**Задание 19.**

В интегрированной среде разработки *Microsoft* *Visual Studio* разработать программу в режиме *Windows Forms Application* на языке *Visual C#*, представляющую собой инструмент фильтрации наборов данных согласно фиксированным критериям отбора*.*

**В качестве результата выполненной работы сдаётся**:

1. Архив *\*.rar* или *\*.zip*, содержащий разработанное решение (*solution*). Для отправки архива по электронной почте рекомендуется использовать сервисы файлового обмена, генерирующие ссылки на файлы (например, *Google Drive*, *Яндекс.Диск*, *Облако Mail.Ru* и тому подобные), поскольку протоколы электронной почты часто автоматически отклоняют архивы, содержащие исполняемые (*exe*) файлы.

2. Файлы формата *\*.csv*, хранящие информацию, необходимую для формирования результатов запросов *LINQ*.

3. Отчёт в формате *Microsoft Office Word* по выполненной работе.

При работе на платформе *GitHub* размещение материалов 1-3 выполняется в удобном для автора формате (архив и/или каталоги).

**Внимание!** Не забудьте выполнить автоматическую нумерацию страниц в отчёте.

Отчёт по выполненной работе может быть подготовлен при учёте представленной ниже структуры:

0. Титульный лист.

1. Формулировку цели работы.

2. Описание задачи согласно выданному варианту.

3. Составление сети Петри запрограммированного технологического процесса.

4. Составление схем алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка ).

5. Подбор тестовых примеров.

6. Листинг (код) составленного программного обеспечения.

7. Графический пользовательский интерфейс программного обеспечения и его описание.

8. Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

9. Расчёт тестовых примеров с использованием составленного программного обеспечения.

10. Формулировку вывода о проделанной работе.

Рекомендации к отчёту, доказывающие самостоятельность выполнения работы и упрощающие процедуру проверки отчёта преподавателем:

1. Выполнение дополнительных скриншотов для случаев, когда текстовое описание проделанных действий становится громоздким или трудным к восприятию.

2. Нумерация рисунков (если есть) с подписями, содержащими названия рисунков, например, «Рисунок 1 – Схема алгоритма программы».

**Цель работы (одна из возможных формулировок)**: 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.

**Рабочее задание:**

1. В тексте индивидуального задания согласно варианту (с учётом текста запросов *LINQ*) подчеркнуть все существительные, выписать их отдельно, устранив повторы – это файлы «*csv*», которые необходимо создать и наполнить содержимым.

2. Продумать и создать эргономичный графический пользовательский интерфейс для удобства отправки запросов и просмотра результатов запросов.

3. Запланировать выбор фильтра из комбинированного списка.

4. Результаты запросов демонстрировать в обычном экранном списке.

5. В не анонимных классах предусмотреть переопределение метода *ToString()* для корректного вывода информации в экранный список.

6. Стараться избегать возникновения анонимных классов в создаваемом программном обеспечении.

**Памятка программисту №1. Переход от циклов к запросам**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace tstLinqFromMetanit01

{

public partial class frmMain : Form

{

public frmMain()

{

InitializeComponent();

}

string[] people = { "Владимир", "Алексей",

"Иван", "Вадим", "Владислав", "Николай" };

string[] filters =

{ "(не выбран)",

"Имена, начинающиеся с \"В\" (условие)",

"Имена, начинающиеся с \"В\" (LINQ)",

"Имена, начинающиеся с \"В\" (LINQ Ext)",

"Количество имён, начинающиеся с \"В\" (LINQ Ext)"

};

private void RefreshList(IList<string> col)

{

lstMain.Items.Clear();

foreach (string s in col)

lstMain.Items.Add(s);

}

private void btnOrigin\_Click(object sender, EventArgs e)

{

//создаем новый список для результатов

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

//проходим по массиву

foreach (string person in people)

selectedPeople.Add(person);

//сортируем список

selectedPeople.Sort();

RefreshList(selectedPeople);

}

private void NoFilt()

{

//создаем новый список для результатов

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

RefreshList(selectedPeople);

}

private void NamesStartsWithT()

{

//создаем новый список для результатов

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

//проходим по массиву

foreach (string person in people)

{

//если строка начинается на букву T,

//добавляем в список

if (person.ToUpper().StartsWith("В"))

selectedPeople.Add(person);

}

//сортируем список

selectedPeople.Sort();

RefreshList(selectedPeople);

}

private void NamesStartsWithTLinq()

{

//создаем новый список для результатов

List<string> selectedPeople =

//передаем каждый элемент из people в переменную p <--

( from p in people

//фильтрация по критерию

where p.ToUpper().StartsWith("В")

//упорядочиваем по возрастанию

orderby p

//передаём объект в создаваемую коллекцию

select p).ToList();

RefreshList(selectedPeople);

}

private void NamesStartsWithTLinqExt()

{

// создаем новый список для результатов

List<string> selectedPeople =

//передаём объект в создаваемую коллекцию

people.Where(p => p.ToUpper().

