ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ АСТРАХАНСКОЙ ОБЛАСТИ
ВЫСШЕГО ОБРАЗОВАНИЯ
«АСТРАХАНСКИЙ ГОСУДАРСТВЕННЫЙ
АРХИТЕКТУРНО-
СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра «САПРиМ»

ПЛАТФОРМА .NET

Методические указания по выполнению лабораторных работ для студентов направления подготовки
09.04.02 «Информационные системы и технологии»
Программа «Искусственный интеллект в проектировании и производстве»
Направленность (профиль)
«Искусственный интеллект в проектировании городской среды»
очной формы обучения

Астрахань 2021

Составитель: к.т.н., доцент _____ /О.И. Евдошенко
(занимаемая должность, учёное степень и учёное звание) И.О.Ф.

Рецензент: д.т.н., профессор _____ /Т.В. Хошенко
(занимаемая должность, учёная степень и учёное звание) (подпись) И. О. Ф.

Методические указания к выполнению практических работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» очной и заочной форм обучения рассмотрены и одобрены на заседании кафедры «Систем автоматизированного проектирования и моделирования» ГАОУ АО ВО «АГАСУ»

Протокол №___ от_____2021

Зав.кафедрой /Евдошенко О.И.

Согласовано с УМУ ГАОУ АО ВО «АГАСУ»
_____2021

Специалист УМУ _____ / _____
подпись И.О.Ф.

Методические указания к выполнению практических работ для студентов направление подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды» утверждены и рекомендованы к публикации на заседании МКН подготовки «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды».

Протокол № _ от «_____»

Председатель МКН «Информационные системы и технологии» направленность (профиль) «Информационные системы и технологии в строительстве и архитектуре»

Зав.кафедрой,
доцент, к.т.н. /О.И.Евдошенко

Оглавление

Введение

Лабораторный практикум предназначен для практической поддержки курса «Платформа .NET» для студентов направления подготовки 09.04.02 «Информационные системы и технологии» направленность (профиль) «Искусственный интеллект в проектировании городской среды».

Основная цель лабораторного практикума: формирование у студентов системы знаний и практических навыков в области современной разработки приложений с использованием технологий программирования на базе платформы Microsoft .NET.

Учебно-методическое пособие представляет собой сборник лабораторных работ, выполнение которых должно обеспечить формирование устойчивых умений и навыков:

- использования современных технологий программирования на платформе Microsoft .NET;
- реализации алгоритма решения задачи с использованием языка программирования C#;
- анализа технической литературы;
- разработки отдельных компонентов системы и возможности интеграции их в одну систему;
- разработки безопасных консольных, веб-приложений и приложений с графическим интерфейсом, используя различные технологии платформы Microsoft .NET.

Ко всем программным средствам, рассматриваемым в лабораторных работах, приведены краткие инструкции, которые могут служить справочным материалом для самостоятельной работы. Этой же цели служит список литературы, приведенный в практикуме.

РАЗДЕЛ 1. ТЕХНОЛОГИЯ LINQ TO OBJECT

Лабораторная работа №1. Создание LINQ-запросов в C#. Поэлементные операции, агрегирование и генерирование последовательностей, теоретико-множественные операции (1 ч)

Запрос является выражением, которое извлекает данные из источника данных. Запросы, как правило, выражается в специализированном языке запросов. Различные языки были разработаны для различных типов источников данных, например, SQL для реляционных баз данных и XQuery для XML. Поэтому разработчики были вынуждены изучать новый язык запросов для каждого типа источника данных или формата данных. LINQ упрощает эту ситуацию, предлагая согласованную модель для работы с данными через различные виды источников и форматов данных. Вы можете использовать те же основные закономерности кодирования для запроса и преобразования данных в XML документов, баз данных SQL, наборах данных ADO.NET, .NET коллекций, и любой другой формат, для которого поставщик LINQ доступен.

Все операции LINQ запросов состоят из трех различных действий:

1. Получить источник данных.
2. Создать запрос.
3. Выполнить запрос.

Следующий пример показывает, как три части операции запроса выражаются в исходном коде. Пример использует целочисленный массив в качестве источника данных для удобства, однако, те же принципы применимы к другим источникам данных.

```
class IntroToLINQ
{
    static void Main()
    {
        // The Three Parts of a LINQ Query:
        //  1. Data source.
        int[] numbers = new int[7] { 0, 1, 2, 3, 4, 5, 6 };

        // 2. Query creation.
        // numQuery is an IEnumerable<int>
        var numQuery =
            from num in numbers
            where (num % 2) == 0
            select num;

        // 3. Query execution.
        foreach (int num in numQuery)
        {
            Console.Write("{0,1} ", num);
        }
    }
}
```

Если необходимо указать несколько условий, используйте логические операторы || *(ИЛИ)*, *&& (И)*.

Задания для самостоятельного выполнения

Поэлементные операции, агрегирование и генерирование последовательностей

1. Дана целочисленная последовательность, содержащая как положительные, так и отрицательные числа. Вывести ее первый положительный элемент и последний отрицательный элемент.

2. Даны цифра D (однозначное целое число) и целочисленная последовательность A. Вывести первый положительный элемент последовательности A, оканчивающийся цифрой D. Если требуемых элементов в последовательности A нет, то вывести 0.

3. Даны целое число *L* (> 0) и строковая последовательность *A*. Вывести последнюю строку из *A*, начинающуюся с цифры и имеющую длину *L*. Если требуемых строк в последовательности *A* нет, то вывести строку «Not found».

Указание. Для обработки ситуации, связанной с отсутствием требуемых строк, использовать операцию ??.

4. Даны символ C и строковая последовательность A. Если A содержит единственный элемент, оканчивающийся символом C, то вывести этот элемент; если требуемых строк в A нет, то вывести пустую строку; если требуемых строк больше одной, то вывести строку «Error».

5. Даны символ *C* и строковая последовательность *A*. Найти количество элементов *A*, которые содержат более одного символа, и при этом начинаются и оканчиваются символом *C*.

6. Дана целочисленная последовательность. Найти количество ее положительных двузначных элементов, а также их среднее арифметическое (как вещественное число). Если требуемые элементы отсутствуют, то дважды вывести 0 (первый раз как целое, второй – как вещественное).

7. Даны целое число *L* (> 0) и строковая последовательность *A*. Строки последовательности *A* содержат только заглавные буквы латинского алфавита. Среди всех строк из *A*, имеющих длину *L*, найти наибольшую (в смысле лексикографического порядка). Вывести эту строку или пустую строку, если последовательность не содержит строк длины *L*.

8. Дана строковая последовательность. Найти сумму длин всех строк, входящих в данную последовательность.

9. Даны целое число *D* и целочисленная последовательность *A*. Начиная с первого элемента *A*, большего *D*, извлечь из *A* все нечетные положительные числа, поменяв порядок извлеченных чисел на обратный.

Лабораторная работа №2. Создание LINQ-запросов в C#. Поэлементные операции, агрегирование и генерирование последовательностей, теоретико-множественные операции (1 ч)

Изучаемые методы LINQ: First, FirstOrDefault, Last, LastOrDefault, Single, SingleOrDefault (поэлементные операции); Count, Sum, Average, Max, Min, Aggregate (агрегирование); Range (генерирование последовательностей); Union, Intersect, Except (теоретико-множественные операции); Distinct, Reverse (удаление повторяющихся элементов и инвертирование); Where, TakeWhile, SkipWhile, Take, Skip (фильтрация).

Задания для самостоятельного выполнения

Поэлементные операции, агрегирование и генерирование последовательностей

1. Дана последовательность непустых строк. Используя метод Aggregate, получить строку, состоящую из начальных символов всех строк исходной последовательности.

