Министерство науки и высшего образования РФ

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа «Компьютерных технологий и информационных систем»

ОТЧЕТ

по дисциплине «Практикум по программированию»

**Лабораторная работа № 2**

**Выполнил:**

Cтудент гр. 5130902/30003 Блинков П.В.

**Проверил**

Ст. преподаватель Журавская А.М.

Санкт-Петербург

2024 г.

## **Задание**

Вариант 4

Часть 1

Предметная область: **Аэропорт.**

Касса аэропорта имеет список тарифов на различные направления. При покупке

билета регистрируются паспортные данные.

Система должна:

* позволять вводить данные о тарифах;
* позволять вводить паспортные данные пассажира и регистрировать покупку билета;
* рассчитывать стоимость купленных пассажиром билетов;
* рассчитывать стоимость всех проданных билетов.

**Код программы**

#include <string>

#include <iostream>

#include <vector>

using namespace std;

bool IsNumber(const string& input)

{

for (char c : input) {

if (!isdigit(c)) {

cout << "Incorrect input. It's not a number. \n";

return false;

}

}

return true;

}

bool IsValidNumber(const string& input, int length) {

if (input.length() != length) {

cout << "Error: number must contain " << length << " digits.\n";

return false;

}

for (char c : input) {

if (!isdigit(c)) {

cout << "Error: the input contains non-digits symbols.\n";

return false;

}

}

return true;

}

bool IsValidNumber2(const string& input) {

if (input.length() > 9) {

cout << "Error. Your number is too much.\n";

return false;

}

return true;

}

bool IsTime(const string& input) {

if (input.length() != 5 || input[2] != ':') {

cout << "Error: Time must be in the format 'hh:mm'.\n";

return false;

}

string hours = input.substr(0, 2);

string minutes = input.substr(3, 2);

if (!IsNumber(hours) || !IsNumber(minutes)) {

cout << "Error: Hours and minutes must be numbers.\n";

return false;

}

int hh = stoi(hours);

int mm = stoi(minutes);

if (hh < 0 || hh >= 24 || mm < 0 || mm >= 60) {

cout << "Error: Invalid time value.\n";

return false;

}

return true;

}

bool IsDate(const string& input) {

if (input.length() != 10 || input[2] != '-' || input[5] != '-') {

cout << "Error: Date must be in the format 'dd-mm-yyyy'.\n";

return false;

}

string dayStr = input.substr(0, 2);

string monthStr = input.substr(3, 2);

string yearStr = input.substr(6, 4);

if (!IsNumber(dayStr) || !IsNumber(monthStr) || !IsNumber(yearStr)) {

cout << "Error: Day, month, and year must be numbers.\n";

return false;

}

int day = stoi(dayStr);

int month = stoi(monthStr);

int year = stoi(yearStr);

if (day < 1 || day > 31 || month < 1 || month > 12 || year < 1900 || year > 2006) {

cout << "Error: Invalid date value.\n";

return false;

}

vector<int> daysInMonth = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };

if (month == 2 && ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))) {

daysInMonth[1] = 29;

}

if (day > daysInMonth[month - 1]) {

cout << "Error: Invalid day for the given month.\n";

return false;

}

return true;

}

class Tariff {

private:

string place;

int cost;

string timeofFlight;

public:

Tariff(string plc, int cst, string toF) : place(plc), cost(cst), timeofFlight(toF) {}

string getplace() const {

return place;

}

int getcost() const {

return cost;

}

string gettimeofFlight() const {

return timeofFlight;

}

};

class PassengerInfo

{

private:

string name;

string surname;

string datebirth;

int series;

int number;

vector<Tariff> tickets;

public:

PassengerInfo(string nm, string srnm, string db, int srs, int nmbr) : name(nm), surname(srnm), datebirth(db), series(srs), number(nmbr) {};

void buyticket(Tariff ticket)

{

tickets.push\_back(ticket);

}

string getname() {

return name;

}

string getsurname() {

return surname;

}

string getdatebirth() {

return datebirth;

}

int getseries() {

return series;

}

int getnumber() {

return number;

}

int boughttickets() {

int summary = 0;

for (auto& ticket : tickets) {

summary += ticket.getcost();

}

return summary;

}

~PassengerInfo() { tickets.clear(); }

};

class airport

{

private:

static airport\* Airportptr;

vector<Tariff> Tariffs;

vector<PassengerInfo> Passengers;

int sumSales = 0;

airport() {};

public:

airport(const airport\* obj) = delete;

static airport\* getairportptr()

{

if (Airportptr == nullptr) {

Airportptr = new airport;

}

return Airportptr;

}

vector<Tariff> addTariff(string place, int cost, string timeofFlight) {

Tariffs.push\_back(Tariff(place, cost, timeofFlight));

return Tariffs;

}

const vector<Tariff>& getTariffs() const {

return Tariffs;

}

PassengerInfo\* CheckExistanceOfPassenger(int srs, int nmbr) {

for (auto& passenger : Passengers) {

if (passenger.getseries() == srs && passenger.getnumber() == nmbr)