StartsWith("В")).

OrderBy(p => p).ToList();

RefreshList(selectedPeople);

}

private void NamesStartsWithTLinqCount()

{

int number = (from p in people where p.ToUpper().StartsWith("В") select

p).Count();

List<string> selectedPeople = new List<string>()

{

number.ToString()

};

RefreshList(selectedPeople);

}

private void frmMain\_Load(object sender, EventArgs e)

{

cmbFilt.Items.AddRange(filters);

cmbFilt.SelectedIndex = 0;

}

private void cmbFilt\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (cmbFilt.SelectedIndex)

{

case 1:

NamesStartsWithT();

break;

case 2:

NamesStartsWithTLinq();

break;

case 3:

NamesStartsWithTLinqExt();

break;

case 4:

NamesStartsWithTLinqCount();

break;

default:

NoFilt();

break;

}

}

}

}

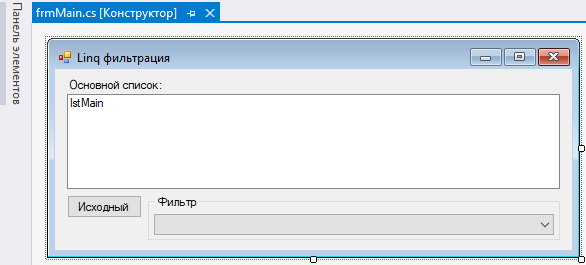


Рисунок 1 – Конструктив главной экранной формы

**Памятка программисту №2. Переход к спискам объектов**

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace tstLinqFromMetanit02

{

public partial class frmMain : Form

{

public frmMain()

{

InitializeComponent();

}

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

{

new Person ("Владимир", 29),

new Person ("Алексей", 22),

new Person ("Иван", 20),

new Person ("Вадим", 31),

new Person ("Владислав", 27),

new Person ("Николай", 29)

};

string[] filters =

{ "(не выбран)",

"Показать имена (LINQ)",

"Показать имена (LINQ Ext)",

"Показать годы рождения (LINQ)",

"Показать годы рождения (LINQ Ext)",

"Показать годы рождения + переменные (LINQ)"

};

private void RefreshList(IList<object> col)

{

lstMain.Items.Clear();

foreach (object s in col)

lstMain.Items.Add(s);

}

private void btnOrigin\_Click(object sender, EventArgs e)

{

//создаем новый список для результатов

List<object> selectedPeople = new List<object>();

//проходим по массиву

foreach (Person person in people)

selectedPeople.Add(person);

RefreshList(selectedPeople);

}

private void NoFilt()

{

//создаем новый список для результатов

List<object> selectedPeople = new List<object>();

RefreshList(selectedPeople);

}

private void ShowNames()

{

List<object> colNames = (from p in people select p.Name as object).ToList();

RefreshList(colNames);

}

private void ShowYears()

{

List<object> colPersonel =

(from p in people

select new

{

FirstName = p.Name,

Year = DateTime.Now.Year - p.Age

} as object).ToList();

RefreshList(colPersonel);

}

private void LetShowYears()

{

List<object> colPersonel =

(from p in people

let name = $"Тов. {p.Name}"

let year = DateTime.Now.Year - p.Age

select new

{

FirstName = name,

Year = year

} as object).ToList();

RefreshList(colPersonel);

}

private void ShowYearsExt()

{

List<object> colPersonel = people.Select(p => new

{

FirstName = p.Name,

Year = DateTime.Now.Year - p.Age

} as object).ToList();

RefreshList(colPersonel);

}

private void ShowNamesExt()

{

//создаем новый список для результатов

List<object> selectedPeople = people.Select(p => p.Name as object).ToList();

RefreshList(selectedPeople);

}

private void frmMain\_Load(object sender, EventArgs e)

{

cmbFilt.Items.AddRange(filters);

cmbFilt.SelectedIndex = 0;

}

private void cmbFilt\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (cmbFilt.SelectedIndex)

{

case 1:

ShowNames();

break;

case 2:

ShowNamesExt();

break;

case 3:

ShowYears();

break;

case 4:

ShowYearsExt();

break;

case 5:

LetShowYears();

break;

default:

NoFilt();

break;

}

}

}

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

public Person (string Name, int Age)

{

this.Name = Name;

this.Age = Age;

}

public override string ToString() => $"Сотрудник: {Name}, Возраст: {Age}";

}

}

ГПИ к данному программному приложению аналогичен изображённому на Рисунке 1 в Приложении №1 постановки.