2. Дана целочисленная последовательность. Используя метод Aggregate, найти произведение последних цифр всех элементов последовательности. Чтобы избежать целочисленного переполнения, при вычислении произведения использовать вещественный числовой тип.

3. Даны целые числа A и B (A < B). Используя методы Range и Average, найти среднее арифметическое квадратов всех целых чисел от A до B включительно: (A2 + (A + 1)2 + … + B2)/(B – A + 1) (как вещественное число).

Фильтрация, сортировка, теоретико-множественные операции

4. Дано целое число *K* (> 0) и строковая последовательность *A*. Из элементов *A*, предшествующих элементу с порядковым номером *K*, извлечь те строки, которые имеют нечетную длину и начинаются с заглавной латинской буквы, изменив порядок следования извлеченных строк на обратный.

5. Даны целые числа *D* и *K* (*K* > 0) и целочисленная последовательность *A*. Найти теоретико-множественное объединение двух фрагментов *A*: первый содержит все элементы до первого элемента, большего *D* (не включая его), а второй – все элементы, начиная с элемента с порядковым номером *K*. Полученную последовательность (не содержащую одинаковых элементов) отсортировать по убыванию.

6. Даны целое число *K* (> 0) и целочисленная последовательность *A*. Найти теоретико-множественную разность двух фрагментов *A*: первый содержит все четные числа, а второй – все числа с порядковыми номерами, большими *K*. В полученной последовательности (не содержащей одинаковых элементов) поменять порядок элементов на обратный.

7. Даны целое число K (> 0) и последовательность непустых строк A. Строки последовательности содержат только цифры и заглавные буквы латинского алфавита. Найти

теоретико-множественное пересечение двух фрагментов A: первый содержит 3K начальных элементов, а второй – все элементы, расположенные после последнего элемента, оканчивающегося цифрой. Полученную последовательность (не содержащую одинаковых элементов) отсортировать по возрастанию длин строк, а строки одинаковой длины – в лексикографическом порядке по возрастанию.

Лабораторная работа №3. Проецирование, группировка и объединение элементов коллекции с помощью LINQ-запросов в C# (1 ч)

Изучаемые запросы LINQ: Select, SelectMany (проецирование), Concat (сцепление); Join, GroupJoin (объединение); DefaultIfEmpty (замена пустой последовательности на одноэлементную); GroupBy (группировка).

Задачи для самостоятельного выполнения

Проецирование

1. Дано целое число *K* (> 0) и строковая последовательность *A*. Каждый элемент последовательности представляет собой несколько слов из заглавных латинских букв, разделенных символами «.» (точка). Получить последовательность строк, содержащую все слова длины *K* из элементов *A* в лексикографическом порядке по возрастанию (слова могут повторяться).

2. Дана строковая последовательность A. Строки последовательности содержат только заглавные буквы латинского алфавита. Получить новую последовательность строк, элементы которой определяются по соответствующим элементам A следующим образом: пустые строки в новую последовательность не включаются, а к непустым приписывается порядковый номер данной строки в исходной последовательности (например, если пятый элемент A имеет вид «ABC», то в полученной последовательности он будет иметь вид «ABC5»). При нумерации должны учитываться и пустые строки последовательности A. Отсортировать полученную последовательность в лексикографическом порядке по возрастанию.

Объединение и группировка

3. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны, имеют ненулевую длину и содержат только цифры и заглавные буквы латинского алфавита. Получить последовательность всевозможных комбинаций вида «E_A=E_B», где E_A – некоторый элемент из *A*, E_B – некоторый элемент из *B*, причем оба элемента оканчиваются цифрой (например, «AF3=D78»). Упорядочить полученную последовательность в лексикографическом порядке по возрастанию элементов *EA*, а при одинаковых элементах *EA* – в лексикографическом порядке по убыванию элементов *EB*.

Указание. Для перебора комбинаций использовать методы SelectMany и Select.

4. Дана последовательность непустых строк. Среди всех строк, начинающихся с одного и того же символа, выбрать наиболее длинную. Если таких строк несколько, то выбрать первую по порядку их следования в исходной последовательности. Полученную последовательность строк упорядочить по возрастанию кодов их начальных символов.

5. Дана целочисленная последовательность A. Сгруппировать элементы

последовательности A, оканчивающиеся одной и той же цифрой, и на основе этой группировки получить последовательность строк вида «D:S», где D — ключ группировки
(т. е. некоторая цифра, которой оканчивается хотя бы одно из чисел последовательности A), а S — сумма всех чисел из A, которые начинаются цифрой D. Полученную последовательность упорядочить по возрастанию ключей.

Указание. Использовать метод GroupBy.

1. Даны последовательности положительных целых чисел *A* и *B*; все числа в последовательности *A* различны. Получить последовательность строк вида «*S*:*E*», где *S* обозначает среднее арифметическое тех чисел из *B*, которые оканчиваются на ту же цифру, что и число *E* – один из элементов последовательности *A* (например, «74:23»); если для числа *E* не найдено ни одного подходящего числа из последовательности *B*, то в качестве *S* указать 0. Расположить элементы полученной последовательности по возрастанию значений найденных сумм, а при равных суммах – по убыванию значений элементов *A*.

2. Даны строковые последовательности *A* и *B*; все строки в каждой последовательности различны и имеют ненулевую длину. Получить последовательность строк вида «*E*:*N*», где *E* обозначает один из элементов последовательности *A*, а *N* — количество элементов из *B*, начинающихся с того же символа, что и элемент *E* (например, «abc:4»); количество *N* может быть равно 0. Порядок элементов полученной последовательности должен определяться исходным порядком элементов последовательности *A*.

Указание. Использовать метод GroupJoin.

Лабораторная работа №4. Создание LINQ-запросов в C# для выборки данных из коллекции. Технология LINQ to Object. (1 ч)

Запустим Visual Studio, закажем новый проект шаблона Windows Forms Application C#. Первая задача заключается в том, чтобы создать список сотрудников предприятия и из этого списка выбрать некоторых сотрудников по какому-либо признаку, например, по стажу. При создании списка объявлен класс Сотрудник, который содержит три свойства: ФИО, Возраст и Стаж. В начале решения заполняем список сотрудников, а затем строим LINQ-запрос для заполнения списка сотрудников со стажем больше 10 лет. Этот список выводим в текстовое поле textBox1, используя цикл foreach.

Затем из панели элементов перенесем в форму текстовое поле. Далее через щелчок правой кнопкой мыши перейдем к вкладке программного кода.

Пошагово пропишем код:

1. Создать класс Сотрудник с полями: ФИО, Возраст, Стаж работы
2. Объявить и заполнить коллекцию (список) Сотрудники

```
List<Сотрудник> Сотрудники = new List<Сотрудник>

                    {

new Сотрудник {ФИО="Карапузова Ирина", Возраст=27, Стаж=10}

              . . .};
```

3. Создать LINQ-запрос для вывода ФИО и возраста сотрудников с стажем больше 10 лет

```
var Результат =  from Сотрудник in Сотрудники
                 where условие (Сотрудник.Стаж>10)
                 orderby имя_поля //сортировка по полю
                 select Сотрудник;
```

4. Вывести результат запроса в TextBox, используя цикл foreach

```
        foreach (var x in Результат)
 textBox1.Text +=
      string.Format("{0} -   возраст " +"- {1}\r\n", x.ФИО, x.Возраст);
```

Решим еще одну задачу, когда в списке необходимо использовать еще вложенный (внутренний) список. Требуется создать список студентов факультета, содержащий фамилию студента и список полученных имтекущих оценок, т. е. список оценок должен быть "вложен" в список студентов. Из списка студентов необходимо выбрать тех, кто имеет в списке своих оценок хотя бы одну двойку. Для решения этой задачи вначале объявляем новый класс *Студент***,** который имеет в качестве свойств класса фамилию студента и список (тип **List)** оценок. Затем строим **LINQ-запрос**, где, поскольку нам необходимо искать элемент списка внутри "вложенного" списка, мы используем предложение **From** два раза.

