Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

**Звіт**

З лабораторної роботи №2

з курсу

«Основи розробки програмного забезпечення на платформі Microsoft.NET»

**Виконав студент** *ІП-01 Галько Міла*
(шифр, ПІБ)

**Перевірила** *Ліщук К.І.*
(ПІБ)

Київ 2022

# Комп‘ютерний практикум № 2. LINQ to XML

**Мета**:

- ознайомитися з обробкою XML документів з використанням технології LINQ to XML

## Постановка задачі комп‘ютерного практикуму № 2

При виконанні комп‘ютерного практикуму необхідно виконати наступні дії:

1) Розробити структуру XML для зберігання даних згідно варіанта № 1.
2) Створити XML-файл з використанням XmlWriter. Дані необхідно вводити з консолі, зберегти його. Завантажити файл з використанням XmlDocument.
3) LINQ to XML:
    a. Вивести зміст файлу, створеного в п.2.
    b. Для файлу, створеного в п.2 розробити як мінімум 15 різних запитів, використовуючи різні дії над отриманими даними. Запити не повинні повторюватись.
4) Створити програмне забезпечення, котре реалізує обробку даних з використання бібліотеки LINQ to XML.
5) Програмне забезпечення необхідно розробити у вигляді консольного застосування на мові C#.
6) Коротко описати архітектуру проекту та створити звіт, котрий завантажити в moodle

## Варіант №1

1) Розробити структуру даних для зберігання інформації про студентів дипломників та їх керівників. Про студентів необхідно зберігати щонайменше наступну інформацію: ПІБ, група, дата народження, середній бал. Про керівників: ПІБ, посада. У одного керівника може бути декілька студентів-дипломників.

## Програмний код:

Посилання на github: https://github.com/MilaHalko/C4_.NET/tree/Lab2/Lab2_XML/Lab2_XML
Також код доданий в кінці документу.

## Про файл XML:

В результаті створення відповідних класів була створена наступна структура файлу XML з елементами: students, supervisors, Subjects. Відповідно до цього, у всіх 3-х елементів спадкоємцями є усі атрибути з їх початкових класів. Де атрибутом були контейнери з об’єктами інших класів, був взятий лише id об’єкта.

## Про програмне забезпечення:

Структура ПЗ має наступну структуру:

1) Створення об’єктів і їх заповнення даними;
2) Створення структури XML файлу (описано у п.«Про файл XML»);
3) Взаємодія з XML файлом для виводу вмісту (Рис. 1);
4) LINQ запити через XML файл (п. «LINQ to XML»).

## LINQ to XML:

Кінцевою та найбільшою частиною є LINQ запити до XML. В ході розробки було створено 15 запитів із використанням різних дій над множинами: order by, where, group by, take, skip while, join, union та ін. (Рис. 2).