**Памятка программисту №3. Работа с полями анонимных классов**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace tstLinqFromMetanit03\_

{

public partial class frmMain : Form

{

public frmMain()

{

InitializeComponent();

}

List<object> people = new List<object>()

{

new Person ("Владимир", 29),

new Person ("Алексей", 22),

new Person ("Иван", 20),

new Person ("Вадим", 31),

new Person ("Владислав", 27),

new Person ("Николай", 29)

};

List<object> companies = new List<object>()

{

new Company ("Яндекс", new List<object>()),

new Company ("Рамблер", new List<object>()),

new Company ("вК", new List<object>())

};

List<object> courses = new List<object>()

{

new Course ("C#"),

new Course ("C++"),

new Course ("Java"),

new Course ("Python"),

new Course ("Pascal"),

new Course ("Delphi")

};

List<object> students = new List<object>()

{

new Student ("Иван"),

new Student ("Алексей"),

new Student ("Владислав"),

new Student ("Николай"),

new Student ("Владимир"),

new Student ("Кирилл"),

new Student ("Дмитрий"),

new Student ("Александр"),

new Student ("Григорий"),

new Student ("Фёдор"),

new Student ("Андрей"),

new Student ("Георгий"),

};

IEnumerable<object> currCol;

string[] filters =

{ "(не выбран)",

"Курсы",

"Обучающиеся",

"Сотрудники",

"Компании",

"Каждому студенту - каждый курс (LINQ)",

"Перечень сотрудников из компаний (LINQ Ext)",

"Перечень сотрудников из компаний (LINQ)",

"Перечень сотрудников из компаний + компания (LINQ Ext)",

"Перечень сотрудников из компаний + компания (LINQ)",

};

private void RefreshList(IList<object> col)

{

lstMain.Items.Clear();

foreach (object s in col)

lstMain.Items.Add(s);

}

private void NoFilt() => RefreshList(new List<object>());

private void ShowCourses() => RefreshList(courses);

private void ShowStudents() => RefreshList(students);

private void ShowWorkers() => RefreshList(people);

private void ShowCompanies() => RefreshList(companies);

private void SetEveryCourseToEveryStudent()

{

IList<object> colEnrollments =

(from c in courses

from s in students

select new

{

(s as Student).Name, (c as Course).Title

} as object).ToList();

RefreshList(colEnrollments);

}

private void ShowCompanyWorkersExt()

{

List<object> colWorkers =

companies.SelectMany(c => (c as Company).Staff).ToList();

RefreshList(colWorkers);

}

private void ShowCompanyWorkers()

{

List<object> colWorkers =

(from c in companies

from wrk in (c as Company).Staff

select wrk).ToList();

RefreshList(colWorkers);

}

private void ShowCompanyWorkersInCompanyExt()

{

List<object> colWorkersInCompany =

companies.SelectMany(c => (c as Company).Staff,

(c, wrk) => new

{

(wrk as Person).Name,

Company = (c as Company).Name

} as object).ToList();

RefreshList(colWorkersInCompany);

}

private void ShowCompanyWorkersInCompany()

{

List<object> colWorkers =

(from c in companies

from wrk in (c as Company).Staff

select new

{

(wrk as Person).Name,

Company = (c as Company).Name

} as object).ToList();

RefreshList(colWorkers);

}

private void frmMain\_Load(object sender, EventArgs e)

{

int counter = 0;

cmbFilt.Items.AddRange(filters);

foreach (Company c in companies)

{

c.Staff.Add(people[counter]);

c.Staff.Add(people[counter + 1]);

counter += 2;

}

cmbFilt.SelectedIndex = 0;

}

private void cmbFilt\_SelectedIndexChanged(object sender, EventArgs e)

{

switch (cmbFilt.SelectedIndex)

{

case 1:

ShowCourses();

break;

case 2:

ShowStudents();

break;

case 3:

ShowWorkers();

break;

case 4:

ShowCompanies();

break;

case 5:

SetEveryCourseToEveryStudent();

break;

case 6:

ShowCompanyWorkersExt();

break;

case 7:

ShowCompanyWorkers();

break;

case 8:

ShowCompanyWorkersInCompanyExt();

break;

case 9:

ShowCompanyWorkersInCompany();

break;

default:

NoFilt();

break;

}

}

private void lstMain\_SelectedIndexChanged(object sender, EventArgs e)

{

object o;

switch (cmbFilt.SelectedIndex)

{

case 5:

o = lstMain.Items[lstMain.SelectedIndex];

MessageBox.Show(o.GetType().GetProperty("Name").GetValue(o,

null).ToString() + " - " +

o.GetType().GetProperty("Title").GetValue(o, null).ToString());

break;

case 8:

case 9:

o = lstMain.Items[lstMain.SelectedIndex];