1. Объявляем класс Студент, содержащий фамилию студента и список полученных им

оценок. Для хранения оценок используем список:

```
List<int> Оценки
```

2. Создадим список Студенты и заполним его значениями

```
List<Студент> Студенты = new List<Студент>
            {
new Студент {Фамилия="Иванов", Оценки= new List<int> {5, 4, 4, 5}}
. . . . }
```

3. Создадим LINQ-запрос для выбора студентов, имеющих двойки. Для доступа к внутреннему списку предложение from используется еще раз

```
var СписокДвоечников = from Студент in Студенты
                       from Оценка in Студент.Оценки
                       where Оценка <= 2
                       orderby Студент.Фамилия
                       select new { Студент.Фамилия, Оценка };
```

4. Вывести полученный результат в текстовое поле

Задачи для самостоятельного выполнения:

Внимание! Данные необходимо хранить в списках (коллекции).

1. В организации имеется 3 отдела. В каждом отделе имеется от 3 до 5 сотрудников. Используя группировку по отделу, вывести список сотрудников и средний оклад по каждому отделу. Определите долю суммы окладов всех сотрудников одного отдела в общей сумме окладов по всему предприятию.

2. Даны сведения о счетах клиентов в банке: ФИО клиента, номер счета (пример: 40562**810**45718745, 40562**840**45718745, 40562**978**45718745), сумма остатка. Вывести общую сумму одного клиента по всем счетам в указанной валюте. Валюта определяется по выделенным в образце цифрам (810 – рубли, 978 – евро, 840 – доллары) в номере счета. Пользователь должен иметь возможность выбора валюты и ввода фамилии клиента.

Руб. - 195000

Eur - 1000

USD – 500

3. Дано целое число *K* – код одного из клиентов фитнес-центра. Исходная последовательность содержит сведения о клиентах этого фитнес-центра. Каждый элемент последовательности включает следующие целочисленные поля:

<Код клиента> <Год> <Номер месяца>

<Продолжительность занятий (в часах)>

Для каждого года, в котором клиент с кодом K посещал центр, определить месяц, в котором продолжительность занятий данного клиента была наименьшей для данного года (если таких месяцев несколько, то выбирать первый из этих месяцев в исходном наборе;

месяцы с нулевой продолжительностью занятий не учитывать). Сведения о каждом годе выводить на новой строке в следующем порядке: наименьшая продолжительность занятий, год, номер месяца. Упорядочивать сведения по возрастанию продолжительности занятий, а при равной продолжительности – по возрастанию номера года. Если данные о клиенте с кодом K отсутствуют, то записать в результирующий файл строку «Нет данных».

Указание. Для отбора данных, связанных с клиентом K, использовать метод Where. Затем выполнить группировку по полю «год» и для каждой полученной последовательности выбрать требуемый месяц с помощью сортировки по набору ключей «продолжительность занятий, номер месяца». Обработку особой ситуации, связанной с отсутствием требуемых данных, выполнять с использованием метода DefaultIfEmpty с параметром «Нет данных».

4. Исходная последовательность содержит сведения об абитуриентах. Каждый элемент последовательности включает следующие поля:

<Номер школы> <Год поступления> <Фамилия>

Для каждого года, присутствующего в исходных данных, вывести число различных школ, которые окончили абитуриенты, поступившие в этом году (вначале указывать число школ, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по возрастанию числа школ, а для совпадающих чисел — по возрастанию номера года.

5. Из последовательности (см. п.4) определить, в какие годы общее число абитуриентов для всех школ было наибольшим и наименьшим, и вывести это число, а также годы, в которые оно было достигнуто (годы упорядочивать по возрастанию, каждое число выводить на новой строке).

6. Исходная последовательность содержит сведения о задолжниках по оплате коммунальных услуг, живущих в 144-квартирном 9-этажном доме. Каждый элемент последовательности включает следующие поля:

<Задолженность> <Фамилия> <Номер квартиры>

Задолженность указывается в виде дробного числа (целая часть — рубли, дробная часть — копейки). В каждом подъезде на каждом этаже располагаются по 4 квартиры. Длякаждого из 4 подъездов дома найти трех жильцов с наибольшей задолженностью и вывести сведения о них: задолженность (выводится с двумя дробными знаками), номер подъезда, номер квартиры, фамилия жильца. Считать, что в наборе исходных данных все задолженности имеют различные значения. Сведения о каждом задолжнике выводить на отдельной строке и упорядочивать по убыванию размера задолженности (номер подъезда при сортировке не учитывать). Если в каком-либо подъезде число задолжников меньше трех, то включить в полученный набор всех задолжников этого подъезда.

7. Даны два класса Student и Subject. Класс Student содержит сведения об оценках

учащихся по школьным предметам. Каждый элемент последовательности содержит данные об одной оценке и включает следующие поля:

<ФИО> <Класс> <Код предмета> <Оценка>

Класс Subject содержит список предметов, включающий следующие поля:

<Код предмета> <Название предмета>

Класс задается целым числом, оценка – целое число в диапазоне 2-5. Для каждого класса определить средний балл по каждому предмету. Полученные сведения выводить на отдельной строке, указывая класс, название предмета и средний балл. Данные расположить по возрастанию номера класса и по убыванию среднего балла.

8. Даны последовательности A и B, включающие следующие поля: *A: категория, артикул товара, страна производитель; B: артикул товара, цена, название магазина.*Для каждой категории товаров определить количество магазинов, предлагающих товары данной категории, а также количество стран, в которых произведены товары данной категории, представленные в магазинах (вначале выводится количество магазинов, затем название категории, затем количество стран). Если для некоторой категории не найдено ни одного товара, представленного в каком- либо магазине, то информация о данной категории не выводится. Сведения о каждой категории выводить на новой строке и упорядочивать по убыванию количества магазинов, а в случае одинакового количества — по названиям категорий в алфавитном порядке.

9. Даны последовательности A, B и C, включающие следующие поля: A: улица, код потребителя, год рождения; B: страна производителя, категория, артикул товара; C: артикул товара, код потребителя, название магазина. Для каждого года рождения изA определить страну, в которой было произведено максимальное количество товаров, приобретенных потребителями этого года рождения (вначале выводится год, затем название страны, затем максимальное количество покупок). Если для некоторой пары «год–страна» отсутствует информация о проданных товарах, то эта пара не обрабатывается (в частности, если потребители некоторого года рождения не сделали ни одной покупки, то информация об этом годе не выводится). Если для какого-либо года рождения имеется несколько стран с наибольшим числом приобретенных товаров, товыводятся данные о первой из таких стран (в алфавитном порядке). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию номера года.

Лабораторная работа №5. Создание методами пространства имен System.XML.Linq и поиск строк с помощью LINQ-запроса в XML-файлах (1 ч)

Кроме пространства имен **System.Xml**, содержащего классы для обработки XML-документов, в C# имеем пространство имен **System.Xml.Linq**, содержащее классы, которые позволяют легко и эффективно изменять документы XML, а также организовывать **LINQ-запросы**. В данном примере оформим сведения о студентах в XML- документ. Этот документ будет иметь интуитивно понятную структуру: фамилия студента, группа и мобильный телефоны. Создав такой XML- документ и получив соответствующий XML-файл, его очень удобно просмотреть в MS Excel в виде таблицы, содержащей три столбца: студент, группа и мобильный телефон.

