts, the code achieves a high degree of flexibility, readability, and extensibility. This not only enhances the user experience but also facilitates easier maintenance and future enhancements to the travel planning application.

***6)Abstract class:***

In this C++ program, incorporating an abstract class, named Location, would provide a generalized blueprint for all types of locations, including cities, hotels, and activities. This abstract class would define common attributes and methods shared among different locations, enforcing a consistent interface across the itinerary. Derived classes, such as City, Hotel, and Activity, would inherit from Location and provide specialized implementations of its pure virtual functions, tailored to their respective types of locations. This abstraction promotes code reusability, encapsulation, and maintainability, enhancing the overall structure and flexibility of the travel planning application.

***7)Saving Itinerary information to a Text File:***

By implementing file I/O operations, the program can export travel information, allowing users to easily store, share, or manually adjust their travel plans. This approach ensures that important data is accessible and backed up for future use or sharing, providing added convenience and flexibility. Ultimately, this feature enhances the user experience and makes the application more robust.

**CODE**:

// Importing required libraries

#include<iostream>

#include<string>

#include<stdlib.h>

#include <fstream>

#include<string.h>

using namespace std;

static int total\_cost\_of\_trip = 0;

char iten\_name[100];

ofstream outputFile(iten\_name);

// Creating Abstract Class (Base Class)

class Location{

protected:

string city\_name;

string country\_name;

public:

Location(string city, string country):city\_name(city), country\_name(country){} virtual void display()=0;

virtual int getPrice(){return 0;}

};

//Creating City Class (inherited from Location)

class City:public Location{

public:

// Constructor

City():Location("", ""){

city\_name= "City";

}

// Getting City Information from User

void city\_info(){

cin.ignore();

cout<<"Enter City : ";

getline(cin , city\_name);

cout<<"Enter Country : ";

cin>>country\_name;

}

// Displaying City Information

void display(){

cout<<"City: "<<city\_name<<endl;

cout<<"Country: "<< country\_name<<endl;

}

// Saving City Information into Text File

void city\_save(char iten\_name[100]){

ofstream outputFile(iten\_name,ios::app);

outputFile<<"City: "<<city\_name<<endl;

outputFile<<"Country: "<< country\_name<<endl;

}

};

// Creating Hotel Class (inherited from Location)

class Hotel:public Location{

protected:

string hotel\_name;

int hotel\_price\_per\_day;

int stay\_duration;

public:

// Defining Constructor

Hotel():Location("", ""){

hotel\_name="Default Hotel";

hotel\_price\_per\_day=0;

stay\_duration=1;

}

// Getting Hotel Information from User

void hotel\_info(){

cin.ignore();

cout<<"Enter Hotel Name : ";

getline(cin, hotel\_name);

cout<<"Enter Hotel Price per Day : ";

cin>>hotel\_price\_per\_day;

cout<<"Enter Number of Days of Stay : ";

cin>>stay\_duration;

}

// Displaying Hotel Information

void showdetails(){

cout<<"Hotel : "<<hotel\_name<<endl;

cout<<"Stay Duration: "<< stay\_duration<<" days"<< endl;

}

void display()override{

cout<<"City: "<<city\_name<<endl;

cout<<"Country: "<< country\_name<<endl;

}

// Returning Total Hotel Expense

int getPrice()override{

return hotel\_price\_per\_day\*stay\_duration;

}

// Saving Hotel Information to Text File

void save\_hotel(char iten\_name[100]){

ofstream outputFile(iten\_name,ios::app);

outputFile<<"Hotel : "<<hotel\_name<<endl;

outputFile<<"Stay Duration: "<< stay\_duration<<" days"<< endl;

}

};

// Creating Activity Class (Location Inherited)

class Activity:public Location{

protected:

string act\_name;

int activity\_price;

public:

// Defining Constructor

Activity():Location("", ""){

act\_name="Default Activity";

activity\_price = 0;

}

// Getting Activity Information from User

void act\_info(){

cin.ignore();

cout<<"Enter Activity Name : ";

getline(cin, act\_name);

cout<<"Enter Activity Price : ";

cin>>activity\_price;

}

// Showing Activity Information

void showdata(){

cout<<"Activity : " <<act\_name<<endl;

//display();

//cout<<"Activity Price per Day:"<<activity\_price<<endl;

}

void display()override{

cout<<"City:"<<city\_name<<endl;

cout<< "Country: "<<country\_name<<endl;

}

// Returning Activity Price

int getPrice()override{

return activity\_price;