MessageBox.Show(o.GetType().GetProperty("Name").GetValue(o,

null).ToString() + " - " +

o.GetType().GetProperty("Company").GetValue(o, null).ToString());

break;

default:

break;

}

}

}

public class Course

{

public string Title { get; set; }

public Course(string Title)

{

this.Title = Title;

}

public override string ToString() => $"Курс: {Title}";

}

public class Student

{

public string Name { get; set; }

public Student(string Name)

{

this.Name = Name;

}

public override string ToString() => $"Обучающийся: {Name}";

}

public class Person

{

public string Name { get; set; }

public int Age { get; set; }

public Person(string Name, int Age)

{

this.Name = Name;

this.Age = Age;

}

public override string ToString() => $"Сотрудник: {Name}, Возраст: {Age}";

}

public class Company

{

public string Name { get; set; }

public List<object> Staff { get; set; }

public Company(string Name, List<object> Staff)

{

this.Name = Name;

this.Staff = Staff;

}

private string ShowStaff(List<object> Staff)

{

string str = "";

foreach (object person in Staff)

str += (person as Person).Name + ", ";

return str.Substring(0, str.Length - 2);

}

public override string ToString() => $"Компания: {Name}, Сотрудники: {ShowStaff(Staff)}";

}

public class NameAndCourse

{

public string Name;

public string Title;

}

}

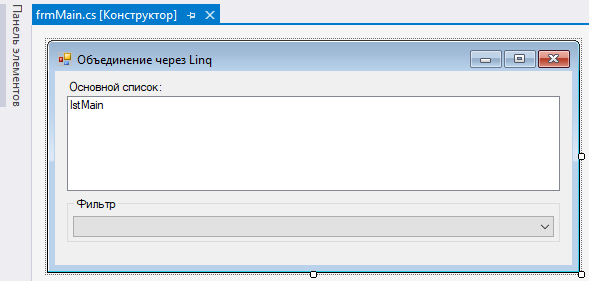


Рисунок 2 – Конструктив главной экранной формы без кнопок

**Вариант №1. Тема «Магазин по продаже оргтехники»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники некоторого магазина оргтехники. Магазин продаёт типовой товар. Постоянным клиентам, а также покупателям, набравшим товар на сумму *R* рублей, предоставляется скидка.

Реализовать запросы *LINQ*:

– показывать весь имеющийся товар с соответствующими ценами за единицу;

– находить товар по введённому наименованию (или фрагменту наименования);

– находить товар по производителю (или фрагменту имени производителя);

– формировать товарный чек (корзину) покупок;

– подбирать товар под определённый бюджет (не превышающий по цене указанного значения);

– рассчитывать товарный чек с учётом скидок;

– показывать реализованные продажи (**а.** по выбранному товару; **б.** по магазину в целом) за указанный через календарь на месяц (*monthCalendar*) отчётный период.

**Вариант №2. Тема: «Архив дел уголовного розыска»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники архива уголовного розыска. В архиве лежат Дела за разные годы.

Реализовать запросы *LINQ*:

– выдавать список Дел по фамилии, имени, отчеству (реализация допускает объединение этой информации в Ф.И.О.) осуждённого;

– выдавать список Дел, которые вёл указанный следователь (реализация допускает объединение этой информации в Ф.И.О.);

– находить Дела по их содержанию (совпадение по отдельным ключевым словам или ключевым словосочетаниям);

– выдавать список Дел по указанной статье преступления (по характеру преступления).

**Вариант №3. Тема: «Кафедра Университета»**

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.

**Вариант №4. Тема: «Больница»**

Описание предметной области для подготовки данных. Пользователями являются медицинские работники, пациенты и посетители больницы (навещающие). Необходима информация о медицинских работниках больницы, лечащихся у них пациентах и размещении пациентов по палатам.

Реализовать запросы *LINQ*:

– выдавать сведения о медицинских работниках;

– выдавать сведения о медицинских работниках, наблюдающих за определённым пациентом;

– выдавать сведения о пациентах;

– выдавать сведения о пациентах, наблюдаемых у определённого медицинского работника;

– выдавать размещение пациентов по палатам.

**Вариант №5. Тема «Автосервис»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники автосервиса. Автосервис предлагает различные услуги по ремонту, покраске, помывке, тюнингу и прочим работам по восстановлению и модификации автомобилей. В автосервисе работают сотрудники, специализирующиеся на определённых видах работ. Каждый специалист может принять к работе и обслужить несколько автомобилей. Каждый автомобиль, находящийся в сервисе, может содержать несколько проблем/неполадок. Один и тот же клиент может владеть несколькими автомобилями. Для постоянных клиентов предусмотрена система скидок.