Запустим среду **Visual Studio**, выберем проект шаблона **Console Application**, укажем любое имя. Затем на вкладке программного кода введем текст, следую шагам:

1. Объявим директиву

```
using System.Xml.Linq;
```

2. Создаем новый XML-документ

```
XDocument XMLдокумент = new XDocument(структура XML-документа: п.3+п.4)
```

3. Задаем имя корневого элемента

```
new XElement("Студенты",элементы и вложенные элементы: п.4)
```

4. Зададим вложенные элементы, имя элемента и его значение, а также атрибуты элемента, их имена и значения

```
new XElement("Студент", // - имя (Name) элемента
new XAttribute("Фамилия", "Олег"), // - атрибут элемента Строка new
XElement("Группа", "ИТ-31"), // - имя элемента и его значениеnew
XElement("Мобильный_телефон", "+7(902)120-20-20")),
```

5. Повторим код пункта 4 для каждого студента

6. Сохраняем и выводим на экран полученный документ

```
XMLдокумент.Save(@"C:\имя_файла.XML");
Console.WriteLine(XMLдокумент);
Console.ReadKey();
```

Чтобы понять текст программы, рассмотрим структуру полученного XML- файла, а для этого откроем этот файл с помощью браузера.

Здесь весь XML-документ вложен в так называемый корневой элемент между начальным тегом <*Студенты*> и конечным тегом </*Студенты*>. Элемент *Студент* вложены в корневой элемент. В соответствующей таблице MS Excel элементы *Студент* будут представлять строку в таблице. В свою очередь элемент *Студент* содержит в себе атрибут *Фамилия* и два вложенных в него элемента, имена (Name) которых — *Группа* и *Мобильный_телефон*. Именно поэтому в MS Excel отобразится таблица с тремя колонками

(один атрибут и два элемента): «Фамилия», «Группа» и «Мобильный_телефон».

Элемент может иметь один или несколько атрибутов (а может и не иметь, как, скажем, элемент *Мобильный_телефон*).

После запуска данной программы будет выведено на консоль содержимое XML-документа (без XML-объявления), а также будет создан XML-файл. Открыв этот файл с помощью MS Excel, получим таблицу телефонных контактов студентов.

Задание для самостоятельного выполнения (блок 1):

Создать 3 XML-файла. Первый файл должен содержать сведения о сотрудниках (**код сотрудника, полное ФИО, должность по штатному расписанию, серия паспорта, № паспорта, дата приема на работу, личный e-mail сотрудника, право подписи договоров, код отдела**), второй – об отделах (**код отдела, название отдела, код руководителя)**, третий – о начисления заработной платы (**код сотрудника, месяц (01-12), год, итого начислено)**.

Имеем базу данных, которую мы получили на каком-то этапе обработки в виде XML, и нам требуется "отфильтровать" записи в этой базе на предмет содержания в некотором поле определенной строки, например, отобразить список телефон студентов необходимой группы.

Для решения этой задачи запустим **Visual Studio** и выберем проект шаблона **Windows Forms Application,** укажем любое имя. Далее, попав в конструктор формы, из панели элементов **Toolbox** перетащим текстовое поле **TextBox** для вывода в него найденных строк из таблицы XML. В свойствах текстового поля разрешим ввод множества строк (а не одной), для этого свойство **Multiline** переведем в состояние true. Затем на вкладке программного кода введем текст, следую шагам:

1. Извлекаем информацию из XML-файла

```
var КорневойЭлемент = System.Xml.Linq.
XElement.Load("имя_файла.XML");
```

2. Используем LINQ-запрос для выборки данных

```
var Записи =
          from x in КорневойЭлемент.Elements("Студент")
          where (string)x.Element("Группа") == "ИТ-31"
          select x.Element("Мобильный_телефон").Value;
```

3. Для выборки записей по значению атрибута конструкцию where

необходимо заменить на:

```
where (string)x.Attribute("Фамилия").Value=="Иванов"
```

4. Вывести коллекцию записей.

Извлекаем корневой элемент из XML-документа. Затем организуем типовой, стандартный **LINQ-запрос**. Результат запроса попадает в коллекцию записей, которую выводим, используя оператор цикла **foreach**.

Задачи для самостоятельного выполнения (блок 2):

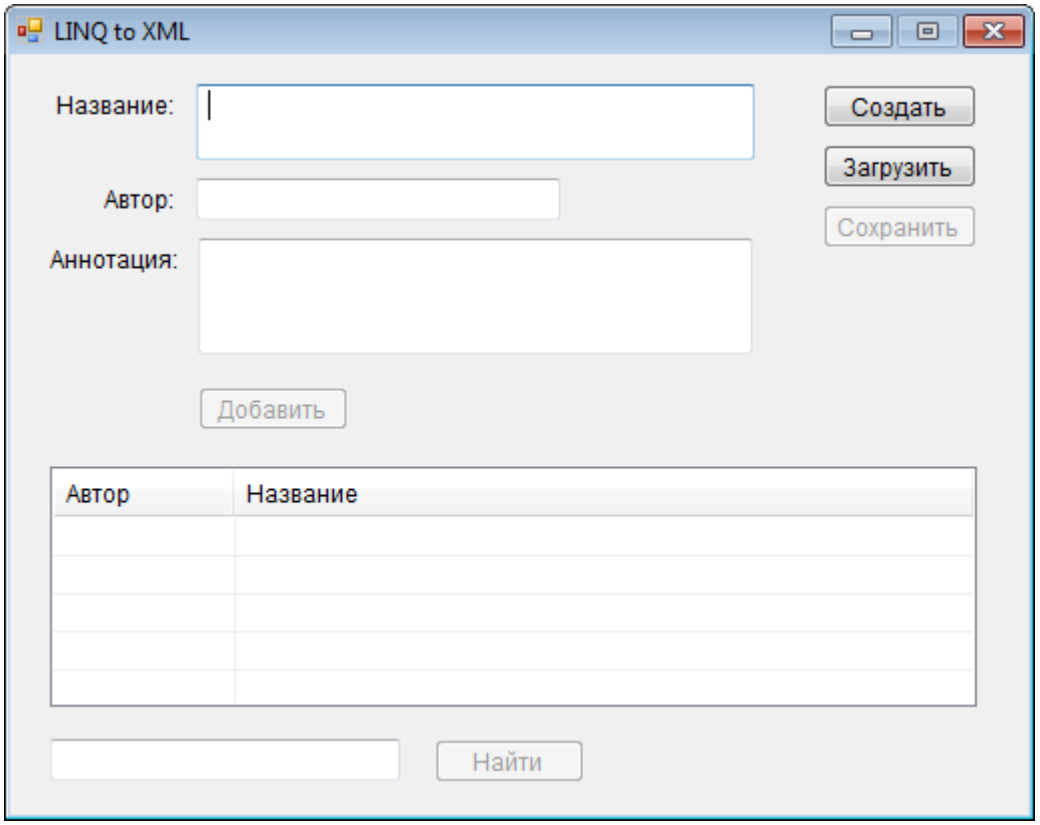
1. Организовать поиск сотрудников по серии и номеру паспорта.
2. Вывести сотрудников с указанием информации о количестве проработанныхлет, месяцев, дней в организации (например, 10 лет 9 месяцев 3 дня). Выделить красным цветом сотрудников, которые проработали более 25 лет.
3. Сгруппировать сотрудников по отделам. По каждому отделу вывести фамилии работающих сотрудников. Выделить тех, которые имеют право подписи договоров в каждом отделе.
4. Определить количество работающих сотрудников по каждому отделу и по каждой должности внутри отдела. Для каждого отдела определить долю работающих сотрудников из общего контингента.
5. Для каждого руководителя отдела вывести список работающих сотрудников.
6. Для сотрудника вывести сводный отчет по начислениям заработной платы каждого месяца в текущем году. Если в какой-то месяц начисления отсутствуют, вывести соответствующую информацию.
7. Определить минимальную и максимальную заработную плату в каждом отделе.