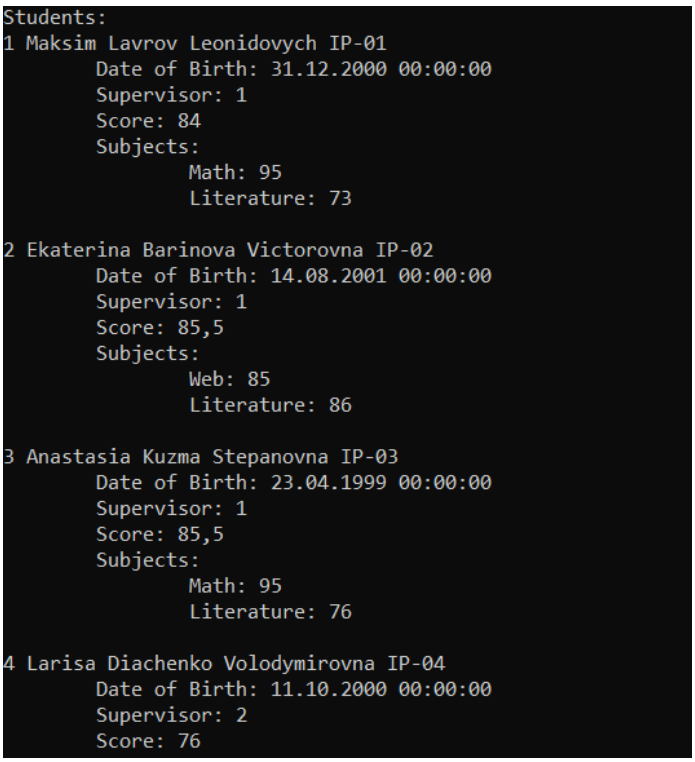


Рис. 1 - Приклад результату виведення вмісту файлу

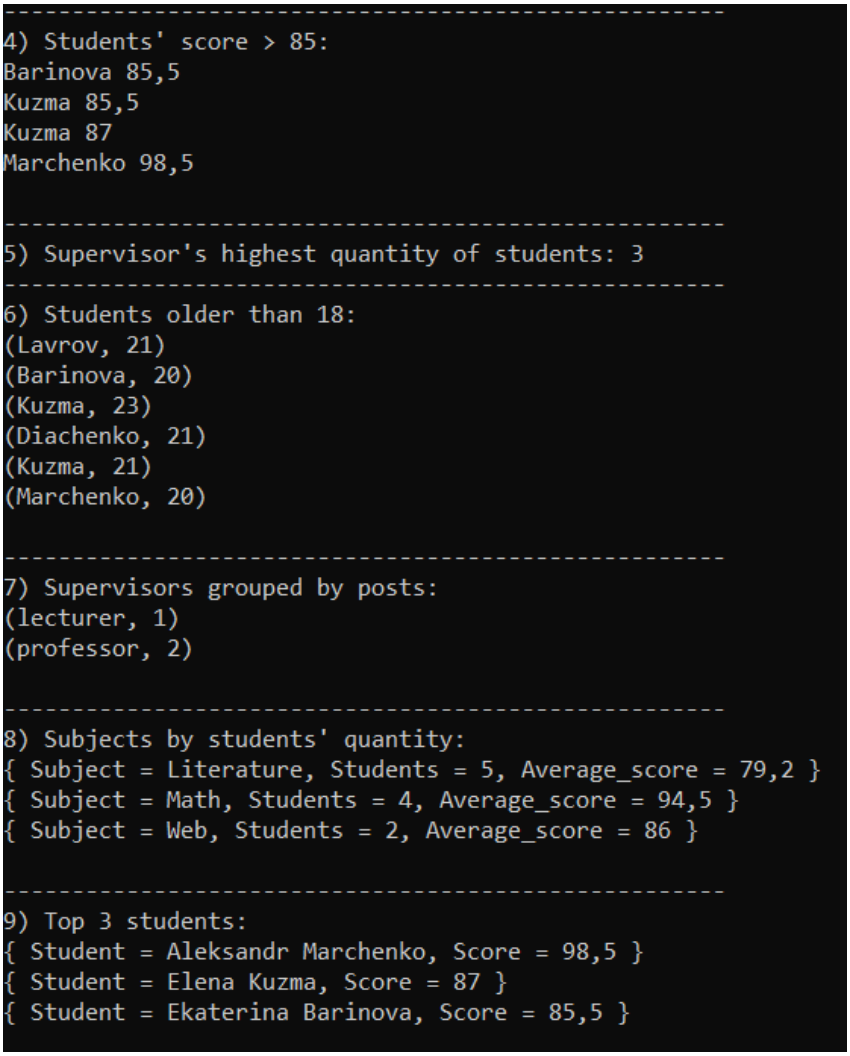


Рис. 2 – Результати виведення запитів 4-9

Файл Student.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Student
    {
        public int Id { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Patronymic { get; set; }
        public string Group { get; set; }
        public DateTime BirthDate { get; set; }