Реализовать запросы *LINQ*:

– выдавать список услуг, предлагаемых автосервисом с указанием цен на соответствующие услуги;

– выдавать список автомобилей, находящихся в автосервисе;

– выдавать информацию об указанном автомобиле (выводить перечень оказываемых услуг);

– выдавать информацию о работе, проделанной указанным мастером за отчётный период времени (день, месяц, квартал, год), определяемых через календарь на месяц (*monthCalendar*);

– рассчитывать общую стоимость услуг для клиентов.

**Вариант №6. Тема «Библиотека»**

Описание предметной области для подготовки данных. Пользователями являются читатели и сотрудники частной библиотеки. Библиотека содержит книги разных авторов, изданий, тематик. Необходимо вести поиск выбранной книги на полке.

Реализовать запросы *LINQ*:

– выдавать список книг по названию;

– выдавать список трудов указанного автора (учитывать труды, выполненные в соавторстве);

– выдавать список книг по указанной тематике;

– выдавать список книг указанного издательства;

– выдавать местонахождение указанной книги.

**Вариант №7. Тема: «Детский сад»**

Описание предметной области для подготовки данных. Пользователями являются администраторы детского сада. Детский сад посещают дети от 2-х до 7-ми лет. Дети, которым менее 3-х лет, в сентябре посещают ясли. В каждой группе работает пара воспитателей (один – утром, другой – вечером), а также нянечка. Детей в каждой группе не может быть больше 15-ти человек. С детьми проводятся занятия музыкой, физкультурой и рисованием в отдельных комнатах по два раза в неделю. С детьми обязательно гуляют два раза в день.

Реализовать запросы *LINQ*:

– показывать список детей указанной группы;

– показывать список детей указанного возраста;

– показывать занятость указанного воспитателя;

– показывать занятость указанной группы в указанный день недели (выбирать в выпадающем списке);

– показывать процентное отношение мальчиков и девочек в указанной группе;

– находить название (и/или №) группы по Ф.И.О. ребёнка (допускается поиск частичного соответствия).

**Вариант №8. Тема: «Гостиница»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники гостиницы и клиенты-постояльцы. В гостинице имеются номера, которые различаются по количеству мест, количеству комнат, удобствам и т.д. От всех этих параметров и их сочетания зависит цена гостиничного номера. Посетители могут занимать номера, бронировать их заранее. Постоянным посетителям и детям до 12 лет с родителями предоставляется скидка.

Реализовать запросы *LINQ*:

– предоставлять (и бронировать) посетителям номер в гостинице в зависимости от их потребностей (количества мест для размещения группы гостей и т.д.);

– показывать количество свободных (занятых) мест по категории номеров указанного типа;

– показывать список постоянных посетителей и предоставляемую им скидку;

– показывать информацию по указанному номеру (количество мест, комнат, наличие холодильника, телевизора, бара и т.д.).

**Вариант №9. Тема: «Аптека»**

Описание предметной области для подготовки данных. Пользователями являются посетители и сотрудники аптеки. В аптеку города поступает ассортимент лекарств со склада каждые 7 дней. Аптека предлагает услуги по продаже лекарств и их бронированию. Срок бронирования лекарств – 3 дня. В справочной системе аптеки можно получить информацию о лекарствах, находящихся в самой аптеке: название, форма выпуска, срок годности, аннотация, цена, изготовитель.

Реализовать запросы *LINQ*:

– выдавать сведения о лекарствах;

– предоставлять покупателям возможность бронирования лекарств, сроком на 3 дня;

– выдавать информацию о поступлении лекарства в аптеку, исходя из ассортимента на складе;

– выдавать информацию о продажах указанного лекарства за неделю / месяц / год – период определяется через календарь на месяц (*monthCalendar*);

– выполнять поиск лекарства по названию, форме выпуска, изготовителю;

– выдавать список лекарств, применяемых для лечения указанной болезни (лёгких недугов).

**Вариант №10. Тема: «Университет»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники деканата. В Университете проходят обучение студенты, распределённые по группам. По каждой специальности обучается несколько групп. Каждая кафедра ведёт занятия по нескольким дисциплинам. Занятия для студентов проводят преподаватели с разных кафедр. Лабораторные работы обязательно ведут два преподавателя.

Реализовать запросы *LINQ*:

– выдавать информацию о студенте по № зачётной книжки, по Ф.И.О.;

– выдавать список дисциплин, читаемых указанной кафедрой;

– выдавать список преподавателей, проводящих занятия в указанной группе;

– выдавать список групп, обучающихся на данной специальности.

**Вариант №11. Тема: «Продажа билетов в кино»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники кинотеатра (кассиры). В кинотеатре имеется несколько залов, в которых одновременно могут воспроизводиться разные фильмы. Каждый фильм могут показывать несколько раз в день по сеансам (в установленные часы). На фильм может накладываться ограничение по возрасту.

