Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 19

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Запросы LINQ»

Выполнил: ст. гр. ТУУ-211

Голюдбин А.В.

Вариант №3

Проверил: к.т.н., доц. Сафронов А.И.

1. **Формулировка цели работы**

Освоить навыки подготовки наборов данных по нечётко сформулированному описанию предметной области при учёте особенностей запросов, отклики на которые необходимо предоставлять заявленным (потенциальным) пользователям программного обеспечения.

1. **Описание задачи**

Описание предметной области для подготовки данных. Пользователями являются сотрудники из числа руководящего состава кафедры. Необходима информация о преподавателях и инженерах, работающих на кафедре, а также об их занятости (по научно-исследовательским работам, преподаваемым дисциплинам, организационной работе с обучающимися, методической работе).

Реализовать запросы *LINQ*:

– выдавать сводную информацию обо всех работниках кафедры;

– выдавать информацию о занятости преподавателя научно-исследовательской работой;

– выдавать информацию о занятости преподавателя организационной работой с обучающимися;

– выдавать информацию о занятости преподавателя методической работой;

– выдавать информацию о преподавателе, ведущем указанный вид занятий по указанной дисциплине;

– выдавать информацию о видах занятий, которые проводятся по выбранной из списка (впечатывать наименования запрещено) дисциплине.

