Лабораторная работа №12

**Тема:** Обработка исключительных ситуаций.

**Цель:** Научиться обрабатывать исключительные ситуации, возникновение которых возможно в процессе работы программы.

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

Выполнение работы

Вариант 18

**Пример 1.** Обработка деление на ноль.

Проверка работы примера (рисунок 1).

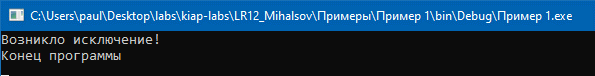


Рисунок 1 – Выполнение примера 1

Пример 2. Обработка ввода.

Проверка работы примера (рисунок 2).

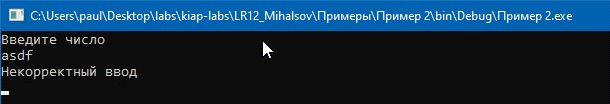


Рисунок 2 – Выполнение примера 2

Пример 3. Проверка двойного исключения.

Проверка работы примера (рисунок 3).

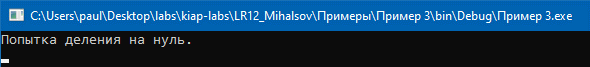


Рисунок 3 – Выполнение примера 3

Пример 4. Получения информации об исключении.

Проверка работы примера (рисунок 4).

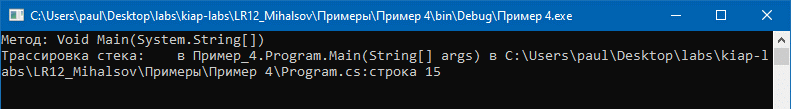


Рисунок 4 – Выполнение примера 4

Пример 5. Вызов исключения при обработке исключения.

Проверка работы примера (рисунок 5).

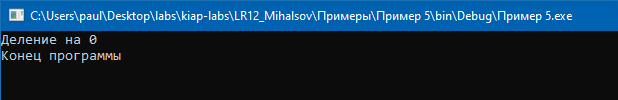


Рисунок 5 – Выполнения примера 5

Пример 6. Валидация данных при исключении.

Проверка работы примера (рисунок 6).

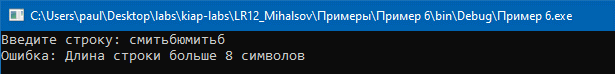


Рисунок 6 – Выполнение примера 6

Пример 7. Двойная обработка исключения.

Проверка работы примера (рисунок 7).

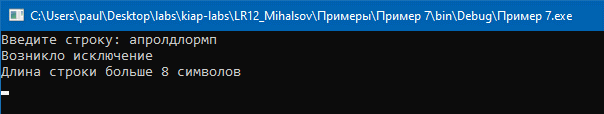


Рисунок 7 – Выполнения примера 7

**Задание.** Модернизируйте программу из лабораторной работы №10, предусмотрев все исключительные ситуации и обработав их.

Листинг программы

using System;

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;

public 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 MIN\_PERSONES = 10;

int MAX\_PERSONES = 100;

Random rnd = new Random();

Date rd = new Date();

airport ap = new airport("аэропорт Жакас-квещан");

ap.addPersonePlane("Минск", "Барон Мюнхаузен", rd.newRandomDate(rnd));

ap.addPersonePlane("Амстердам", "Барон Фуррен оф Дойчланд", rd.newRandomDate(rnd));

ap.addPersonePlane("Караганда", "граф Отто оф Бюргерштрассе", rd.newRandomDate(rnd));

ap.addPersonePlane("Караганда", "граф Широс од Ниццен", rd.newRandomDate(rnd));

ap.addPersonePlane("Краков", "князь Александр", rd.newRandomDate(rnd));

ap.addlongPlane("Омск", rnd.Next(MIN\_PERSONES, MAX\_PERSONES), rd.newRandomDate(rnd));

ap.addlongPlane("Донецк", rnd.Next(MIN\_PERSONES, MAX\_PERSONES), rd.newRandomDate(rnd));

ap.addlongPlane("Дублин", rnd.Next(MIN\_PERSONES, MAX\_PERSONES), rd.newRandomDate(rnd));

ap.addlongPlane("Дрезден", rnd.Next(MIN\_PERSONES, MAX\_PERSONES), rd.newRandomDate(rnd));

ap.addlongPlane("Гамбург", rnd.Next(MIN\_PERSONES, MAX\_PERSONES), rd.newRandomDate(rnd));

ap.addcargoPlane("Нью-Йорк", "вакцина х2020", rd.newRandomDate(rnd), "O1");

ap.addcargoPlane("Новый Орлеан", "слидки золата", rd.newRandomDate(rnd), "O1");

ap.addcargoPlane("Новосибирск", "гуманитарная помощь", rd.newRandomDate(rnd), "O1");

ap.addcargoPlane("Ницца", "игровое оборудование", rd.newRandomDate(rnd), "O1");

List<string> plases = new List<string>

{

"Минск",

"Амстердам",

"Караганда",

"Караганда",

"Краков",

"Омск",

"Донецк",

"Дублин",

"Дрезден",

"Гамбург",

"Нью-Йорк",

"Новый Орлеан",

"Новосибирск",

"Ницца",

};

foreach (var i in ap.listAllInfo())

{

Console.WriteLine(i);

};

int input = -1;

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");

try

{

input = int.Parse(Console.ReadLine());

} catch(System.FormatException)

{

input = -1;

}

if (!(0 <= input && input <= 4))

{

Console.WriteLine("Введено некоретное значение необходимо ввести число от 0 до 4");

input = -1;

}

switch (input)

{

case 0: break;

case 1:

Console.WriteLine("Сколько?");

try

{

input = int.Parse(Console.ReadLine());

} catch (System.FormatException)

{

input = -1;