Реализовать запросы *LINQ*:

– показывать сеансы на указанное время, если имеются ограничения по возрасту, то дополнительно сообщать об этом;

– показывать список фильмов, идущих сегодня в кинотеатре, по залам;

– выдавать информацию о цене билета на указанный сеанс и указанный фильм;

– выдавать информацию о кратком содержании, задействованных актёрах и режиссёрах указанного фильма;

– продавать билет на выбранный сеанс указанного фильма;

– показывать продажи билетов по сеансам.

**Вариант №12. Тема: «Хлебопекарня»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники хлебопекарни. Хлебопекарня выпекает изделия на заказ. Продукты заказывают у поставщиков, выпечку отправляют в магазины города.

Реализовать запросы *LINQ*:

– показывать ассортимент товара и его цену;

– показывать сумму (количество) заказываемого продукта у указанного поставщика за отчётный период, определяемый через календарь на месяц (*monthCalendar*);

– показывать список товара (**а.** общую сумму заказа; **б.** сумму заказа указанного товара), заказанного указанным магазином;

– показывать количество произведённого товара (**а.** за отчётный период, определяемый через календарь на месяц (*monthCalendar*); **б.** по названию).

**Вариант №13. Тема: «Складской учёт»**

Описание предметной области для подготовки данных. Пользователями являются складские работники. Некоторая фирма содержит склад товаров. Эти товары фирма получает от производителя и расфасовывает их для магазинов. Фирма имеет возможность оптовой продажи ассортимента товаров. Для постоянных клиентов предусмотрена система скидок.

Реализовать запросы *LINQ*:

– выдавать ассортимент товара, находящегося на складе в настоящий момент;

– выдавать ассортимент товара, заказанного указанным магазином;

– показывать список продаж за указанный период времени, определяемый через календарь на месяц (*monthCalendar*);

– показывать список клиентов, имеющих скидку.

**Вариант №14. Тема: «Издательство»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники издательства. Издательство печатает книги, журналы, методические пособия и т.д. Каждый автор может подготавливать работы в соавторстве.

Реализовать запросы *LINQ*:

– выдавать список трудов указанного автора (учитывать труды, выполненные в соавторстве);

– выдавать список трудов по выбранному разделу (книги, журналы, методические пособия и т.д.);

– выдавать информацию об указанном авторе;

– показывать количество напечатанных работ за отчётный период (квартал, год), определяемый через календарь на месяц (*monthCalendar*).

**Вариант №15. Тема: «Строительная компания»**

Описание предметной области для подготовки данных. Пользователями являются клиенты компании. Компания строит дачные дома, бани и т.д. Имеются готовые проекты с фиксированной ценой, проекты на заказ стоят дороже на *х*%. На объект выезжает бригада, состоящая из рабочих и прораба.

Реализовать запросы *LINQ*:

– показывать список предоставляемых услуг и цену на них;

– показывать количество свободных (занятых) бригад;

– показывать работу, проводимую указанной бригадой;

– показывать список работ, проводимых у указанного заказчика;

– рассчитывать стоимость выполненных услуг для указанного заказчика.

**Вариант №16. Тема: «Контора адвоката»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники конторы. Контора оказывает юридические услуги (ведение дела в суде, консультация) по разным делам (гражданские, уголовные). Каждый адвокат специализируется в одной области (жилищные, семейные, убийства и т.д.). На каждое обращение в контору заводится Дело (№, содержание, адвокат, клиент).

Реализовать запросы *LINQ*:

– показывать список предоставляемых услуг и их цену;

– выдавать список клиентов, обращавшихся за указанной услугой;

– выдавать список свободных адвокатов по выбранной услуге;

– выдавать содержание Дела по его номеру.

**Вариант №17. Тема: «Спортивная команда по баскетболу»**

Описание предметной области для подготовки данных. Пользователем является администратор команды. Женский баскетбольный клуб, играющий в суперлиге в дивизионе А, состоит из 4 тренеров (главный и второй тренеры для основного и дублирующего состава), 24 игроков (12 – основой состав и 12 – дубль), врача, массажиста и администратора команды. На каждого члена команды имеется досье (Ф.И.О., возраст, рост, вес, звание (м/с, заслуженный тренер СССР или России и т.д.), амплуа). Результаты каждого матча фиксируются в таблицу индивидуально (фолы, подборы (свой/чужой щит), перехваты, потери, очки). Команда играет в чемпионате России и Евролиге.

Реализовать запросы *LINQ*:

– выводить счёт указанного матча для основного состава (дубля);

– показывать список команд чемпионата России (Евролиги);

– выдавать сведения об указанном игроке;

– выдавать список игроков для двух составов для указанного амплуа (центровой, разыгрывающий, защитник и др.);

– выдавать список игроков выбранной команды с тренерами;

– показывать результат указанного матча по указанному игроку (команде в целом);

– показывать самого результативного игрока в указанном матче.