}

// Saving Activity Information to Text File

void save\_act(char iten\_name[100]){

ofstream outputFile(iten\_name,ios::app);

outputFile<<"Activity : " <<act\_name<<endl;

}

};

// Creating Date Class

class Date {

protected:

int day;

int month;

int year;

public:

// Getting Date Information from User

void fdate(){

cout<<"Date : "<<endl;

cout<<"Day: ";

cin>>day;

cout<<"Month: ";

cin>>month;

cout<<"Year: ";

cin>>year;

}

// Showing Date Information

void showdata(Date const d1, Date const d2){

cout<<"From: "<<d1.day<<"/"<<d1.month<<"/"<<d1.year;

cout<<" To: "<<d2.day<<"/"<<d2.month<<"/"<<d2.year<<endl;

}

// Saving Date information to Text File

void savedata(Date const d1, Date const d2 ,char iten\_name[100])

{

ofstream outputFile(iten\_name , ios::app);

outputFile<<"From: "<<d1.day<<"/"<<d1.month<<"/"<<d1.year;

outputFile<<" To: "<<d2.day<<"/"<<d2.month<<"/"<<d2.year<<endl;

}

// Adding friend function to count number of Days

friend int noofdays(Date const d1, Date const d2);

};

// Defining Function to count number of Days

int noofdays(Date const d1, Date const d2){

int x = 0;

if (d2.year>d1.year){

x+=(d2.year-d1.year)\*365;

}

if (d2.month>d1.month){

x+=(d2.month-d1.month)\*30;

}

if (d1.month>d2.month){

x-=(d1.month-d2.month)\*30;

}

if (d2.day>d1.day){

x+=(d2.day-d1.day);

}

if (d1.day>d2.day){

x-=(d1.day-d2.day);

}

if (x<0){

cout<<"Invalid";

return 0;

}

return x;

}

// Creating Planner class to add up all the Information

class Planner{

public:

Location\* itinerary[1000];

int current=0;

int total=0;

// Operator Overloading to add Location

Planner& operator+=(Location\* location){

if (current<10){

itinerary[current++]=location;

}else{

cout<<"Cannot add more locations. Maximum limit reached."<< endl;

}

return \*this;

}

// Displaying Itinerary Information

void displayItinerary(){

cout<< "Trip Itinerary:"<<endl;

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

itinerary[i]->display();

}

}

// Getting Total cost of the trip

int getTotalCost(){

total\_cost\_of\_trip += total;

return total;

}

// Calculating Total Cost of Trip

void calculateTotalCost(){

total=0;

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

total+=itinerary[i]->getPrice();

}

}

};

// Getting Total Expense of the trip

void bookTrip(Planner& planner){

planner.calculateTotalCost();

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

cout<<"Total itinerary Expense : Rs."<< planner.getTotalCost()<< endl;

}

// Saving Total Expense to Text File

void save\_bookTrip(Planner& planner , char iten\_name[100]){

planner.calculateTotalCost();

ofstream outputFile(iten\_name,ios::app);

outputFile<<"----------------------------------------------"<<endl;

outputFile<<"Total Itinerary Expense : Rs."<< planner.getTotalCost()<< endl;

}

// Driver Code

int main(){

City c1[10];

Hotel h1[10];

Activity a1[100];

Date d1[10] , d2[10];

char iten\_name[100];

int n;

// Getting Itinerary Name

cout << "Enter Itinerary Name : ";

cin.getline(iten\_name, 100);

// Getting Number of Cities

cout << "Enter Number of Cities (Max 10): ";

cin >> n;

// Getting all the information from User

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

cout << "Location : " << i+1 << endl;

c1[i].city\_info();

h1[i].hotel\_info();

a1[i].act\_info();

d1[i].fdate();

d2[i].fdate();

cout << endl;

}

system ("CLS");

// Adding up all the information

Planner trip1;

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

trip1+=&h1[i];

trip1+=&a1[i];

}

// Displaying all the information

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

cout << " Your Itinerary " << endl;

cout << " "<<iten\_name<<" " << endl;

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

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

cout << " LOCATION : " << i+1 << " " << endl;

d1[i].showdata(d1[i],d2[i]);

c1[i].display();

h1[i].showdetails();

a1[i].showdata();

}

bookTrip(trip1);

cout << "Total Number of Days : " << noofdays(d1[0],d2[n-1]) + 1 << endl;

cout << endl;