        public int SupervisorID { get; set; }


        public Dictionary<Subject, int> Scores = new Dictionary<Subject, int>();
        public float AverageScore {
            get
            {
                float sum = 0;
                foreach (var score in Scores)
                {
                    sum += score.Value;
                };
                return sum / Scores.Count;
            }
        }

        public override string ToString()
        {
            return String.Format(@"Student: {0} {1} {2} ID: {3} Group: {4}", LastName, FirstName, Patronymic, Id,
Group);
        }
    }
}
```

Файл Supervisor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Supervisor
    {
        public int ID { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Patronymic { get; set; }
        public string Post { get; set; }
        public ICollection<Student> Students { get; set; } = new List<Student>();

        public override string ToString()
        {
            string students = string.Join(" ", Students);
            return String.Format(@"
                Supervisor: {0} {1} {2}
                Post: {3}
                    Students: {4}", LastName, FirstName, Patronymic, Post, students);
        }

        internal object Join(List<Student> students, Func<object, object> p1, Func<Student, Student> p2, Func<obje
ct, object, object> p3)
        {
            throw new NotImplementedException();
        }
    }
}
```

Файл Subject.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Lab1_LINQ
{
    internal class Subject
    {
        public string Name { get; set; }
        public ICollection<Student> Students { get; set; } = new List<Student>();