**Вариант №18. Тема: «Поликлиника»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники регистрационного отдела поликлиники. Необходима информация о врачах, ведущих приём, расписании приёма, и пациентах, закреплённых за участками проживания.

Реализовать запросы *LINQ*:

– выдавать сводную информацию обо всех врачах поликлиники;

– выдавать сводную информацию о пациентах;

– выдавать информацию о записи пациента к врачу;

– выдавать информацию о приёме врачей на указанную дату (использовать *dateTimePicker*);

– выдавать информацию о пациентах, имеющих льготы на приобретение лекарств.

**Вариант №19. Тема: «Туристическое бюро»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники фирмы. Фирма оказывает услуги в туристическом бизнесе. Клиентам предлагаются путёвки в разные страны, города. Путёвки отличаются содержанием программы (отдых, экскурсии, туризм и т.д.), имеется возможность выбора путёвки по цене (в зависимости от места проживания, всё включено и т.д.). Для постоянных клиентов и детей до 12 лет имеются скидки.

Реализовать запросы *LINQ*:

– выдавать список стран;

– выдавать список городов;

– покупать путёвку в выбранное место с расчётом её стоимости;

– показывать весь ассортимент путёвок в указанное место;

– выбирать путёвку по содержанию, по цене и т.д.;

– показывать список самых популярных путёвок (по месту пребывания, по содержанию, в целом).

**Вариант №20. Тема: «Фильмотека»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники и посетители магазина. Магазин продаёт различные виды фильмов (на различных носителях).

Реализовать запросы *LINQ*:

– найти фильм по названию (по жанру, исполнителям, режиссёру);

– показывать список возможных носителей для выбранного фильма;

– показывать список фильмов по жанрам;

– показывать информацию по выбранному фильму (жанр, исполнители, режиссёр, краткое содержание);

– показывать список наиболее продаваемых фильмов;

– показывать количество проданных фильмов за отчётный период (сумму продаж).

**Вариант №21. Тема: «Магазин обуви»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники магазина. В магазине имеется несколько отделов обуви (мужская, женская, детская, спортивная и т.д.). Обувь имеет размер, цвет, фасон, цену, разделяется на зимнюю, летнюю, весна-осень и т.д., нумеруется артикулом. Для постоянных клиентов предусмотрена система скидок в виде карты лояльности.

Реализовать запросы *LINQ*:

– выдавать информацию о количество пар по заданному артикулу, размеру, цвету и т.д.

– выдавать ассортимент обуви по отделам, сезону;

– выдавать информацию об указанной обуви (цена, страна изготовитель);

– продавать выбранный товар;

– показывать продажи за выбранный период времени, определяемый через календарь на месяц (*monthCalendar*);

– показывать держателей карт лояльности.

**Вариант №22. Тема: «Транспортная/логистическая компания»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники компании. Компания предоставляет услуги по доставке грузов воздушным, речным, морским путём, по железной дороге или автомобилем по различным маршрутам.

Реализовать запросы *LINQ*:

– выдавать список маршрутов, обслуживаемых компанией и цену на них;

– выдавать список транспортных единиц, занятых на указанном маршруте;

– выдавать список вариантов проезда по указанному маршруту (морской и т.д.);

– рассчитывать стоимость услуг, оказанных по перевозке указанного груза. Постоянным клиентам предоставляется скидка.

– показывать грузооборот по указанному маршруту (в процентном соотношении).

**Вариант №23. Тема: «Магазин детских товаров»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники магазина. Магазин продаёт детские товары разного вида: одежда, обувь, питание, игрушки, письменные принадлежности, предметы по уходу за телом и т.д. По каждому виду имеется богатый ассортимент товаров. Постоянным клиентам и покупателям, чья сумма покупки превысила *R* руб., предоставляется система скидок.

Реализовать запросы *LINQ*:

– находить товар по его названию, показывать отдел, где его можно приобрести;

– показывать ассортимент выбранного отдела;

– показывать товарооборот по выбранным отделам (сумму продаж);

– рассчитывать стоимость покупки, учитывая скидки;

– показывать наличие выбранного размера (цвета, фасона) для обуви и одежды.

**Вариант №24. Тема: «Компания по услугам связи»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники и абоненты компании услуг связи. Каждый абонент обслуживается по выбранному тарифу. Каждый тариф имеет свой набор услуг и стоимость за единицу предоставляемой услуги.

Реализовать запросы *LINQ*:

– показывать список услуг;

– показывать список тарифов и цену за единицу услуги;

– показывать список абонентов по указанному тарифу;

– показывать список самых распространённых тарифов;

– показывать процентное распределение тарифов среди абонентов.