}

if(!(0 < input && input <= ap.listAllInfo().Count))

{

Console.WriteLine("Введено некоретное значение необходимо ввести число от 1 до {0}", ap.listAllInfo().Count);

input = -1;

continue;

}

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

{

Console.WriteLine(i);

};

break;

case 2:

string ind1 = "";

string ind2 = "";

vvod1:

try

{

Console.WriteLine("Введите рейс 1");

ind1 = Console.ReadLine();

if (!(1 <= int.Parse(ind1) && int.Parse(ind1) <= ap.CorrectNumberOfPlane))

{

Console.WriteLine("Не верный диапозон");

goto vvod1;

}

if(ind1.Length != 6)

{

Console.WriteLine("Не верная маска");

goto vvod1;

}

}

catch (System.FormatException)

{

Console.WriteLine("Данные имеют не верный формат, необходимо ввести цисло от 1 до {0} с маской 000000", ap.CorrectNumberOfPlane);

goto vvod1;

}

vvod2:

try

{

Console.WriteLine("Введите рейс 2");

ind2 = Console.ReadLine();

if (!(1 <= int.Parse(ind2) && int.Parse(ind2) <= ap.CorrectNumberOfPlane))

{

Console.WriteLine("Не верный диапозон");

goto vvod2;

}

if (ind1.Length != 6)

{

Console.WriteLine("Не верная маска");

goto vvod2;

}

}

catch (System.FormatException)

{

Console.WriteLine("Данные имеют не верный формат, необходимо ввести цисло от 1 до {0} с маской 000000", ap.CorrectNumberOfPlane);

goto vvod2;

}

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

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

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 = -1;

try

{

h = int.Parse(Console.ReadLine());

}

catch (System.FormatException)

{

Console.WriteLine("Необходимо ввести число от 0 до 24");

h = -1;

}

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

{

Console.WriteLine(i);

}

break;

case 4:

Console.WriteLine("Введите место");

string p = Console.ReadLine();

bool isFind = false;

foreach (string i in plases) if (i == p) isFind = true;

if(!isFind)

{

Console.WriteLine("Нет такого места в списках!!!");

continue;

}

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

{

Console.WriteLine(i);

}

break;

}

} while (input != 0);

Console.ReadKey();

}

}

}

Проверка ввода некорректного значения в меню выбора действия (рисунок 8).

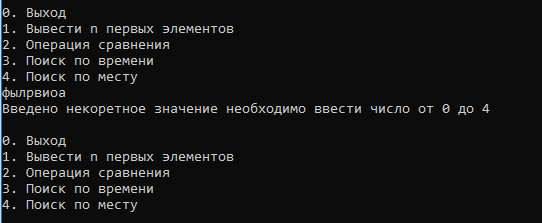


Рисунок 8 – Проверка ввода некорректного значения

Проверка вывести некорректное число рейсов (рисунок 9).

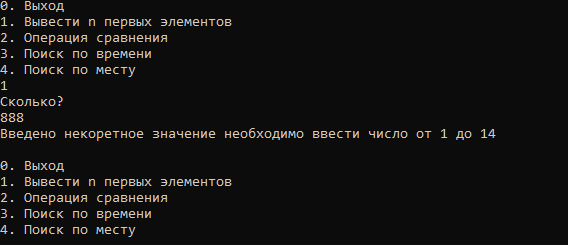


Рисунок 9 – Проверка вывода некорректное количество рейсов

Проверка ввода номера рейса (рисунок 10).

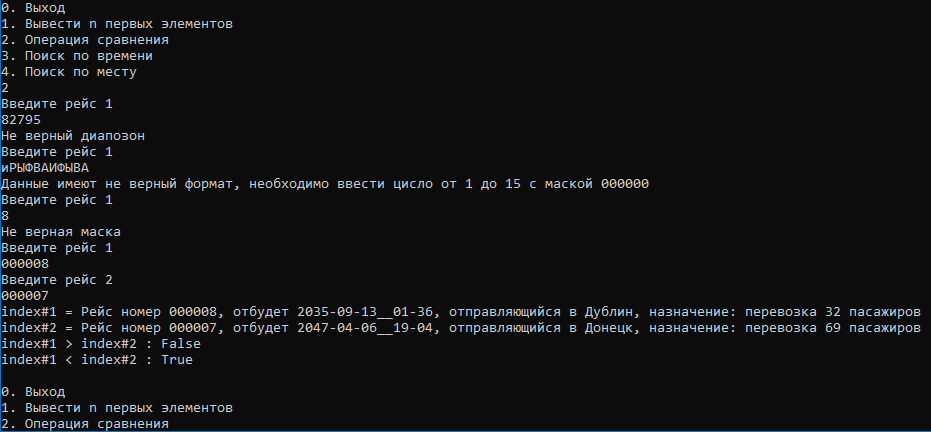


Рисунок 10 – Проверка ввода номера рейса

Проверка ввода времени (рисунок 11).

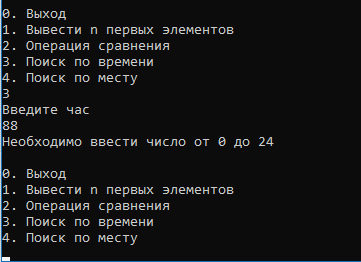


Рисунок 11 – Проверка ввода времени

Проверка ввода пункта назначения (рисунок 12).

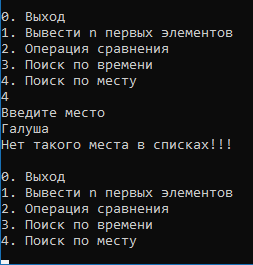


Рисунок 12 – Проверка ввода пункта назначения

**Вывод:** в результате выполнения лабораторной работы были получены практические навыки по обработки программных исключений различными способами.