        public override string ToString()
        {
            string students = string.Join(" ", Students);
            return String.Format(@"
                Subject: {0}
                    Students: {1}", Name, students);
        }
    }
}
```

Файл Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Xml;
using System.Xml.Linq;

namespace Lab2_XML
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            // Students
            Student st1 = new Student { Id = 1, FirstName = "Maksim", LastName = "Lavrov", Patronymic = "Leonid
            ovych", Group = "IP-01", BirthDate = new DateTime(2000, 12, 31) };
            Student st2 = new Student { Id = 2, FirstName = "Ekaterina", LastName = "Barinova", Patronymic = "V
            ictorovna", Group = "IP-02", BirthDate = new DateTime(2001, 8, 14) };
            Student st3 = new Student { Id = 3, FirstName = "Anastasia", LastName = "Kuzma", Patronymic = "Step
            anovna", Group = "IP-03", BirthDate = new DateTime(1999, 4, 23) };
            Student st4 = new Student { Id = 4, FirstName = "Larisa", LastName = "Diachenko", Patronymic = "Vol
            odymirovna", Group = "IP-04", BirthDate = new DateTime(2000, 10, 11) };
            Student st5 = new Student { Id = 5, FirstName = "Elena", LastName = "Kuzma", Patronymic = "Vyloriiv
            na", Group = "IT-01", BirthDate = new DateTime(2001, 2, 11) };
            Student st6 = new Student { Id = 6, FirstName = "Aleksandr", LastName = "Marchenko", Patronymic = "
            Aleksandrovych", Group = "IT-02", BirthDate = new DateTime(2002, 3, 15) };

            List<Student> students = new List<Student> { st1, st2, st3, st4, st5, st6 };


            // Supervisors
            List<Student> students1 = new List<Student> { st1, st2, st3 };
            List<Student> students2 = new List<Student> { st4 };
            List<Student> students3 = new List<Student> { st5, st6 };

            Supervisor sup1 = new Supervisor { ID = 1, FirstName = "Victor", LastName = "Barinov", Patronymic =
             "Petrovich", Post = "professor", Students = students1 };
            Supervisor sup2 = new Supervisor { ID = 2, FirstName = "Victoria", LastName = "Honcharova", Patrony
            mic = "Yevhenevna", Post = "lecturer", Students = students2 };
            Supervisor sup3 = new Supervisor { ID = 3, FirstName = "Robert", LastName = "Dawny", Patronymic = "
            Second", Post = "professor", Students = students3 };

            List<Supervisor> allSupervisors = new List<Supervisor> { sup1, sup2, sup3 };


            // Subjects
            List<Student> students4 = new List<Student> { st1, st3, st4, st6 };
            List<Student> students5 = new List<Student> { st2, st5, st6 };
            List<Student> students6 = new List<Student> { st1, st2, st3, st4 };
```

```
Subject sub1 = new Subject { Name = "Math", Students = students4 };
Subject sub2 = new Subject { Name = "Web", Students = students5 };
Subject sub3 = new Subject { Name = "Literature", Students = students6 };

List<Subject> allSubjects = new List<Subject> { sub1, sub2, sub3 };


// Set Subjects' Scores & SupervisorIDs for Students
st1.Scores[sub1] = 95;
st1.Scores[sub3] = 73;
st1.SupervisorID = 1;

st2.Scores[sub2] = 85;
st2.Scores[sub3] = 86;
st2.SupervisorID = 1;

st3.Scores[sub1] = 95;
st3.Scores[sub3] = 76;
st3.SupervisorID = 1;

st4.Scores[sub1] = 89;
st4.Scores[sub3] = 63;
st4.SupervisorID = 2;

st5.Scores[sub2] = 87;
st5.SupervisorID = 3;

st6.Scores[sub1] = 99;
st6.Scores[sub3] = 98;
st6.SupervisorID = 3;


// XML File Writer
XmlWriterSettings settings = new XmlWriterSettings();
settings.Indent = true;

using (XmlWriter writer = XmlWriter.Create("university.xml", settings))
{
    writer.WriteStartElement("university");

    writer.WriteStartElement("students");
    foreach (var st in students)
    {
        writer.WriteStartElement("student");
        writer.WriteAttributeString("id", st.Id.ToString());
        writer.WriteElementString("fName", st.FirstName);
        writer.WriteElementString("lName", st.LastName);
        writer.WriteElementString("patronymic", st.Patronymic);
        writer.WriteElementString("group", st.Group);
        writer.WriteElementString("DOB", st.BirthDate.ToString());
        writer.WriteElementString("supId", st.SupervisorID.ToString());
        writer.WriteElementString("score", st.AverageScore.ToString());

        writer.WriteStartElement("subjects");
        foreach (var sub in st.Scores)
        {
            writer.WriteStartElement("subject");
            writer.WriteAttributeString("name", sub.Key.Name);
            writer.WriteElementString("score", sub.Value.ToString());
            writer.WriteEndElement();
        }
        writer.WriteEndElement();
        writer.WriteEndElement();
    }
    writer.WriteEndElement();

    writer.WriteStartElement("supervisors");
    foreach (var s in allSupervisors)
    {
        writer.WriteStartElement("supervisor");
        writer.WriteAttributeString("id", s.ID.ToString());
        writer.WriteElementString("fName", s.FirstName);