Лабораторная работа №6. Создание проекта на Visual C# для работы с XML-файлами (1 ч)

Данная программа показывает, как используя **LINQ-запрос** создать XML-файл, загрузить существующий документ и добавить в него информацию, выбрать нужную информацию и отобразить ее в поле компонента **ListView**.

Задание: *создать приложение, которое позволит выполнять добавление информации в XML-файл, её извлечение и поиск.*

Выходная форма приложения должна иметь примерно следующий вид:



Для начала работы необходимо:

1. Объявите в классе переменную

```
XDocument doc;
```

2. Надстройте компонент **ListView**

2.1 Добавьте две колонки: *Автор* и *Название*. Установите их ширину.

2.2. Установите видимость линий сетки между колонками и строками.

3. Для кнопок «Найти», «Добавить» и «Сохранить» установить **Enabled** в false.

Для кнопки **«Загрузить»:**

1. Загрузить данные из файла в переменную

```
doc = XDocument.Load("\\books.xml");
```

2. Активировать кнопки «Найти», «Добавить» и «Сохранить»

3. Объявить

```
IEnumerable<XElement> books = doc.Elements();
```

4. В цикле добавить содержимое XML-файла в **ListView**

```
foreach (XElement aBooks in books.Elements())
            {
     listView1.Items.Add(aBooks.Element("Author").Value);// первый узел- элемент
                    listView1.Items[listView1.Items.Count - 1].
     SubItems.Add(aBooks.Element("Title").Value);// остальные узлы - подэлементы
}
```

5. Деактивировать кнопку «Создать»

Для кнопки «Добавить»:

```
        doc.Element("корневой_элемент").Add(new XElement("book",
                      new XElement("Author", источник1),
                      new XElement("Title", источник2),
                      new XElement("Description", источник3)
                    )
);
```

источник – объект, из которого извлекаются данные для записи вфайл (текстовое поле, метка, список и.т.д.).

Для кнопки **«Создать»**:

1. Проверить существует ли уже файл

1.1. Создать объект класса FileInfo

```
FileInfo fi = new FileInfo("\\books.xml");
```

1.2. Для проверки использовать условие с методом **Exists**

1.3. Если файл уже существует, то сообщить пользователю и дать возможность выбора – заменить или нет.

2. Если файл не существует или пользователь выбрал команду заменить:

```
                doc =
                 new XDocument(
        //данный блок может повторяться несколько раз
                     new XElement("books", new
                         XElement("book",
                             new XElement("Author", "Иванов И.И."),
                             new XElement("Title", "Математика"), new
                             XElement("Description", "")
                          ),
         //
)
```

3. Сохранить в физическом файле

```
doc.Save("\\books.xml");
```

Для кнопки **«Найти»**

1. Очистить **ListView**

2. Создать LINQ-запрос для выбора названий книг одного автора

```
IEnumerable<XElement> query =
     from b in doc.Elements().Elements()
     where b.Elements("Author").Any(n =>
     n.Value.Contains(источник))
     select b;
```

3. Использовать цикл для формирования и вывода результата

```
foreach (XElement e1 in имя_LINQ_запрос) //1 цикл
{
    bool firstElement = true;
    foreach (XElement e2 in e1.Elements()) // 2 цикл
    {
    }

}
```

1. В теле второго цикла необходимо сделать проверку: если `firstElement == true,` тогда добавляем значение первого узла (элемента) из `e2.Value` и присваиваем `firstElement = false` иначе добавляем значение остальных узлов (подэлементов).

Задание для самостоятельного выполнения

Создать приложение, которое позволит выполнять добавление, извлечение и поиск информации из XML-файла. XML-файл должен описывать ФИО сотрудника, год рождения, домашний адрес, телефон, сведения о работе (название должности, дата начала работы, дата окончания работы) и заработной плате (год, месяц, итого).

Предусмотреть возможность редактирования и добавления новых данных о сотруднике, должности и заработной плате.

Приложение должно осуществлять поиск сотрудника по фамилии, выводить историю о трудовой деятельности и начислениях заработной платы сотрудника.

Приложение должно осуществлять вывод начислений заработной платы сотрудника за определённый год с указанием максимальной, минимальной и средней зарплаты.

ФИО сотрудника: Иванов Иван Иванович		
Год рождения: 25.06.1995		
Домашний адрес: г. Астрахань, ул. Победы, 40 кв. 25		
Телефон: 960456789		
Сведения о работе		
Инженер	01.01.2017	31.08.2017
Программист	01.09.2017	31.08.2019
…	…	…
Директор	01.09.2019	

Зарплата		
2017	01	12500
2017	10	16000
…		…
2019	09	50000

Структура XML-файла:

```
<Сотрудник>
    <ФИО></ФИО>
    <Год рождения></Год рождения>
    <Домашний адрес></Домашний адрес>
    <Телефон></Телефон>
    <Список_Работ>
        <Работа>
            <Название></Название>
            <Дата_начала></ Дата_начала>
            <Дата_окончания></ Дата_окончания>
        </Работа>
        <Работа>
            <Название></Название>
            <Дата_начала></ Дата_начала>
            <Дата_окончания></ Дата_окончания>
        </Работа>
    </Список_Работ>
    <Список_Зарплат>
        <Зарплата>
            <Год></Год>
            <Месяц></Месяц>
            <Размер></Размер>
        </Зарплата>
        <Зарплата>
            <Год></Год>
            <Месяц></Месяц>
            <Размер></Размер>
        </Зарплата>
    </Список_Зарплат>
</Сотрудник>
```

РАЗДЕЛ 2. РАБОТА С БАЗАМИ ДАННЫХ. ENTITY FRAMEWORK

Лабораторная работа №7. База данных, основанная на службах. (4 ч)

Задание для самостоятельного выполнения

1. Создать таблицу, содержащую следующую информацию:

<Код клиента> <Год> <Номер месяца>
<Продолжительность занятий (в часах)>

2. Сделайте запросы для выборки данных:

2.1. Определить год, в котором суммарная продолжительность занятий всех клиентов была наибольшей, и вывести этот год и наибольшую суммарную продолжительность. Если таких годов было несколько, то вывести наименьший из них.

2.2. Для каждого клиента, присутствующего в исходных данных, определить суммарную продолжительность занятий в течение всех лет (вначале выводить суммарную продолжительность, затем код клиента). Сведения о каждом клиенте выводить на новой строке и упорядочивать по убыванию суммарной продолжительности, а при их равенстве – по возрастанию кода клиента.

2.3. Для каждого года, в котором клиент с кодом K посещал центр, определить число месяцев, для которых продолжительность занятий данного клиента превосходила 15 часов (вначале выводить число месяцев, затем год). Если для некоторого года требуемые месяцы отсутствуют, то вывести для него 0. Сведения о каждом годе выводить на новой строке; данные упорядочивать по убыванию числа месяцев, а при равном числе месяцев – по возрастанию номера года. Если данные об указанном клиенте отсутствуют, то записать в результирующий файл строку «Нет данных».

Указание. Для отбора данных, связанных с клиентом K, использовать метод Where. Затем выполнить группировку по полю «год» и для каждой полученной последовательности выбрать требуемый месяц с помощью сортировки по набору ключей «продолжительность занятий, номер месяца». Обработку особой ситуации, связанной с отсутствием требуемых данных, выполнять с использованием метода DefaultIfEmpty с параметром «Нет данных».