**Вариант №25. Тема: «Отдел кадров»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники отдела кадров. В фирме работают сотрудники на разных должностях и на разных ставках. У сотрудников могут быть дети, сотрудники могут быть пенсионерами или иметь инвалидность. Сотрудникам предоставляется очередной отпуск. Некоторые сотрудники могут быть в отпуске по уходу за ребёнком.

Реализовать запросы *LINQ*:

– показывать список сотрудников фирмы по Ф.И.О., по должности, ставке, с детьми / без детей, пенсионеров, находящихся в очередном отпуске, находящихся в отпуске по уходу за ребёнком;

– выдавать информацию о должностях сотрудников;

– выдавать информацию о предыдущих местах работы сотрудников;

– находить сотрудников по Ф.И.О.

**Вариант №26. Тема: «Школа»**

Описание предметной области для подготовки данных. Пользователями являются представители администрации школы. В школе учатся дети, распределённые по классам. У каждого класса свой классный руководитель. Каждый учитель ведёт своё направление (предмет). Вести учитель может максимум 6 уроков в день. По выбранному направлению (предмету) – обязательно только один учитель.

Реализовать запросы *LINQ*:

– показывать список учащихся указанного класса;

– показывать классного руководителя указанного класса;

– показывать занятость указанного учителя;

– показывать успеваемость указанного ученика;

– показывать список учеников, учащихся без троек.

**Вариант №27. Тема: «Кондитерская фабрика»**

Описание предметной области для подготовки данных. Пользователем является администрация фабрики. Фабрика изготавливает кондитерские изделия. Для изготовления товара требуются продукты, которые фабрика заказывает у поставщиков. Готовые товары расфасовываются для магазинов.

Реализовать запросы *LINQ*:

– показывать список магазинов, заказывающих указанный товар;

– показывать список продуктов, заказываемых у указанного поставщика;

– показывать ассортимент указанного товара и цену;

– выбирать наиболее популярный вид указанного товара;

– показывать стоимость произведённого товара за отчётный период времени, определяемый через календарь на месяц (*monthCalendar*).

**Вариант №28. Тема: «Компания по продаже недвижимости»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники и клиенты фирмы. Компания покупает и продаёт дома, а также квартиры. С каждой сделки компания получает проценты. Клиенты так же покупают и продают свои дома и квартиры.

Реализовать запросы *LINQ*:

– показывать все предложения на указанный вид недвижимости;

– показывать спрос на указанный вид недвижимости;

– показывать сведения о клиенте по его № паспорта (или Ф.И.О.);

– находить спрос (предложение) по выбранной цене, расположении, жилплощади;

– показывать прибыль компании (без учёта налогов и т.д.) за отчётный период, определяемый через календарь на месяц (*monthCalendar*);

– показывать список самых популярных спросов или предложений.

**Вариант №29.Тема: «Овощной магазин»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники магазина. Магазин получает товар со склада и расфасовывает его для продажи. Некоторые овощи могут быть испорчены.

Реализовать запросы *LINQ*:

– показывать ассортимент товара на выбранный день;

– показывать приход-расход выбранного товара;

– показывать прибыль магазина (без учёта налогов и т.д.) по указанному товару (за отчётный период времени);

– показывать в процентном отношении количество хорошего и испорченного товара для выбранного товара;

– показывать количество проданных единиц указанного товара за отчётный период, определяемый через календарь на месяц (*monthCalendar*).

**Вариант №30. Тема: «Мебельный магазин»**

Описание предметной области для подготовки данных. Пользователями являются сотрудники магазина. Магазин продаёт товары, изготовленные на фабриках, берёт заказы на изготовление мебели по индивидуальному проекту при увеличении стоимости на некоторый нефиксированный процент.

Реализовать запросы *LINQ*:

– показывать ассортимент предлагаемой мебели и цену товара;

– показывать количество проданной мебели выбранного образца за отчётный период времени, определяемый через календарь на месяц (*monthCalendar*);

– показывать список сделанных заказов за отчётный период времени, определяемый через календарь на месяц (*monthCalendar*);

– рассчитывать стоимость заказа клиента по индивидуальному проекту.

**Вариант №31. Тема: «Магазин музыкальных произведений»**

Описание предметной области для подготовки данных. Пользователями являются клиенты магазина. Покупатели самостоятельно выбирают и находят понравившийся им товар.

Реализовать запросы *LINQ*:

– показать все произведения указанного композитора (исполнителя), имеющиеся в магазине;

– показать местоположение выбранного произведения;

– показать список носителей для выбранного произведения;

– показать список произведений по жанру исполнения;

– находить произведение по названию, году выпуска, альбому и т.д.

– показывать список произведений указанного композитора (исполнителя) по выбранным годам творчества, определяемым через календарь на месяц (*monthCalendar*).