        writer.WriteElementString("lName", s.LastName);
        writer.WriteElementString("patronymic", s.Patronymic);
        writer.WriteElementString("post", s.Post);

        writer.WriteStartElement("sup-students");
```

```
            foreach (var st in s.Students)
            {
                writer.WriteStartElement("sup-student");
                writer.WriteAttributeString("id", st.Id.ToString());
                writer.WriteEndElement();
            }
            writer.WriteEndElement();
            writer.WriteEndElement();
        }
        writer.WriteEndElement();

        writer.WriteStartElement("Subjects");
        foreach (var s in allSubjects)
        {
            writer.WriteStartElement("sub-students");
            foreach (var st in s.Students)
            {
                writer.WriteStartElement("sub-student");
                writer.WriteAttributeString("id", st.Id.ToString());
                writer.WriteAttributeString("subject", s.Name);
                writer.WriteEndElement();
            }

        }
        writer.WriteEndElement();

        writer.WriteEndElement();
    }

     var doc = new XmlDocument();
    doc.Load("university.xml");

    Console.WriteLine("Students:");
    foreach (XmlNode st in doc.GetElementsByTagName("student"))
    {
        var id = st.Attributes["id"].Value;
        var fName = st["fName"].InnerText;
        var lName = st["lName"].InnerText;
        var patronymic = st["patronymic"].InnerText;
        var group = st["group"].InnerText;
        var DOB = st["DOB"].InnerText;
        var supID = st["supId"].InnerText;
        var score = st["score"].InnerText;

        var subjects = new Dictionary<string, string>();
        foreach (XmlNode sub in st["subjects"])
        {
            subjects[sub.Attributes["name"].Value] = sub["score"].InnerText;
        }

        Console.WriteLine("{0} {1} {2} {3} {4}\n" +
                          "\tDate of Birth: {5}\n" +
                          "\tSupervisor: {6}\n" +
                          "\tScore: {7}\n" +
                          "\tSubjects:",
                          id, fName, lName, patronymic, group, DOB, supID, score);
        foreach (var s in subjects)
        {
            Console.WriteLine("\t\t{0}: {1}", s.Key, s.Value);
        }
        Console.WriteLine();
    }
    Console.WriteLine();

    Console.WriteLine("Supervisors:");
    foreach (XmlNode sup in doc.GetElementsByTagName("supervisor"))
    {
        var id = sup.Attributes["id"].Value;
        var fName = sup["fName"].InnerText;
        var lName = sup["lName"].InnerText;
        var patronymic = sup["patronymic"].InnerText;
        var post = sup["post"].InnerText;

        var studentsID = new List<string>();
        foreach (XmlNode st in sup["sup-students"])
        {
            studentsID.Add(st.Attributes["id"].Value);
        }
```

```
    Console.WriteLine("{0} {1} {2} {3} - {4}", id, fName, lName, patronymic, post);
    Console.Write("\tStudentsID: ");
    for (int i = 0; i < studentsID.Count; i++)
    {
        Console.Write(studentsID[i]);
        if (i != studentsID.Count - 1)
        {
            Console.Write(", ");
        }
    }
    Console.WriteLine("\n");
}

Console.WriteLine("Sub-students:");
foreach (XmlNode subSt in doc.GetElementsByTagName("sub-student"))
{
    var id = subSt.Attributes["id"].InnerText;
    var subName = subSt.Attributes["subject"].InnerText;
    Console.WriteLine("{0} - studentID: {1}", subName, id);
}
Console.WriteLine();


Console.WriteLine("LINQ to XML");
XDocument d = XDocument.Load("university.xml");

//-------------------------------------------------------------------------------------
string mess1 = "Students + scores";
var stsByGroup = from st in d.Root.Elements("students").Elements("student")
                 select new
                 {
                     Name = st.Element("fName").Value + " " + st.Element("lName").Value,
                     Score = st.Element("score").Value
                 };
PrintLINQ(stsByGroup, mess1);


//-------------------------------------------------------------------------------------
string mess2 = "Students + subjects' quantity";
var stsSubsQuantity = from st in d.Root.Elements("students").Elements("student")
                      orderby st.Element("subjects").Elements("subject").Count()
                      orderby st.Element("lName").Value
                      select new
                      {
                          Name = st.Element("lName").Value,
                          SubjectsCount = st.Element("subjects").Elements("subject").Count()
                      };
PrintLINQ(stsSubsQuantity, mess2);


//-------------------------------------------------------------------------------------
string mess3 = "Students grouped by Lastnames";
var stsGroupedLNames = from st in d.Root.Descendants("student")
                       .GroupBy(s => s.Element("lName").Value)
                       .Select(s => new { Lastname = s.Key, Quantity = s.Count() }) select st;
PrintLINQ(stsGroupedLNames, mess3);