2.4. Найти элемент последовательности с минимальной и максимальной продолжительностью занятий. Вывести эту продолжительность, а также соответствующие ей год и номер месяца (в указанном порядке на той же строке). Если имеется несколько элементов с минимальной продолжительностью, то вывести данные того из них, который является последним в исходной последовательности.

Указание. Для нахождения требуемого элемента следует использовать методы OrderByDescending и Last.

3. Создать таблицу, содержащую следующую информацию:

<Номер школы> <Год поступления> <Фамилия>

4. Сделайте запросы для выборки данных:

4.1. Найти годы, для которых число абитуриентов было не меньше среднего значения по всем годам (вначале указывать число абитуриентов для данного года, затем год). Сведения о каждом годе выводить на новой строке и упорядочивать по убыванию числа абитуриентов, а для совпадающих чисел — по возрастанию номера года.

4.2. Для каждой школы найти годы поступления абитуриентов из этой школы и вывести номер школы и найденные для нее годы (годы располагаются на той же строке, что и номер школы, и упорядочиваются по возрастанию). Сведения о каждой школе выводить на новой строке и упорядочивать по возрастанию номеров школ.

4.3. Для каждой пары «год–школа», присутствующей в исходных данных, найти число абитуриентов, относящихся к этому году и школе, и вывести год, номер школы и найденное число абитуриентов. Сведения о каждой паре «год–школа» выводить на новой строке и упорядочивать по убыванию года, а для совпадающих годов — по возрастанию номера школы.

5. Создать следующие таблицы:

Таблица 1: <Цена (в рублях)> <Артикул товара> <Название магазина>

Таблица 2: <Код потребителя> <Название магазина> <Артикул товара>

Для каждого товара определить количество покупок данного товара и максимальную цену покупки (вначале выводится количество покупок, затем артикул товара, затем максимальная цена покупки). Если некоторый товар ни разу не был продан, то информация об этом товаре не выводится. Сведения о каждом товаре выводить на новой строке и упорядочивать по возрастанию количества покупок, а в случае одинакового количества — по артикулам товаров в алфавитном порядке.

Лабораторная работа №8. LINQ to DataSet. LINQ to SQL (6 ч)

Блок 1 - LINQ to DataSet

Инструкция для выполнения задания:

1. Для хранения данных использовать DataSet и DataTable
2. Организовать добавления данных
3. Для выборки данных использовать LINQ-запросы

Задания для самостоятельного выполнения

1. Дана база данных, содержащая таблицы, включающие следующие поля:

Таблица 1: <Код потребителя> <Улица проживания> <Год рождения>

Таблица 2: <Скидка (в процентах)> <Код потребителя> <Название магазина>

Для каждого потребителя, указанного в таблице 1, определить количество магазинов, в которых ему предоставляется скидка (вначале выводится количество магазинов, затем код потребителя, потом его улица проживания). Если у некоторого потребителя нет скидки ни в одном магазине, то для него выводится количество магазинов, равное 0. Сведения о каждом потребителе выводить на новой строке и упорядочивать по возрастанию количества магазинов, а при равном количестве – по возрастанию кодов потребителей.

2. Используя БД (см. п.1.), для каждого магазина определить потребителей, имеющих максимальную скидку в этом магазине (вначале выводится название магазина, затем код потребителя, его год рождения и значение максимальной скидки). Если для некоторого магазина имеется несколько потребителей с максимальной скидкой, то вывести данные о потребителе с минимальным кодом. Сведения о каждом магазине выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке.

3. Добавить в базу данных еще две таблицы:

Таблица 3: <Категория> <Страна-производитель> <Артикул товара>

Таблица 4:<Код потребителя> <Артикул товара> <Название магазина>

Для каждой страны-производителя и улицы проживания определить максимальный размер скидки (в процентах) для изделий, произведенных в данной стране и приобретенных потребителями, живущими на данной улице (вначале выводится название страны, затем название улицы, потом максимальный размер скидки). Если для некоторой пары «страна–улица» все товары были приобретены без скидки, то в качестве максимального размера скидки выводится 0. Если для некоторой пары «страна–улица» отсутствует информация о приобретенных товарах, то информация о данной паре не выводится. Сведения о каждой паре «страна–улица» выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке, а для одинаковых названий – по названиям улиц (также в алфавитном порядке).

Блок 2 - LINQ to SQL

Инструкция для выполнения задания:

1. Для разработки базы данных использовать СУБД Microsoft SQL Server
2. Организовать добавления и редактирования данных
3. Для выборки данных использовать LINQ-запросы

Задания для самостоятельного выполнения

1. Дана строка S — название предмета. Таблица базы данных содержит сведения об оценках учащихся по предметам. Каждый запись таблицы содержит данные об одной оценке и включает следующие поля:

<Название предмета> <Фамилия> <Инициалы> <Оценка> <Класс>

Полных однофамильцев (с совпадающей фамилией и инициалами) среди учащихся нет. Класс задается целым числом, оценка — целое число в диапазоне 2–5. Название предмета указывается с заглавной буквы. Для каждого класса, присутствующего в наборе исходных данных, определить число учащихся, имеющих по предмету *S* среднюю оценку не менее 3.5 и при этом не получивших ни одной двойки по этому предмету. Сведения о каждом классе выводить на отдельной строке, указывая номер класса и число найденных учащихся (число может быть равно 0). Данные упорядочивать по возрастанию номера класса.

2. Дана база данных, включающая следующие две таблицы:

Таблица 1: <Цена (в рублях)> <Название магазина> <Артикул товара>

Таблица 2: <Артикул товара> <Название магазина> <Код потребителя>

Для каждой пары «магазин–товар», указанной в таблице 1, определить суммарную стоимость продаж этого товара в данном магазине (вначале выводится название магазина, затем артикул товара, затем суммарная стоимость его продаж). Если товар ни разу не был продан в некотором магазине, то для соответствующей пары «магазин–товар» в качестве суммарной стоимости выводится 0. Сведения о каждой паре «магазин–товар» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а в случае одинаковых названий — по артикулам товаров (также в алфавитном порядке).

3. Добавьте еще одну таблицу:

Таблица 3: <Страна-производитель> <Артикул товара> <Категория>

3.1. Для каждого магазина, указанного в таблице 2, и каждой страны- производителя определить суммарную стоимость товаров из данной страны, проданных в данном магазине (вначале выводится название магазина, затем название страны, затем суммарная стоимость). Если для некоторой пары «магазин–страна» отсутствует информация о проданных товарах, то в качестве суммарной стоимости выводится 0. Сведения о каждой паре «магазин–страна» выводить на новой строке и упорядочивать по названиям магазинов в алфавитном порядке, а

для одинаковых названий магазинов — по названиям стран (также в алфавитном порядке).

3.2. Для каждой страны-производителя определить количество приобретенных товаров из данной страны и суммарную стоимость приобретенных товаров (вначале выводится название страны, затем количество товаров, затем суммарная стоимость). Если сведения о проданных товарах для некоторой страны- производителя отсутствуют, то информация об этой стране не выводится. Сведения о каждой стране выводить на новой строке и упорядочивать по названиям стран в алфавитном порядке.

Лабораторная работа №9. Entity Framework. Работа с несколькими таблицами (6 ч)

Инструкция для выполнения задания:

1. Для разработки базы данных использовать СУБД Microsoft SQL Server и подход Code First.
2. Организовать добавление, редактирование и удаление данных.
3. Для выборки данных использовать LINQ-запросы.

