#include <string>

#include "Date.h"

#include <iostream>

#pragma once

using namespace std;

class Trip

{

private:

int code;

string dst;

Date date;

public:

//constructor

Trip(int tCode = 1, string tDst = "Unknown", Date tDate = Date());

Trip(const Trip& other);

~Trip();

//get

int getCode();

string getDst();

Date getDate();

//set

void setCode(int newCode);

void setDst(string newDst);

void setDate(Date newDate);

//validate

bool validate(int tCode);

bool validate(string tDst);

//operator

const Trip& operator=(const Trip& oTrip);

//print

void PrintTrip();

};

#include "Trip.h"

#include <string>

#include <iostream>

using namespace std;

//constructor

Trip::Trip(int tCode, string destination, Date tDate) {

if (validate(tCode)) {

code = tCode;

}

else {

code = 0;

}

if (validate(destination)) {

dst = destination;

}

else {

dst = "Invalid";

}

date = tDate;

}

Trip::Trip(const Trip& other) {

\*this = other;

}

Trip::~Trip() {

}

//get

int Trip::getCode(){

return this->code;

}

string Trip::getDst() {

return this->dst;

}

Date Trip::getDate() {

return this->date;

}

//set

void Trip::setCode(int nCode) {

if (validate(nCode)) {

this->code = nCode;

}

else {

this->code = 0;

}

}

void Trip::setDst(string nDst) {

if (validate(nDst)) {

this->dst = nDst;

}

else {

this->dst = "Invalid";

}

}

void Trip::setDate(Date nDate) {

this->date = nDate;

}

//validate

bool Trip::validate(int number) {

return number > 0;

}

bool Trip::validate(string destination) {

for (int i = 0; destination[i] != '\0'; i++) {

if (isdigit(destination[i])) {

return false;

}

}

return true;

}

//operator

const Trip& Trip::operator=(const Trip& other) {

if (this != &other) {

code = other.code;

dst = other.dst;

date = other.date;

}

return \*this;

}

//print

void Trip::PrintTrip() {

cout << "Trip info:\n" << "Code: " << this->getCode() << "\nDestination: " << this->getDst() << "\nDate: " << endl;

this->getDate().PrintDate();

}

#include <iostream>

#include "Date.h"

#include "Trip.h"

using namespace std;

bool isEmpty(int size);

void PrintTrips(Trip\* arr, int size);

void EditTrips(Trip\* arr, int size);

Trip\* AddTrips(Trip\* arr, int size);

int main() {

int option, size = 0;

Trip\* arr = new Trip[0];

do {

cout << "\nPlease choose an option:\n1.Print All\n2.Edit Trip\n3.Add Trip\n4.Exit\n" << endl;

cin >> option;

switch (option) {

case 1:

PrintTrips(arr, size);

break;

case 2:

EditTrips(arr, size);

break;

case 3:

arr = AddTrips(arr, size);

size++;

break;

case 4:

cout << "\nBye!\n";

break;

default:

cout << "\nInvalid Option!\n";

break;

}

} while (option != 4);

return 0;

}

bool isEmpty(int size) {

return size == 0;

}

void PrintTrips(Trip\* arr, int size) {

if (isEmpty(size)) {

cout << "\nNo trips to print!\n";

return;

}

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

cout << "\nTrip #" << i+1 << ":\n";

arr[i].PrintTrip();

}

cout << "\nEnd of Trip list.\n";

}

void EditTrips(Trip\* arr, int size) {

if (isEmpty(size)) { //if no trips

cout << "\nNo trips in array to edit.\n";

return;

}

int tCode, tIndex = -1; //index check

Trip t = Trip();

cout << "\nPlease enter the code of the trip you'd like to edit.\n";

cin >> tCode;

for (int i = 0; i < size; i++) {//look through array

if (arr[i].getCode() == tCode) {

tIndex = i;

t = arr[i];

break;

}

}

if (tIndex == -1) { // didn't find code

cout << "\nCould not find a trip with this code!\n";

return;

}

//if we did find one

string answer, newDestination;

cout << "\nWould you like to update the destination? Y/N\n";

cin >> answer;

if (answer.compare("Y") == 0) {

cout << "\nPlease enter a new destination.\n";

cin >> newDestination;

t.setDst(newDestination);

}

cout << "\nWould you like to update the date? Y/N\n";

cin >> answer;

if (answer.compare("Y") == 0) {

int tDay, tMonth, tYear;

cout << "\nPlease enter a day.\n";

cin >> tDay;

cout << "\nPleae enter a month.\n";

cin >> tMonth;

cout << "\nPlease enter a year.\n";

cin >> tYear;

Date d = Date(tDay, tMonth, tYear);

t.setDate(d);

}

arr[tIndex] = t;

}

Trip\* AddTrips(Trip\* arr, int size) {

int tDay, tMonth, tYear, tCode;

string tDestination;

cout << "\nPlease enter the code of the trip you'd like to add.\n";

cin >> tCode;

cout << "\nPlease enter a destination.\n";

cin >> tDestination;

cout << "\nPlease enter a day.\n";

cin >> tDay;

cout << "\nPleae enter a month.\n";

cin >> tMonth;

cout << "\nPlease enter a year.\n";

cin >> tYear;

Date d = Date(tDay, tMonth, tYear);

Trip t = Trip(tCode, tDestination, d);

Trip\* newArray = new Trip[size + 1];

if (isEmpty(size)) {

newArray[0] = t;

}

else {

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

newArray[i] = arr[i];

}

newArray[size] = t;

}

return newArray;

}

#pragma once

class Date

{

private:

int day; //1-30

int month; // 1-12

int year; // 0+

public:

//constructor

Date(int nDay = 1, int nMonth = 1, int nYear = 2000);

//get

int getDay(); //this

int getMonth();

int getYear();

//set

void setDay(int newDay);

void setMonth(int newMonth);

void setYear(int newYear);

//validate

bool validate(int number, int type); //type is day/month/year ; 1 = day, 2 = month; 3 = year

void PrintDate(); // print in DD/MM/YY

//validate(25,1) = true;

//validate(25,2) = false;

//validate(25,3) = true;

};

#include "Date.h"

#include <iostream>

using namespace std;

Date::Date(int nDay, int nMonth, int nYear) {

if (validate(nDay, 1)) {

day = nDay;

}

else {

day = 0;

}

if (validate(nMonth, 2)) {

month = nMonth;

}

else {

month = 0;

}

if (validate(nYear,3)) {

year = nYear;

}

else {

year = 0;

}

}

//get

int Date::getDay() {

return this->day;

}

int Date::getMonth() {

return this->month;

}

int Date::getYear() {

return this->year;

}

//set

void Date::setDay(int newDay) { //must validate

if (validate(newDay, 1)) {

this->day = newDay;

}

else {

this->day = 0;

}

}

void Date::setMonth(int newMonth) {

if (validate(newMonth, 2)) {

this->month = newMonth;

}

else {

this->month = 0;

}

}

void Date::setYear(int newYear) {

if (validate(newYear, 3)) {

this->year = newYear;

}

else {

this->year = 0;

}

}

//validate

bool Date::validate(int number, int type) {

switch (type) {

case 1: //day

return number <= 30 && number >= 1;

case 2: //month

return number <= 12 && number >= 1;

case 3:

return number > 0;

default:

return false;

}

}

//print

void Date::PrintDate() {

cout << this->getDay() << "/" << this->getMonth() << "/" << this->getYear() << endl;

}a