{

return &passenger;

}

}

return nullptr;

}

Tariff\* CheckExistanceOfTariff(string place) {

for (auto& tariff : Tariffs) {

if (tariff.getplace() == place) {

return &tariff;

}

}

return nullptr;

}

void ticketpurchase(string nm, string srnm, string db, int srs, int nmbr, string place) {

PassengerInfo\* passenger = CheckExistanceOfPassenger(srs, nmbr);

if (passenger == nullptr) {

Passengers.push\_back(PassengerInfo(nm, srnm, db, srs, nmbr));

passenger = &Passengers.back();

}

Tariff\* tariff = CheckExistanceOfTariff(place);

if (tariff != nullptr) {

passenger->buyticket(\*tariff);

sumSales += tariff->getcost();

}

}

int countsumSales() const {

return sumSales;

}

};

airport\* airport::Airportptr = nullptr;

enum Menu { inputTariff = 1, showTariffs, inputPassport, showPassengerInfo, CountSummaryMoney, quit, wrongInput };

int main()

{

airport\* Airport = airport::getairportptr();

vector<Tariff> Tariffs;

string action;

Menu decision;

do {

cout << "What would you like to do?\n"

<< "1. Input tariff information\n"

<< "2. Show tariff information\n"

<< "3. Input passport information\n"

<< "4. Show Passengers info\n"

<< "5. Count Summary money of bought tickets\n"

<< "6. Exit the program\n"

<< "Your choice: ";

cin >> action;

if (action == "1") decision = inputTariff;

else if (action == "2") decision = showTariffs;

else if (action == "3") decision = inputPassport;

else if (action == "4") decision = showPassengerInfo;

else if (action == "5") decision = CountSummaryMoney;

else if (action == "6") decision = quit;

else {

cout << "Wrong input.\n";

continue;

}

if (decision == inputTariff) {

string place, timeofFlight, price;

cout << "Input destination: ";

cin.ignore();

getline(cin, place);

do {

cout << "Enter price of ticket: ";

cin >> price;

} while (!IsNumber(price) || !IsValidNumber2(price));

do {

cout << "Input time of flight (hh:mm): ";

cin >> timeofFlight;

} while (!IsTime(timeofFlight));

Airport->addTariff(place, stoi(price), timeofFlight);

}

else if (decision == showTariffs) {

const auto& Tariffs = Airport->getTariffs(); // Получаем список тарифов

if (Tariffs.empty()) {

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

}

else {

for (const auto& tariff : Tariffs) {

cout << "Destination: " << tariff.getplace()

<< ", Cost: " << tariff.getcost()

<< ", Time of Flight: " << tariff.gettimeofFlight() << endl;

}

}

}

else if (decision == inputPassport) {

string name, surname, datebirth, series, number, place;

cout << "Enter name: ";

cin >> name;

cout << "Enter surname: ";

cin >> surname;

do {

cout << "Enter date of birth (dd-mm-yyyy): ";

cin >> datebirth;

} while (!IsDate(datebirth)); // Проверка даты рождения

do {

cout << "Enter passport series (4 digits): ";

cin >> series;

} while (!IsValidNumber(series, 4));

do {

cout << "Enter passport number (6 digits): ";

cin >> number;

} while (!IsValidNumber(number, 6));

cout << "Enter destination: ";

cin >> place;

Airport->ticketpurchase(name, surname, datebirth, stoi(series), stoi(number), place);

}

else if (decision == showPassengerInfo) {

string series, number;

do {

cout << "Enter passenger series: ";

cin >> series;

} while (!IsValidNumber(series, 4));

do {

cout << "Enter passenger number: ";

cin >> number;

} while (!IsValidNumber(number, 6));

PassengerInfo\* passenger = Airport->CheckExistanceOfPassenger(stoi(series), stoi(number));

if (passenger) {

cout << "Passenger: " << passenger->getname() << " " << passenger->getsurname() << "\n"

<< "Tickets bought: " << passenger->boughttickets() << " rubles.\n";

}

else {

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

}

}

else if (decision == CountSummaryMoney) {

cout << "Total sum of sold tickets: " << Airport->countsumSales() << " rubles.\n";

}

} while (decision != quit);

cout << "Program exited.\n";

return 0;