Задания для самостоятельного выполнения

Разработать базу данных, содержащую следующие таблицы:

Таблица «Сотрудник»

Код сотрудника	Целое число, primary key
ФИО	Текст
Дата рождения	Дата
Пол	Символ (М, Ж)
Адрес	Текст

Таблица «Структурное подразделение»

Код подразделения	Целое число, primary key
Название подразделения	Текст

Таблица «Должность»

Код должности	Целое число, primary key
Название должности	Текст
Оклад	Целое число

Таблица «Журнал перемещений»

Код записи	Целое число, primary key
Код сотрудника	Целое число, внешний ключ (Сотрудник)
Код структурного подразделения	Целое число, внешний ключ (Структурное подразделение)
Код должности	Целое число, внешний ключ (Должность)
Ставка	Вещественное число
Зарплата = Оклад * Ставка	Вещественное число
Дата приема на работу	Дата
Номер приказа	Текст

Статус работы	Символ (Р, У)

2. Выполнить запросы, используя LINQ-запросы:

2.1 Вывести работающих молодых сотрудников (до 30 лет) в каждом структурном подразделении. Сначала выводится подразделение, под ним сотрудники с указанием возраста и должности. Отсортировать в порядке возрастания возраста.

2.2 Вывести всех сотрудников, работающих в организации с указанием стажа работы на данный момент. Отсортировать в порядке убывания стажа.

2.3 Вывести сотрудников, работающих в выбранном из раскрывающегося списка структурном подразделении, отсортировать по ФИО.

2.4 Вывести количество принятых и уволенных сотрудников, за выбранный период, с указанием ФИО, должности, подразделения, ставки, даты приема/увольнения, номер приказа.

2.5 Вывести сводный отчет о количестве занятых ставок каждой должности, отсортировать должности в алфавитном порядке.

РАЗДЕЛ 3. ВВЕДЕНИЕ В ASP NET MVC

Лабораторная работа №10. Программирование простейших Web-ориентированных приложения. (5 ч)

Выполните следующие задания:

1. Создайте две основные таблицы: Сотрудники (Шифр, ФИО, дата рождения, пол) и Должности (Название, Базовый оклад). Сотрудник может занимать несколько должностей одновременно. По каждой должности сотрудника должен быть указан размер ставки. Также необходимо вести учет (по признаку) на каких должностях сотрудник еще числится, а с каких уже уволен с указанием причины увольнения. База данных должна хранить дату принятия на работу и дату увольнения по каждой должности сотрудника, сведения о начислениях заработной платы каждому сотруднику по каждой должности с учетом года и месяца.

2. Разработайте Web-приложения для возможности добавления и просмотра данных. При добавлении сотрудника необходимо проверять сколько ставок (по всем должностям) на текущий момент он занимает. Если данный показатель больше 2, выводить соответствующее предупреждение.

3. Добавьте в разработанное **Web-приложения** возможность выборки данных (для вывода списка используйте объект **GridView**):

3.1. Выбрать сотрудников определенной должности.

3.2. Выбрать все занимаемые должности сотрудника на текущий момент с окладом (Базовый оклад * Размер ставки) и датой принятия на эту должность.

3.3. Вывести список сотрудников, принятых в выбранный интервал времени.

3.4. Вывести название всех имеющихся должностей с указанием максимального и минимального оклада.

4. Предусмотреть в системе следующие функции:

1) Формирование ведомости начисления заработной платы за определенной месяц года

Месяц___Год____

№	ФИО	Суммарный Оклад	Премия	НДФЛ	К выдаче

Суммарный Оклад = B1*S1+ B1*S1+…+ Bn*Sn

B – базовый оклад по должности 1,

S – занимаемая ставка подолжности 1,

n – количество занимаемых должностей

Премия: начисляется в размере 15% от оклада, если сотруднику в этом месяце день

рождения.

НДФЛ: 13% от суммы суммарного оклада и премии.

К выдаче: Суммарный оклад + Премия – НДФЛ

2) Формирование сведений о движении сотрудника в хронологическом порядке.

ФИО сотрудника_____

№	Должность	Дата принятия	Дата увольнения	Причина увольнения

Лабораторная работа №11. Программирование Web-ориентированных приложений с использованием AJAX (5 ч)

Для выполнения работы использовать подход к построению интерактивных пользовательских интерфейсов веб-приложений AJAX и валидацию моделей.

Выполните следующие задания

1. Создайте таблицы: Пользователи (id,логин,пароль,ФИО), Сообщения (id, от_кого, кому, заголовок_сообщения, текст, дата_отправки).
2. Создайте главную Web-форму с кнопками *«Регистрация», «Вход»*
3. При нажатии на кнопку *«Регистрация»* должна открываться вторая Web-форма где пользователь может зарегистрироваться с текстовыми полями: логин, пароль, ФИО и кнопкой *«Сохранить»*. Введенные данные должны сохраняться в таблице *«Пользователи».*
4. При нажатии на кнопку *«Вход»* на главной форме должна открываться третья Web-форма с текстовыми полями: логин, пароль и кнопкой *«Вход»*. Пользователь вводит свои данные. Если они являются верными (сравнение с данными, которые хранятся в таблице), то приложение перенаправляет пользователя на четвертую Web-форму или выдается сообщение, что логин и пароль неверны.
5. На четвертой Web-форме при правильном вводе идентификационных данных, должна появится фамилия, имя и отчество пользователя, а также заголовки сообщений, которые были ему адресованы и от кого (ник). Заголовки сообщения являются гиперссылками, при щелчке на который будет открывать пятая Web-форма с текстом сообщения.
6. Пользователь может отправлять сообщения. Для этого необходимо поместить 2 текстовых поля и кнопку *«Отправить»*. В первое текстовое поле вводится ник получателя, а во второе – сообщение. Все отправленные сообщения сохраняются в таблице *«Сообщения»*.

Лабораторная работа №12. Программирование Web-ориентированных приложений с использованием Bootstrap и JqGrid (6 ч)

Выполните следующие задания

1. Разработать Web-приложение «Личный кабинет студента».

Приложение должно отвечать следующим требованиям:

- авторизация студента в личном кабинете. На форме авторизации установить логотип АГАСУ.

- любая страница личного кабинета должна содержать навигационную панель.

- просмотр в личном кабинете успеваемости. Данные должны быть представлены в таблице (**JqGrid**) с тремя колонками: *Семестр, Название дисциплины, Оценка.* Организовать поиск по таблице.

- просмотр в личном кабинете личных достижений студента. Данные должны быть представлены в таблице с колонками: *Дата, Вид (учеба, спорт, наука), Описание.* Для визуализации вида достижения использовать метки. Организовать фильтрацию данных по дате и виду достижений.

- просмотр в личном кабинете данных о преподавателях. Данные должны быть представлены в таблице с колонками: *ФИО преподавателя, Название дисциплины, Название кафедры.*

2. Для разработки использовать функциональность **Bootstrap**

РАЗДЕЛ 4. РАЗРАБОТКА ИНТЕЛЛЕКТУАЛЬНЫХ ПРИКЛАДНЫХ ПРОГРАММ ДЛЯ ОБРАБОТКИ ЕСТЕСТВЕННОГО ЯЗЫКА И РАСПОЗНАВАНИЯ РЕЧИ

Лабораторная работа №13. Реализация алгоритма «Вперед-Назад» (7 ч)

Цель лабораторной работы: изучение алгоритма «Вперед-Назад», реализация прямого и обратного алгоритма в скрытой марковской модели.

Скрытая марковская модель - это цепь Маркова, которая в основном используется в задачах с временной последовательностью данных. Модель Маркова объясняет, что следующий шаг зависит только от предыдущего шага во временной последовательности.