1. **Сеть Петри запрограммированного технологического процесса**

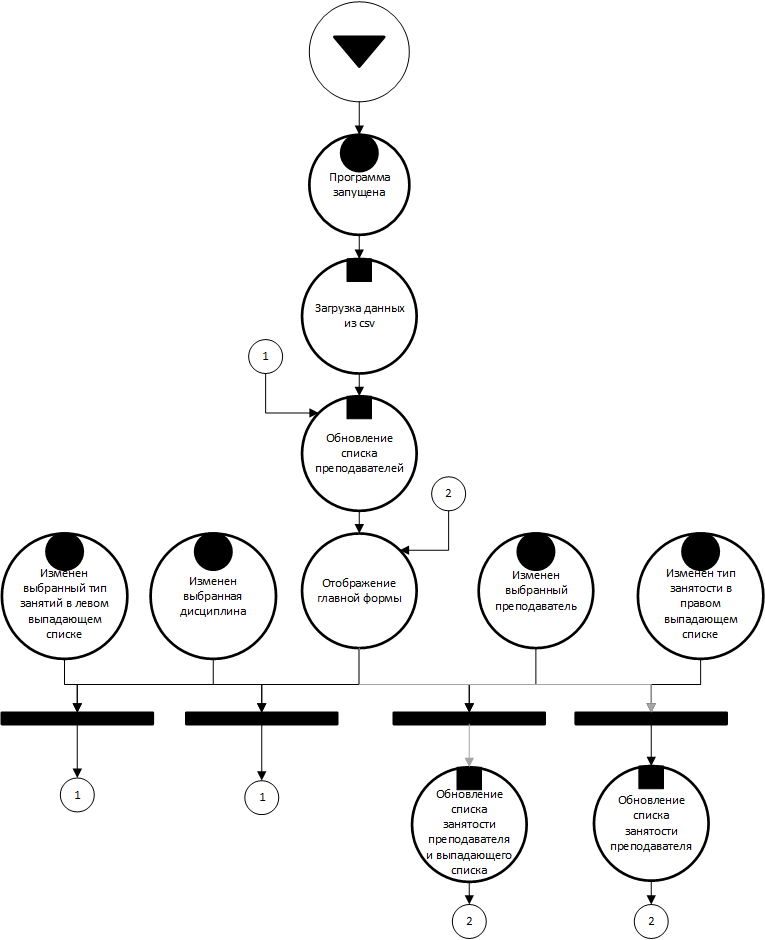


Рисунок 1 – Сеть Петри запрограммированного технологического процесса

1. **Схемы алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка )**

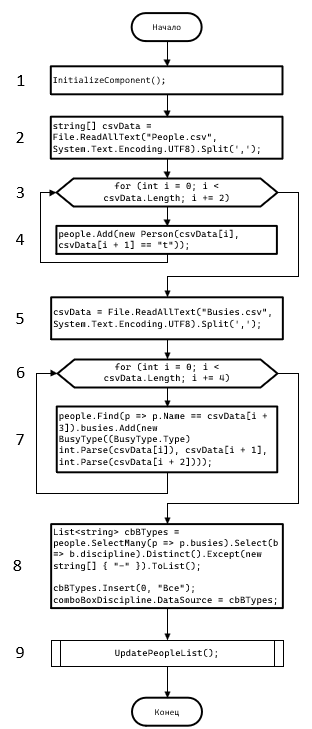


Рисунок 2 – Загрузка данных из *csv*

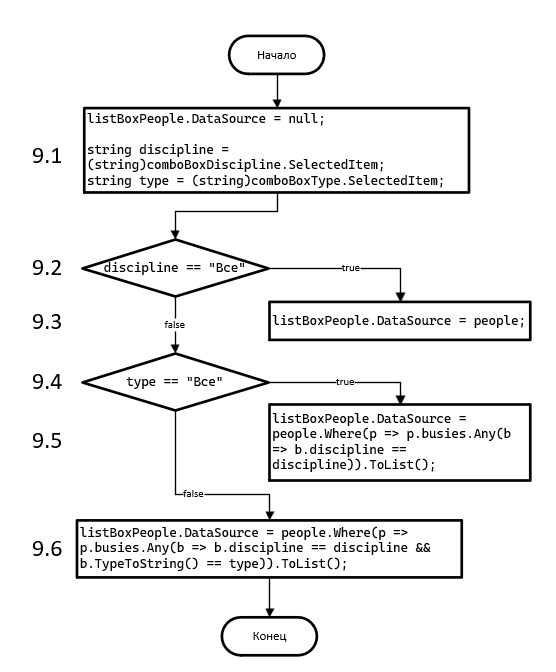


Рисунок 3 – Обновление списка преподавателей

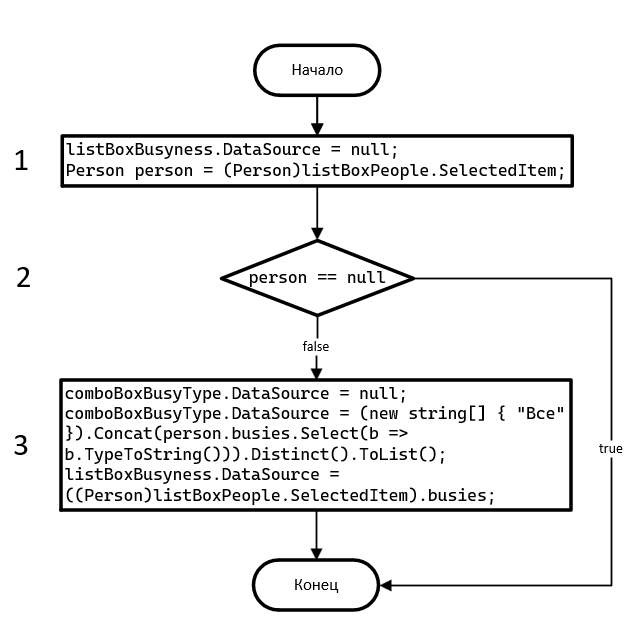


Рисунок 4 – Обновление списка занятости преподавателя и выпадающего списка

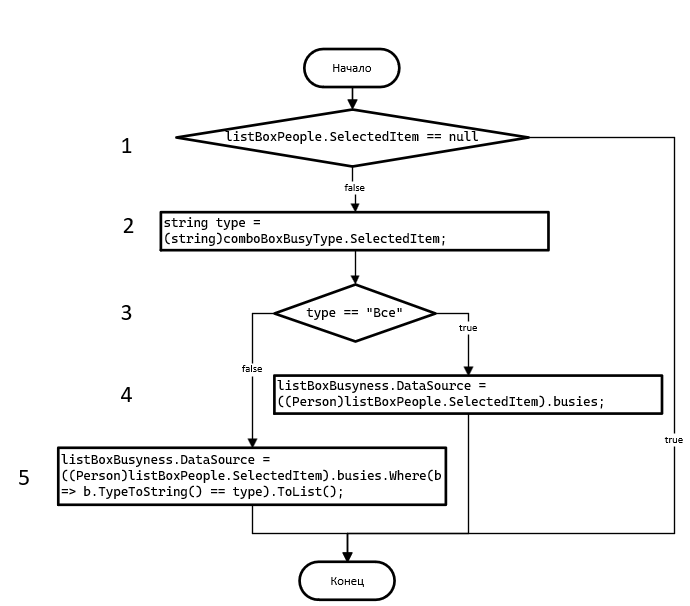


Рисунок 5 – Обновление списка занятости преподавателя

1. **Подбор тестовых примеров**

В качестве тестовых данных созданы два преподавателя и два инженера со следующими занятостями:

Инженер Иванов И.И.:

Методическая работа, 20 часов

Геология, практика, 11 часов

Геология, лабораторная работа, 8 часов

Геология, научно-исследовательская работа, 5 часов

Преподаватель Петров А.Б.:

Организационная работа, 35 часов

Методическая работа, 30 часов

Преподаватель Мюслин Е.Н.:

Программирование, практика. 9 часов

Астрономия, лабораторная работа. 27 часов

ОБЖ, лекция. 18 часов

Программирование, лекция. 25 часов

Для хранения данных *csv*-фалы были заполнены следующим содержанием:

*People.csv*:

Иванов И.И.,f,Петров А.Б.,t,Мюслин Е.Н.,t,Андреев К.А.,f

*Busies.csv*:

2,-,20,Иванов И.И.,4,Геология,11,Иванов И.И.,5,Геология,8,Иванов И.И.,0,Геология,5,Иванов И.И.,1,-,35,Петров А.Б.,2,-,30,Петров А.Б.,4,Программирование,9,Мюслин Е.Н.,5,Астрономия,27,Мюслин Е.Н.,3,ОБЖ,18,Мюслин Е.Н.,3,Программирование,25,Мюслин Е.Н.

1. **Листинг (код) составленного программного обеспечения**

**Файл *Form1.cs***

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace LINQ

{

public partial class Form1 : Form

{

private List<Person> people = new List<Person>();

public Form1()

{

InitializeComponent();

string[] csvData = File.ReadAllText("People.csv", System.Text.Encoding.UTF8).Split(',');

for (int i = 0; i < csvData.Length; i += 2)

people.Add(new Person(csvData[i], csvData[i + 1] == "t"));

csvData = File.ReadAllText("Busies.csv", System.Text.Encoding.UTF8).Split(',');

for (int i = 0; i < csvData.Length; i += 4)

people.Find(p => p.Name == csvData[i + 3]).busies.Add(new BusyType((BusyType.Type) int.Parse(csvData[i]), csvData[i + 1], int.Parse(csvData[i + 2])));

List<string> cbBTypes = people.SelectMany(p => p.busies).Select(b => b.discipline).Distinct().Except(new string[] { "-" }).ToList();

cbBTypes.Insert(0, "Все");

comboBoxDiscipline.DataSource = cbBTypes;

UpdatePeopleList();

}

public void UpdatePeopleList()

{

listBoxPeople.DataSource = null;

string discipline = (string)comboBoxDiscipline.SelectedItem;

string type = (string)comboBoxType.SelectedItem;

if (discipline == "Все")

listBoxPeople.DataSource = people;

else if (type == "Все")

listBoxPeople.DataSource = people.Where(p => p.busies.Any(b => b.discipline == discipline)).ToList();

else

listBoxPeople.DataSource = people.Where(p => p.busies.Any(b => b.discipline == discipline && b.TypeToString() == type)).ToList();

}

private void listBoxPeople\_SelectedIndexChanged(object sender, System.EventArgs e)

{

listBoxBusyness.DataSource = null;

Person person = (Person)listBoxPeople.SelectedItem;

if (person == null)

return;

comboBoxBusyType.DataSource = null;

comboBoxBusyType.DataSource = (new string[] { "Все" }).Concat(person.busies.Select(b => b.TypeToString())).Distinct().ToList();

listBoxBusyness.DataSource = ((Person)listBoxPeople.SelectedItem).busies;

}

private void comboBoxDiscipline\_SelectedIndexChanged(object sender, System.EventArgs e)

{

if (comboBoxDiscipline.SelectedItem == null)

return;

UpdatePeopleList();

}

private void comboBoxType\_SelectedIndexChanged(object sender, System.EventArgs e)

{

if (comboBoxType.SelectedItem == null)

return;

UpdatePeopleList();

}

private void comboBoxBusyType\_SelectedIndexChanged(object sender, System.EventArgs e)

{

if (listBoxPeople.SelectedItem == null)

return;

string type = (string)comboBoxBusyType.SelectedItem;

if (type == "Все")

listBoxBusyness.DataSource = ((Person)listBoxPeople.SelectedItem).busies;

else

listBoxBusyness.DataSource = ((Person)listBoxPeople.SelectedItem).busies.Where(b => b.TypeToString() == type).ToList();

}

}

}

**Файл *Person.cs***

using System.Collections.Generic;

using System.Linq;

namespace LINQ

{

public class Person

{

public string Name { get; private set; }

public bool isTeacher { get; private set; }

public List<BusyType> busies = new List<BusyType>();

public Person(string name, bool isTeacher)

{

Name = name;

this.isTeacher = isTeacher;

}

public override string ToString()

{

return (isTeacher ? "Преподаватель " : "Инженер ") + Name + " Всего ч/м: " + busies.Sum(b => b.hoursPerMonth);

}

}

}

**Файл *BusyType.cs***

namespace LINQ

{

public class BusyType

{

public enum Type { ScienseWork, Management, Metodical, Lecture, Practice, LabWork }

public Type type { get; private set; }

public string discipline { get; private set; }

public int hoursPerMonth { get; private set; }

public BusyType(Type type, string discipline, int hoursPerMonth)

{

this.type = type;

this.discipline = discipline;

this.hoursPerMonth = hoursPerMonth;

}

public string TypeToString()

{

if (type == Type.ScienseWork)

return "Научно-исследовательская работа";

else if (type == Type.Management)

return "Организационная работа";

else if (type == Type.Metodical)

return "Методическая работа";

else if (type == Type.Lecture)

return "Лекция";

else if (type == Type.Practice)

return "Практика";

else if (type == Type.LabWork)

return "Лабораторная работа";

return "NotFound";

}

public override string ToString()

{

return TypeToString() + ", " + discipline + ". " + hoursPerMonth + "ч/м";

}

}

}

1. **Графический пользовательский интерфейс программного обеспечения и его описание**

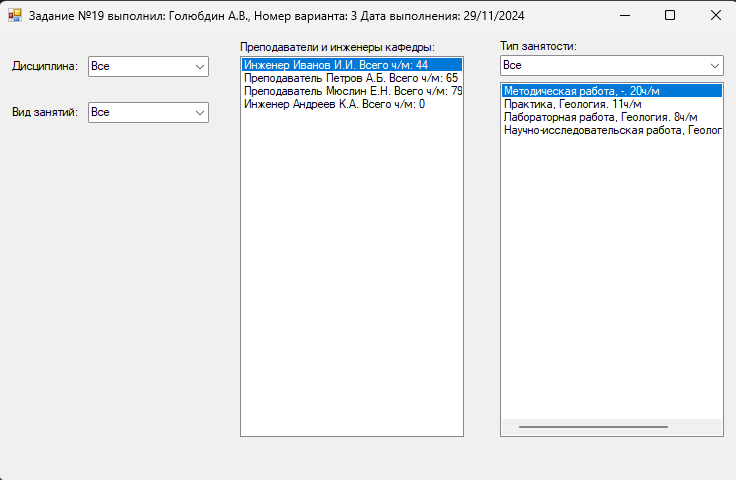
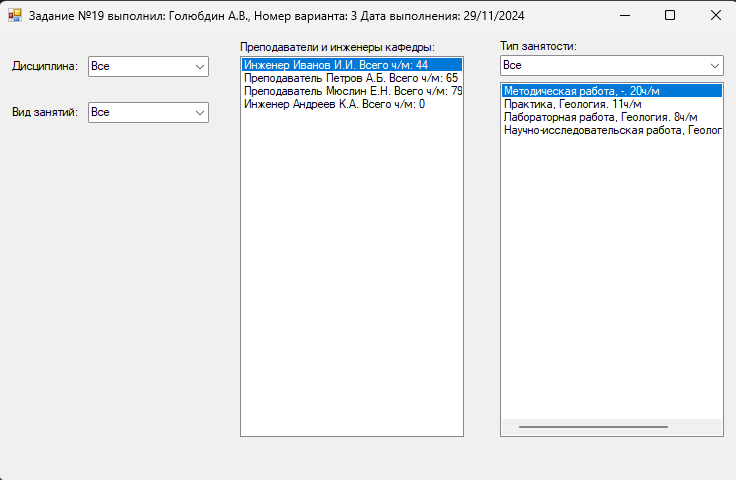


Рисунок 6 – Интерфейс главной формы

В левой части формы находятся формы для поиска сотрудников по дисциплине и виду занятий. По центру список преподавателей, отфильтрованных по введенным слева параметрам. В правой части формы находится список занятостей выбранного в центральном списке преподавателя, отфильтрованный по типам занятости.

1. **Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению**

 Рисунок 7 – Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

В центральной части формы мы можем видеть сводную информацию обо всех работниках кафедры, включая имя, должность и количество часов.

В правом списке с помощью выпадающего списка мы можем вывести информацию о занятости преподавателя научно-исследовательской, методической и организационной работой.

С помощью выпадающих списков слева мы можем выдать информацию о преподавателях, ведущих указанный вид занятий по указанной дисциплине.

Таким образом, графический интерфейс программы соответствует требованиям к разработке.

1. **Расчёт тестовых примеров с использованием составленного программного обеспечения**

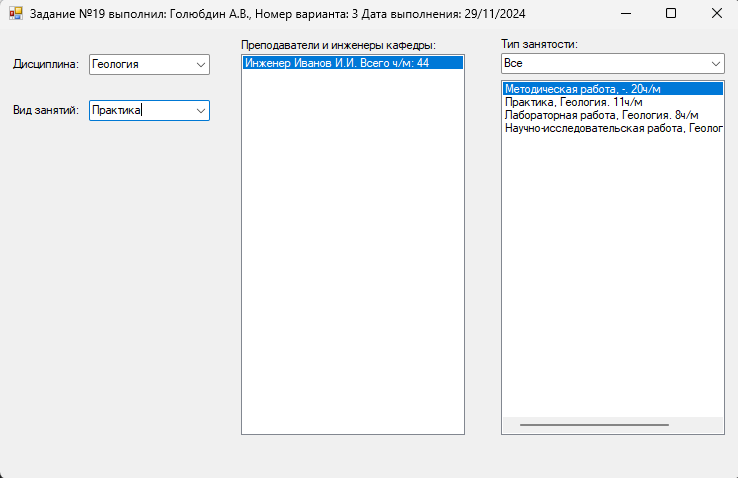


Рисунок 8 – Поиск преподавателя, ведущего практику по геологии

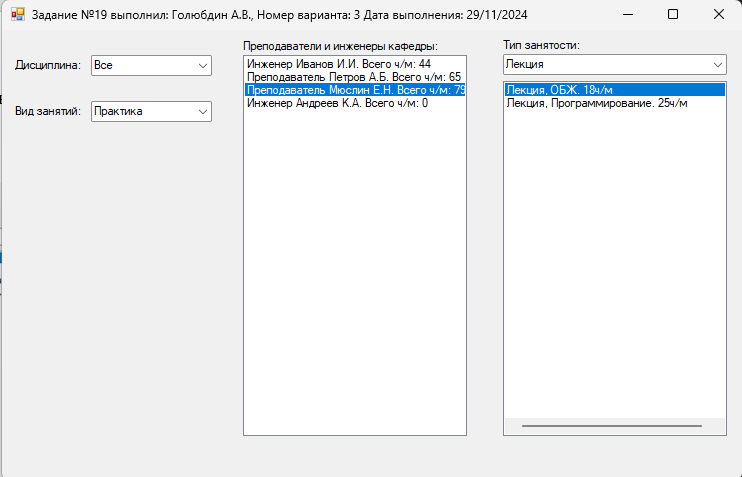


Рисунок 9 – Фильтрация занятости преподавателя по типу – «Лекция»

Не показана ситуация, когда в фильтр вводится опечатка, например, «Парктика»

1. **Формулировка вывода о проделанной работе**

Освоил навыки подготовки наборов данных по нечётко сформулированному описанию предметной области при учёте особенностей запросов, отклики на которые необходимо предоставлять заявленным (потенциальным) пользователям программного обеспечения.

Освоил навыки работы с библиотекой *LINQ* и применил их на практике в данной лабораторной работе.