}

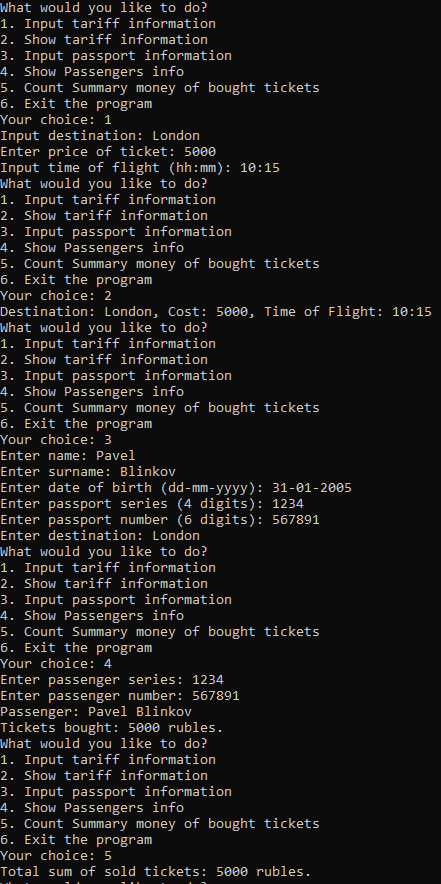
**Пример работы программы**

Рисунок 1 – корректная работа программы

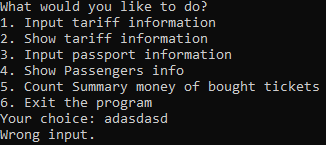


Рисунок 2 – работа программы при некорректно введенных данных

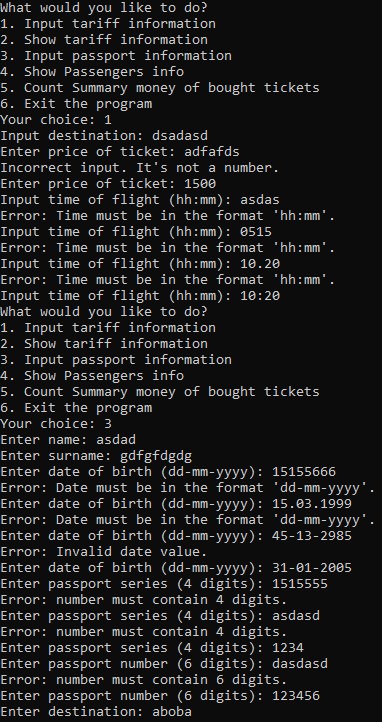


Рисунок 3 – работа программы при некорректно введенных данных

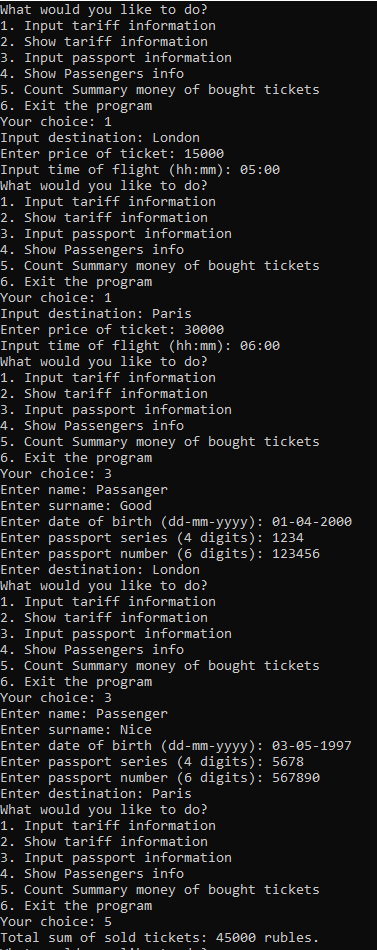


Рисунок 4 – работа функции подсчета суммы проданных билетов

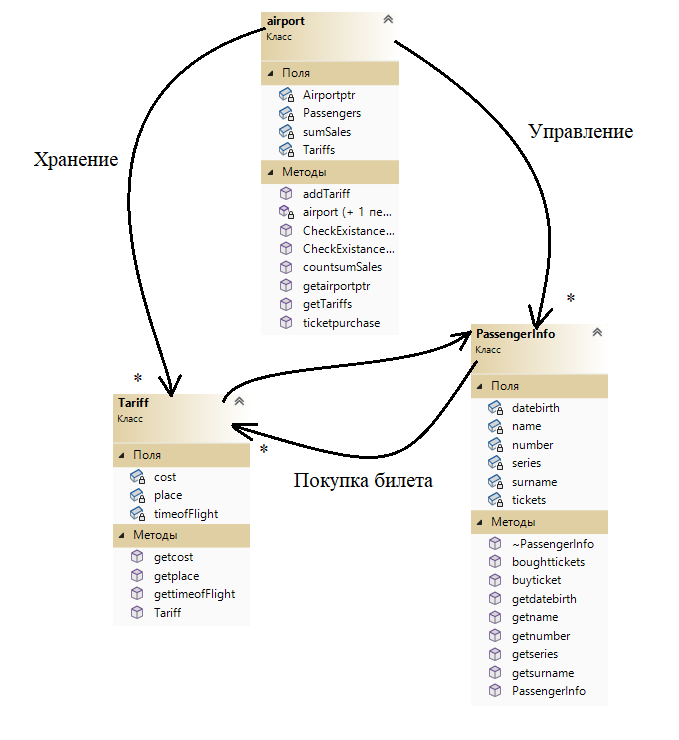


Рисунок 5 – Диаграмма классов

## **Вывод**

В ходе выполнения этой работы, были приобретены навыки создания класса-контейнера и контейнизируемого класса, а также умение сохранять в объекте одного класса объекты других классов.