Скрытая марковская модель (θ) имеет следующие параметры:

Набор из M скрытых состояний (S^M)
Матрица вероятности транзакции (A)
Последовательность T наблюдений (V^T)
Матрица вероятности выбросов (также известная как вероятность наблюдения) (B)
Начальное распределение вероятностей (π)

Алгоритм вперед – назад - это обычный рекурсивный и эффективный способ оценки, скрытой марковской модели, то есть вычисления вероятности последовательности наблюдений при данной модели. Эта вероятность может использоваться для классификации последовательностей наблюдений в приложениях распознавания.

В прямом алгоритме используется вычисленная вероятность на текущем временном шаге, чтобы получить вероятность следующего временного шага.

Учитывая последовательность видимого состояния V^T, какова будет вероятность того, что скрытая марковская модель окажется в определенном скрытом состоянии s на определенном временном шаге t.

Обратный алгоритм - это обращенная во времени версия прямого алгоритма. В обратном алгоритме нам нужно найти вероятность того, что машина будет в скрытом состоянии s на временном шаге t и будет генерировать оставшуюся часть последовательности видимого символа V^T.

Данные:

Есть 2 скрытых состояния (A, B) и 3 видимых состояния (0,1,2).

$$A = \begin{bmatrix} 0{,}54 & 0{,}46 \\ 0{,}49 & 0{,}51 \end{bmatrix}$$

$$B = \begin{bmatrix} 0{,}16 & 0{,}26 & 0{,}58 \\ 0{,}25 & 0{,}28 & 0{,}47 \end{bmatrix}$$

В data.csv имеет два столбца с именами Hidden и Visible.

Задание на лабораторную работу: Необходимо реализовать прямой и обратный алгоритм в скрытой марковской модели.

Лабораторная работа №14. Сверточные коды и алгоритм декодирования Витерби (7 ч)

Цель лабораторной работы: изучение алгоритма декодирования Витерби.

Задание на лабораторную работу: в соответствии с номером варианта требуется закодировать информационную последовательность битов с использованием сверточного кода, а также произвести декодирование принятой кодовой комбинации по алгоритму Витерби, определить наличие ошибок в принятой кодовой комбинации и исправить их.

Задание на выполнение лабораторной работы согласно номеру варианта

Номер варианта	Входной информационный сигнал	Принятый сигнал
1	11010110100010	11011111001011110100011111010111011111101
2	10010110101010	11011111101011010100011111011100001011101
3	01011110101010	11011111001000101000011110011100000011101
4	11110110110011	11011111101010101000011111010000011001101
5	10001110101010	11111111101000101000010100001101110011101
6	10010010000011	11011011101011000011011100001101100011101
7	11111110101000	11011111100010110011011100000101110011101
8	00010110101111	11011110100110110011010001111101110011001
9	10011111101110	11010111010001110011010001111100110011101
10	10110001101001	11011111010010101011010001111100110011101
11	10101101100110	11011110010010101111100010111101110011101
12	10000100001011	11010111010010101111100110111101001100011
13	10110111101011	11011111010000100010010110110101000100011
14	10010111111011	11011110010010100010010110111110011100011
15	11110000101110	11011101010010100010010110111100010101110

Рисунок 1

1. Для сверточного кодера, схема которого приведена на рис. 2, требуется закодировать информационную последовательность битов на входе кодера с целью получения выходного сигнала – сверточного кода.

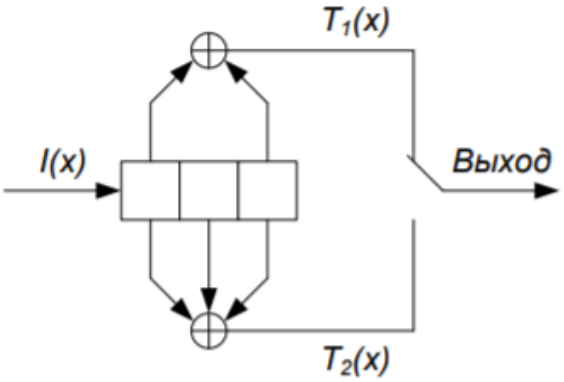


Рисунок 2

2. Написать функцию, реализующую декодирование входных данных декодером, построенному по алгоритму Витерби.

3. В результате выполнения данного этапа студенту требуется найти ошибки в полученной кодовой комбинации, указать их позиции и исправить.

Лабораторная работа №15. Анализ текста как источника больших данных. (4 ч)

Анализ текста состоит из следующих модулей:

– Графематический анализ (сегментация), т. е. выделение в тексте предложений и словоформ, точнее токенов (т. к. в тексте могут быть не только слова) — переход от символов к словам;

– Морфологический анализ — переход от словоформ к их леммам (словарным формам лексем) или основам (ядерным частям слова, за вычетом словоизменительных морфем);

– Синтаксический анализ — выявление синтаксических связей слов и грамматической структуры предложений;

– Семантический и прагматический анализ, при котором определяется смысл фраз и соответствующая реакция системы.

Цель лабораторной работы: познакомиться с библиотеками NLP.

Задание на лабораторную работу:

1. Выбрать вырезку из статьи по машинному обучению.
2. Установить библиотеку NLP для работы с текстом.
3. Провести предварительную обработку текста по приведенным ниже этапам.

Очистка текстовых данных необходима, чтобы выделить атрибуты, которые мы хотим использовать в нашей системе машинного обучения. Очистка (или предварительная обработка) данных обычно состоит из нескольких этапов:

– ***Удалить пунктуацию***

Пунктуация может обеспечить грамматический контекст для предложения, которое поддерживает наше понимание. Но для нашего векторизатора, который считает количество слов, а не контекст, он не добавляет значения, поэтому мы удаляем все специальные символы.

– ***Tokenization***

Токенизация разделяет текст на блоки, такие как предложения или слова. Это дает структуру ранее неструктурированному тексту.

– Удалить стоп-слова

Стоп-слова - это обычные слова, которые, скорее всего, появятся в любом тексте. Они мало говорят нам о наших данных, поэтому мы их удаляем.

– ***Stemming***

Стемминг помогает свести слово к его форме. Часто имеет смысл рассматривать похожие слова одинаково. Он удаляет достаточно, например, «ing», «ly», «s» и т. Д. Простым подходом, основанным на правилах. Это уменьшает корпус слов, но часто фактическими

словами пренебрегают.

– ***Лемматизация***

Лемматизация выводит каноническую форму («лемму») слова. т.е. корневая форма. Это лучше, чем основа, так как он использует подход на основе словаря, то есть морфологический анализ к корню. Stemming обычно быстрее, так как он просто отсекает конец слова, не понимая контекста слова. Лемматизация происходит медленнее и точнее, так как требует осознанного анализа с учетом контекста слова.

Векторизация данных

Векторизация - это процесс кодирования текста в виде целых чисел, то есть числовой формы для создания векторов признаков, чтобы алгоритмы машинного обучения могли понимать наши данные.

– ***Мешок слов***

Bag of Words (BoW) или CountVectorizer описывает наличие слов в текстовых данных. Это дает результат 1, если присутствует в предложении и 0, если не присутствует. Поэтому в каждом текстовом документе создается пакет слов с количеством матриц документов.

– ***TF-IDF***

Вычисляет «относительную частоту» появления слова в документе по сравнению с его частотой во всех документах. Это более полезно, чем термин «частота» для определения «важных» слов в каждом документе.

Заметка: Используется для оценки в поисковых системах, суммирования текста, кластеризации документов.

Методические указания рекомендованы на заседании МКН подготовки 09.04.02 «Информационные системы и технологии» профиль «Искусственный интеллект в проектировании городской среды» к размещению на образовательном портале ГАОУ АО ВО «АГАСУ» (http://moodle.aucu.ru)