// Saving information to text file

int sa\_ex;

cout << "Press 1 to save Itinerary in Text File , Else 0 to exit" << endl;

cin >> sa\_ex;

char txt[5] = ".txt";

char final\_txt[100];

strcat(iten\_name , txt);

strcpy(final\_txt , iten\_name);

if (sa\_ex == 1){

ofstream outputFile(final\_txt,ios::app); // Writing Data to output file

outputFile << "----------------------------------------------" << endl;

outputFile << " Your Itenary " << endl;

outputFile << " "<<iten\_name<<" " << endl;

outputFile << "----------------------------------------------" << endl;

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

outputFile << " LOCATION : " << i+1 << " " << endl;

d1[i].savedata(d1[i],d2[i],final\_txt);

c1[i].city\_save(final\_txt);

h1[i].save\_hotel(final\_txt);

a1[i].save\_act(final\_txt);

}

save\_bookTrip(trip1 , final\_txt);

outputFile << "Total Number of Days : " << noofdays(d1[0],d2[n-1]) + 1 << endl;

outputFile << "Thank you for using Travel Planner!" << endl;

outputFile << "----------------------------------------------" << endl;

outputFile.close();

cout << "Your Itenary was saved with name : " << final\_txt << endl;

cout << "Thank you for using Travel Planner!" << endl;

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

} else {

cout << "Thank you for using Travel Planner!" << endl;

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

}

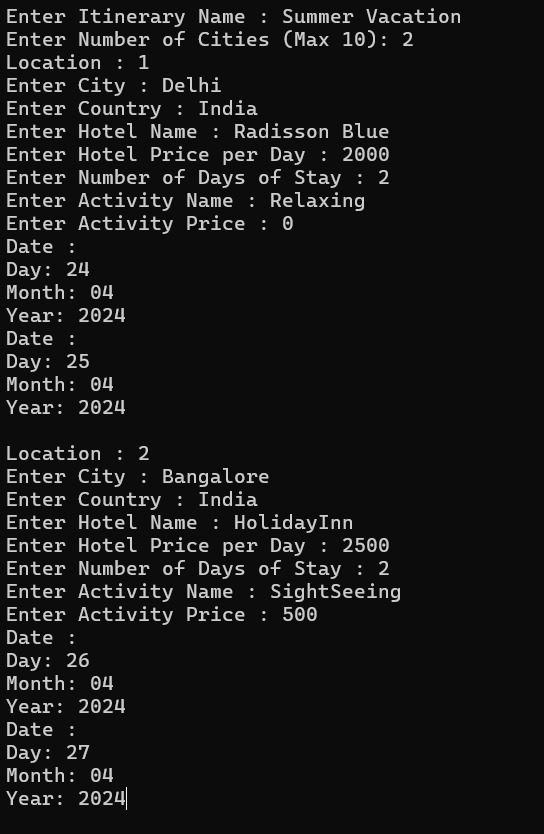
system("pause");

return 0;

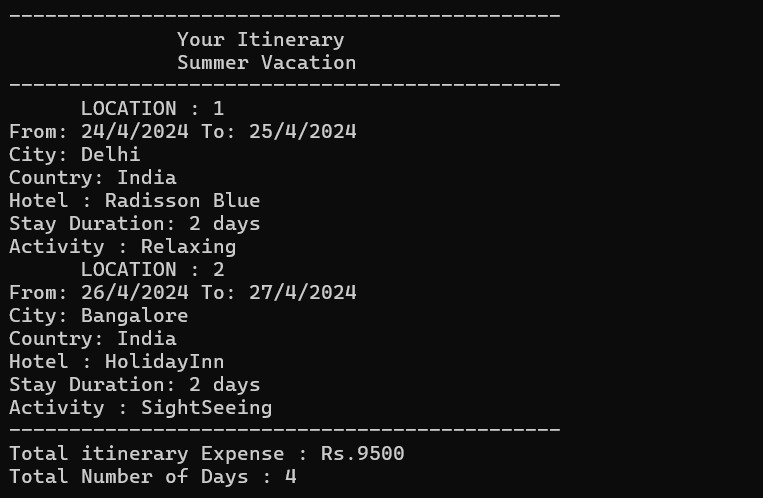
}

**OUTPUT**:

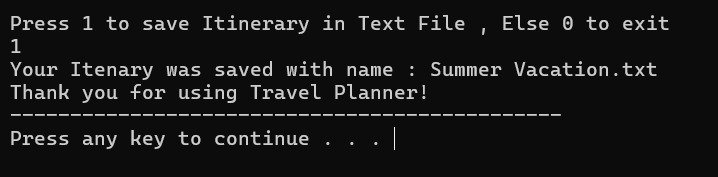
TAKING INPUT:



ITERNARY DISPLAY:



SAVING TO TEXT FILE:



DATA STORED IN TEXTFILE:

