Лабораторная работа №10

**Тема:** Разработка программ, реализующих механизм наследования классов.

**Цель:** Научится разработке программ, реализующих механизм наследования классов**.**

**Техническое оснащение:** MS Windows 10, MS Word 2019, MS VS 2019, MS Visio 2019.

Выполнение работы

Вариант 18

**Пример 1.** Разработаем класс для работы с одномерным массивом. Создадим в нём индексатор, позволяющий обращаться к элементу массива по индексу.

Проверка работы примера (рисунок 1).

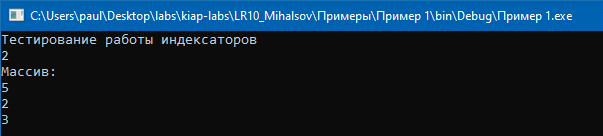


Рисунок 1 – Выполнение работы примера 1

**Пример 2.** Разработаем класс для работы с двумерным массивом (матрицей). Создадим в нём индексатор, позволяющий обращаться к элементу матрицы по индексу.

Проверка работы примера (рисунок 2).

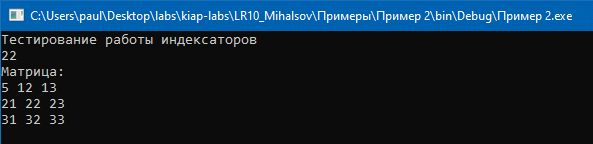


Рисунок 2 – Выполнение работы примера 2

**Пример 3.** Использование абстрактного класса.

Проверка работы примера (рисунок 3).

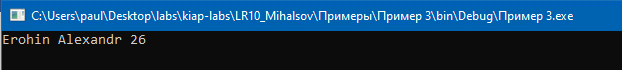


Рисунок 3 – Выполнение работы примера 3

**Пример 4.** Использование наследования

Проверка работы примера (рисунок 4).

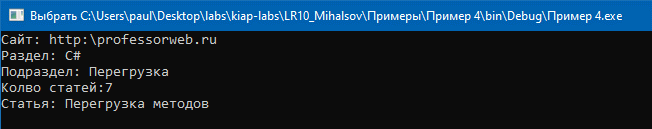


Рисунок 4 – Выполнение работы примера 4

**Пример 5.** Использование конструктора при наследовании

Проверка работы примера (рисунок 5).

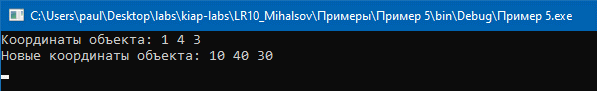


Рисунок 5 – Выполнение работы примера 5

**Пример 6.** Использование базового класса.

Проверка работы примера (рисунок 6).

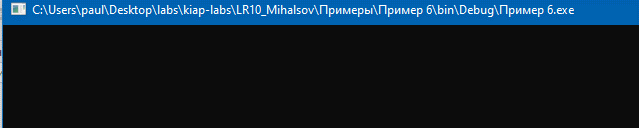


Рисунок 6 – Выполнение работы примера 6

**Пример 7.** Использование виртуальных методов, свойств и индексаторов.

Проверка работы примера (рисунок 7).

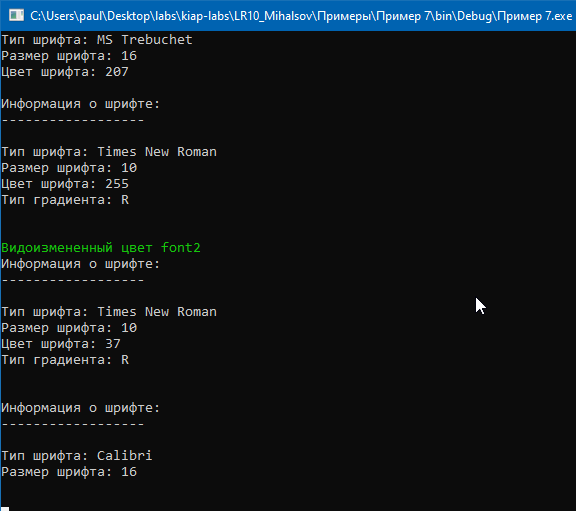


Рисунок 7 – Выполнение работы примера 7

**Задание.** Описать класс «самолет», содержащий следующие закрытые поля:

• название пункта назначения;

• шестизначный номер рейса;

• время отправления.

Описать класс «аэропорт», содержащий закрытый массив самолетов. Обеспечить следующие возможности:

• вывод информации о самолете по номеру рейса с помощью индекса;

• вывод информации о самолетах, отправляющихся в течение часа после введенного с клавиатуры времени;

• вывод информации о самолетах, отправляющихся в заданный пункт назначения;

• перегруженную операцию сравнения, выполняющую сравнение времени отправления двух самолетов.

Информация должна быть отсортирована по времени отправления. Написать программу, демонстрирующую все разработанные элементы классов. Реализовать механизм наследования.

Блок схема алгоритма программы (рисунок 8).



Рисунок 8 – Блок схема алгоритма программы

Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading.Tasks;

namespace Задание

{

public struct Date

{

public int minute;

public int hour;

public int day;

public int mount;

public int year;

public override string ToString()

{

return $"{year.ToString("0000")}-{mount.ToString("00")}-{day.ToString("00")}\_\_{hour.ToString("00")}-{minute.ToString("00")}";

}

public Date newRandomDate(Random rnd)

{

Date d = new Date();

d.year = rnd.Next(2020, 2100);

d.mount = rnd.Next(1, 12);

d.day = rnd.Next(1, 28);

d.hour = rnd.Next(0, 24);

d.minute = rnd.Next(0, 60);

return d;

}

}

abstract class airplane

{

public string PointName

{

get;

private set;

}

public string Number

{

get;

private set;

}

public Date Date

{

get;

private set;

}

protected string Type;

public string info()

{

return $"Рейс номер {Number}, отбудет {Date}, отправляющийся в {PointName}, назначение: {Type}";

}

public airplane(string PointName, string Number, Date Date)

{

this.PointName = PointName;

this.Number = Number;

this.Date = Date;

Type = "error!";

}

public static bool operator >(airplane x, airplane y)

{

if (x.Date.year != y.Date.year) return x.Date.year > y.Date.year ? true : false;

if (x.Date.mount != y.Date.mount) return x.Date.mount > y.Date.mount ? true : false;

if (x.Date.day != y.Date.day) return x.Date.day > y.Date.day ? true : false;

if (x.Date.hour != y.Date.hour) return x.Date.hour > y.Date.hour ? true : false;

return x.Date.minute > y.Date.minute ? true : false;

}

public static bool operator <(airplane x, airplane y)

{

if (x.Date.year != y.Date.year) return x.Date.year < y.Date.year ? true : false;

if (x.Date.mount != y.Date.mount) return x.Date.mount < y.Date.mount ? true : false;

if (x.Date.day != y.Date.day) return x.Date.day < y.Date.day ? true : false;

if (x.Date.hour != y.Date.hour) return x.Date.hour < y.Date.hour ? true : false;

return x.Date.minute < y.Date.minute ? true : false;

}

}

class personePlane : airplane

{

public personePlane(string PointName, string Number, Date Date, string NameofPersone) : base(PointName, Number, Date)

{

Type = $"перелёт {NameofPersone} для личных целей";

}

}

class longPlane : airplane

{

public longPlane(string PointName, string Number, Date Date, string CoutOfPersones) : base(PointName, Number, Date)

{

Type = $"перевозка {CoutOfPersones} пасажиров";

}

}

class cargoPlane : airplane

{

public cargoPlane(string PointName, string Number, Date Date, string cargo, string danger) : base(PointName, Number, Date)

{

Type = $"отправка груза {cargo} с классом опасность {danger}";

}

}

class airport

{

private List<airplane> planes = new List<airplane>();

private string Name;

private uint CorrectNumberOfPlane;

public airport(string Name)

{

this.Name = Name;

this.CorrectNumberOfPlane = 1;

}

private string InitAdd()

{

string num\_s = CorrectNumberOfPlane.ToString("000000");

CorrectNumberOfPlane++;

return num\_s;

}

private void EndAdd()

{

planes.Sort((x,y) => {

return x > y ? -1 : 1;

});

}

public void addPersonePlane(string point, string name, Date Date)

{

string num\_s = InitAdd();

planes.Add(new personePlane(point, num\_s, Date, name));

EndAdd();

}

public void addlongPlane(string point, int count, Date Date)

{

string num\_s = InitAdd();

planes.Add(new longPlane(point, num\_s, Date, count.ToString()));

EndAdd();

}

public void addcargoPlane(string point, string cargo, Date Date, string danger)

{

string num\_s = InitAdd();

planes.Add(new cargoPlane(point, num\_s, Date, cargo, danger));

EndAdd();

}

public airplane this[string num]

{

get

{

foreach (airplane i in planes)

{

if (i.Number == num) return i;

}

return null;

}

}

public List<string> listHourPlaneInfo(int h)

{

List<string> hourPlanes = new List<string>();

foreach(airplane i in planes)

{

if (i.Date.hour == h) hourPlanes.Add(i.info());

}

return hourPlanes;

}

public List<string> listPointPlaneInfo(string Point)

{

List<string> pointPlanes = new List<string>();

foreach (airplane i in planes)

{

if (i.PointName == Point) pointPlanes.Add(i.info());

}

return pointPlanes;

}

public List<string> listAllInfo()

{

List<string> allPlanes = new List<string>();

foreach (airplane i in planes)

{

allPlanes.Add(i.info());

}

return allPlanes;

}

}

class Program

{

static void Main(string[] args)

{

int MAX\_ROWS = 256;

int MIN\_PERSONES = 10;

int MAX\_PERSONES = 100;

List<Tuple<string, string>> cargos = new List<Tuple<string, string>>

{

Tuple.Create<string, string>("cocaine", "A90"),

Tuple.Create<string, string>("heroine", "A10"),

Tuple.Create<string, string>("marihuane", "A0"),

Tuple.Create<string, string>("сигареты", "B1"),

Tuple.Create<string, string>("драг. металы", "С2"),

Tuple.Create<string, string>("алкоголь", "K991"),

Tuple.Create<string, string>("мед. помощь", "O9"),

Tuple.Create<string, string>("escortman", "H8"),

Tuple.Create<string, string>("апарат марки \"ВОЛГА\"", "K7"),

Tuple.Create<string, string>("егурты 0.5", "U8"),

};

List<string> PersoneNames = new List<string>

{

"Панов Юлий",

"Суханов Аввакум",

"Дмитриев Касьян",

"Кузнецов Ян",

"Рыбаков Яков",

"Стрелков Власий",

"Богданов Тимофей",

"Евдокимов Май",

"Константинов Харитон",

"Зайцев Исак",

"Бобылёв Эрик",

"Дмитриев Велор",

"Воронов Владлен",

"Красильников Бронислав",

"Казаков Гурий",

"Блохин Пантелеймон",

"Иванков Прохор",

"Филиппов Игнат",

"Сафонов Азарий",

"Пестов Велорий",

"Третьяков Влас",

"Большаков Севастьян",

"Орлов Платон",

"Большаков Рудольф",

"Кабанов Исак",

"Анисимов Арнольд",

"Борисов Анатолий",

"Зайцев Натан",

"Исаев Людвиг",

"Ковалёв Вольдемар",

"Потапов Ефим",

"Макаров Вальтер",

"Егоров Алексей",

"Молчанов Эрнест",

"Субботин Мирон",

"Захаров Платон",

"Прохоров Геннадий",

"Панфилов Прохор",

"Фролов Святослав",

"Шестаков Иван",

"Евсеев Рудольф",

"Афанасьев Ефим",

"Богданов Алексей",

"Шестаков Любовь",

"Михальцов Павле",

"Кучинский Дмитрий",

"Адольф Гитлер",

"Михаил Палыч",

"Сухач Гнида",

"Налегач Чмоев",

"Цаль Виталий",

"Копченная Гнида",

"Гарик Потный",

"Крылов Вольдемар",

"Никифоров Макар",

"Мухин Корнелий",

"Князев Аполлон",

"Гусев Виталий",

"Третьяков Терентий",

"Носов Леонид",

"Жуков Платон",

"Карпов Иван",

"Сорокин Ипполит",

"Лаврентьев Гордий",

"Иванов Станислав",

"Гаврилов Осип",

"Беспалов Савелий",

"Колобов Людвиг",

"Кириллов Нисон",

"Дроздов Самуил",

"Фролов Мечеслав",

"Беляев Августин",

"Мишин Платон",

};

List<string> Citys = new List<string>

{

"Абингдон (Англия)",

"Амстердам",

"Антверпен",

"Архангельск",

"Астрахань",

"Атланта",

"Афины",

"Баден (Аргау)",

"Базель",

"Бангалор",

"Барселона",

"Бендеры",

"Берлин",

"Берн",

"Биробиджан",

"Бонн",

"Бостон",

"Брайзах-на-Рейне",

"Бремен",

"Будённовск",

"Буффало",

"Буэнос-Айрес",

"Варшава",

"Вашингтон",

"Веймар",

"Вельс",

"Вельск",

"Вена",

"Венеция",

"Винница",

"Владивосток",

"Волгоград",

"Вологда",

"Вормс",

"Воронеж",

"Гамбург",

"Ганновер",

"Гданьск",

"Генуя",

"Глупов",

"Глупов и глуповцы",

"Делфт",

"Денвер",

"Детройт",

"Донецк",

"Дорчестер",

"Дрезден",

"Дублин",

"Дувр",

"Дуйсбург",

"Женева",

"Иваново",

"Иерусалим",

"Измаил",

"Ингельхайм-ам-Райн",

"Иркутск",

"Казань",

"Калгари",

"Кале",

"Калининград",

"Калуга",

"Караганда",

"Каргополь",

"Карлсруэ",

"Кашин",

"Кёльн",

"Киев",

"Киншаса",

"Кишинёв",

"Клайпеда",

"Кола",

"Копенгаген",

"Коряжма",

"Котлас",

"Котор",

"Краков",

"Красноярск",

"Кызыл",

"Лан",

"Лейпциг",

"Лион",

"Лозанна",

"Лондон",

"Лос-Анджелес",

"Луга",

"Львов",

"Льеж",

"Мадрид",

"Майнц",

"Мальборк",

"Мамоново",

"Мангейм",

"Марсель",

"Матера",

"Мезень",

"Мейсен",

"Миасс",

"Милан",

"Минусинск",

"Мирный",

"Монреаль",

"Мурманск",

"Мухосранск",

"Мюнхен",

"Назарет",

"Нарьян-Мар",

"Нахалаль",

"Неаполь",

"Нижний Новгород",

"Нижний Тагил",

"Николаевск-на-Амуре",

"Ницца",

"Новоржев",

"Новосибирск",

"Новый Орлеан",

"Нью-Йорк",

"Нюрнберг",

"Омск",

"Оренбург",

"Оттава",

"Павия",

"Париж",

"Пермь",

"Пиза",

"Подгайцы",

"Полтава",

"Порт-Артур",

"Пошехонье",

"Прага",

"Псков",

"Пушкино",

"Раздельная",

"Рединг (Англия)",

"Реймс",

"Ржев",

"Рига",

"Рим",

"Рио-де-Жанейро",

"Ростов-на-Дону",

"Руан",

"Рязань",

"Салоники",

"Самара",

"Сан-Франциско",

"Сараево",

"Сарапул",

"Северодвинск",

"Сиэтл",

"Советск",

"Сорренто",

"Сочи",

"Стамбул",

"Страсбург",

"Таганрог",

"Тверь",

"Тегеран",

"Тель-Авив",

"Тмутаракань",

"Тобольск",

"Томск",

"Торонто",

"Трир",

"Тулча",

"Турин",

"Тюмень",

"Урюпинск",

"Фонтенбло",

"Хабаровск",

"Ханты-Мансийск",

"Харьков",

"Хельсинки",

"Химки",

"Холмогоры",

"Хьюстон",

"Цюрих",

"Чернигов",

"Чикаго",

"Шаффхаузен",

"Эрфурт",

"Ялуторовск",

};

Random rnd = new Random();

Date rd = new Date();

string airportname = $"аэропорт {Citys[rnd.Next(0, Citys.Count)]} имени {PersoneNames[rnd.Next(0, PersoneNames.Count)]}";

airport ap = new airport(airportname);

int personalPlanes = rnd.Next(0, MAX\_ROWS/3);

for(int i = 0;i < personalPlanes; ++i)

{

ap.addPersonePlane(Citys[rnd.Next(0, Citys.Count)],

PersoneNames[rnd.Next(0, PersoneNames.Count)],

rd.newRandomDate(rnd));

}

int longPlanes = rnd.Next(0, MAX\_ROWS/2);

for (int i = 0; i < personalPlanes; ++i)

{

ap.addlongPlane(Citys[rnd.Next(0, Citys.Count)],

rnd.Next(MIN\_PERSONES, MAX\_PERSONES),

rd.newRandomDate(rnd));

}

int cargoPlanes = rnd.Next(0, MAX\_ROWS/3);

for (int i = 0; i < personalPlanes; ++i)

{

var c = cargos[rnd.Next(0, cargos.Count)];

ap.addcargoPlane(Citys[rnd.Next(0, Citys.Count)],

c.Item1,

rd.newRandomDate(rnd),

c.Item2);

}

Console.WriteLine($"имя аэропорта \"{airportname}\"");

Console.WriteLine($"Созданно персональных рейсов {personalPlanes}");

Console.WriteLine($"Созданно пасажирных рейсов {longPlanes}");

Console.WriteLine($"Созданно грузовых рейсов {cargoPlanes}");

Console.WriteLine($"Всего рейсов {cargoPlanes + longPlanes + personalPlanes}");

int input;

do

{

Console.WriteLine();

Console.WriteLine("0. Выход");

Console.WriteLine("1. Вывести n первых элементов");

Console.WriteLine("2. Операция сравнения");

Console.WriteLine("3. Поиск по времени");

Console.WriteLine("4. Поиск по месту");

input = int.Parse(Console.ReadLine() ?? "0");

switch (input)

{

case 0: break;

case 1:

Console.WriteLine("Сколько?");

input = int.Parse(Console.ReadLine() ?? "0");

foreach (var i in ap.listAllInfo().GetRange(0, input))

{

Console.WriteLine(i);

};

break;

case 2:

Console.WriteLine("Введите рейс 1");

string ind1 = Console.ReadLine();

Console.WriteLine("Введите рейс 2");

string ind2 = Console.ReadLine();

string i1 = ap[ind1].info() ?? "error";

string i2 = ap[ind2].info() ?? "error";

Console.WriteLine($"index#1 = {i1}");

Console.WriteLine($"index#2 = {i2}");

if (i1 == "error" || i2 == "error") break;

Console.WriteLine($"index#1 > index#2 : {ap[ind1] > ap[ind2]}");

Console.WriteLine($"index#1 < index#2 : {ap[ind1] < ap[ind2]}");

break;

case 3:

Console.WriteLine("Введите час");

int h = int.Parse(Console.ReadLine() ?? "-1");

foreach(var i in ap.listHourPlaneInfo(h))

{

Console.WriteLine(i);

}

break;

case 4:

Console.WriteLine("Введите место");

string p = Console.ReadLine() ?? "-1";

foreach (var i in ap.listPointPlaneInfo(p))

{

Console.WriteLine(i);

}

break;

}

} while (input != 0);

Console.ReadKey();

}

}

}

Проверка старта программы (рисунок 9).

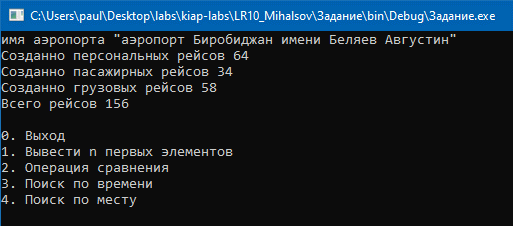


Рисунок 9 – Проверка старта программы

Проверка вывода определенных значений (рисунок 10).

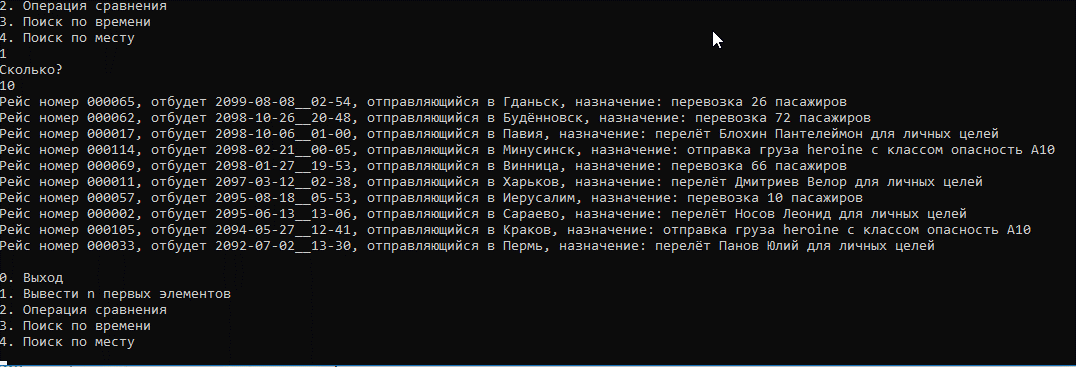


Рисунок 10 – Проверка вывода значений

Проверка операции сравнения (рисунок 11).

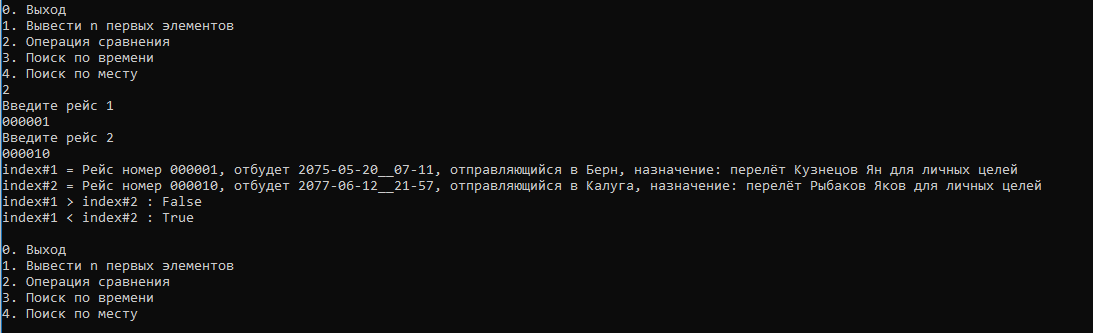


Рисунок 11 – Проверка операции сравнения

Проверка поиска по времени (рисунок 12).

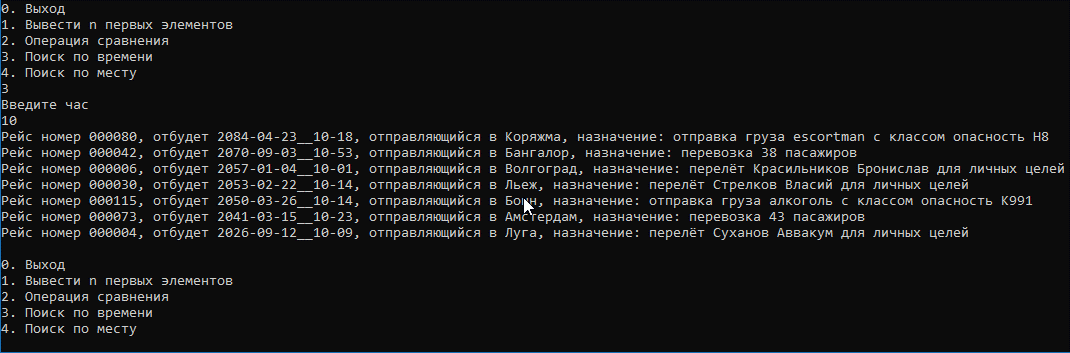


Рисунок 12 – Проверка поиска по времени

Проверка поиска по месту прилёта (рисунок 13).

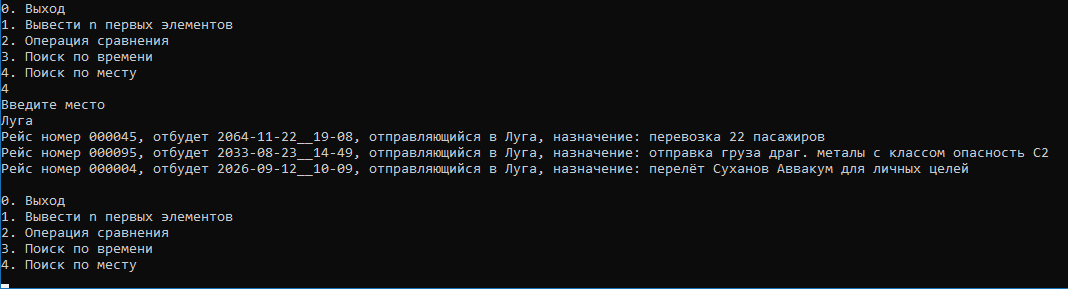


Рисунок 13 – Проверка поиска по месту

**Вывод:** В результате выполнения лабораторной работы были получены практические навыки по использованию механизма наследования в языке программирования C#.