//-------------------------------------------------------------------------------------
string mess4 = "Students' score > 85";
var stsScoreU85 = from st in d.Root.Descendants("student")
                  where Double.Parse(st.Element("score").Value) > 85
                  select st.Element("lName").Value + " " + st.Element("score").Value;
PrintLINQ(stsScoreU85, mess4);


//-------------------------------------------------------------------------------------
string mess5 = "Supervisor's highest quantity of students";
var supsHiStsQ = d.Root.Descendants("supervisor")
                 .Max(s => s.Element("sup-students").Elements("sup-student").Count());
PrintResult(mess5, supsHiStsQ);


//-------------------------------------------------------------------------------------
string mess6 = "Students older than 18";
var adults = d.Root.Descendants("student")
         .Where(s => DateTime.Parse(s.Element("DOB").Value).AddYears(18) <= DateTime.Now)
```

```
            .Select(s => (s.Element("lName").Value,
                          Math.Floor((DateTime.Now - DateTime.Parse(s.Element("DOB").Value)).TotalDays
 / 365.25)));
PrintLINQ(adults, mess6);


//-----------------------------------------------------------------------------------------
string mess7 = "Supervisors grouped by posts";
var supsGBPosts = d.Root.Descendants("supervisor")
                  .GroupBy(s => s.Element("post").Value)
                  .OrderBy(s => s.Count())
                  .Select(s => (s.Key, s.Count()));
PrintLINQ(supsGBPosts, mess7);


//-----------------------------------------------------------------------------------------
string mess8 = "Subjects by students' quantity";
var subsA = d.Root.Descendants("subject")
            .GroupBy(s => s.Attribute("name").Value)
            .OrderByDescending(s => s.Count())
            .Select(s => new
            {
                Subject = s.Key,
                Students = s.Count(),
                Average_score = (from sub in d.Root.Descendants("subject")
                                 where sub.Attribute("name").Value == s.Key
                                 select (double)sub.Element("score")).Average()
            });
PrintLINQ(subsA, mess8);


//-----------------------------------------------------------------------------------------
string mess9 = "Top 3 students";
var top3Sts = d.Root.Descendants("student")
              .OrderByDescending(s => Double.Parse(s.Element("score").Value))
              .Take(3)
              .Select(s => new
              {
                  Student = s.Element("fName").Value + " " + s.Element("lName").Value,
                  Score = s.Element("score").Value
              });
PrintLINQ(top3Sts, mess9);


//-----------------------------------------------------------------------------------------
string mess10 = "Middle students (skip while score > 90 & take 4)";
var sub3MiddleSts = d.Root.Descendants("student")
                    .OrderByDescending(s => Double.Parse(s.Element("score").Value))
                    .SkipWhile(s => Double.Parse(s.Element("score").Value) > 90)
                    .Take(4)
                    .Select(s => (s.Element("fName").Value, s.Element("score").Value));
PrintLINQ(sub3MiddleSts, mess10);


//-----------------------------------------------------------------------------------------
string mess11 = "Students joined supervisors";
var supsSts = from st in d.Root.Descendants("student")
              join sup in d.Root.Descendants("supervisor")
              on st.Element("supId").Value equals sup.Attribute("id").Value
              orderby st.Element("fName").Value
              select new
              {
                  Student = st.Element("fName").Value + " " + st.Element("lName").Value,
                  Supervisor = sup.Element("fName").Value + " " + sup.Element("lName").Value
              };
PrintLINQ(supsSts, mess11);


//-----------------------------------------------------------------------------------------
string mess12 = "Union Supervisor 1 & 3 students' id";
var sups23Sts = d.Root.Descendants("supervisor")
                .Where(s => s.Attribute("id").Value == "1")
                .Union(d.Root.Descendants("supervisor")
                       .Where(x => x.Attribute("id").Value == "3"))
                .Select(s => new
                {
                    Supervisor = s.Element("lName"),
                    Students = s.Element("sup-students").Elements("sup-student")
```

```
                            });
            PrintMessage(mess12);
            Console.WriteLine();
            foreach (var sup in sups23Sts)
            {
                foreach (var item in sup.Students)
                {
                    Console.Write("{0} ", item.Attribute("id").Value);
                }
            }
            Console.WriteLine("\n");


            //-------------------------------------------------------------------------------------------
            string mess13 = "Intersect -> middle students";
            var sts46InterSubs = d.Root.Descendants("student")
                                .Where(s => Double.Parse(s.Element("score").Value) < 90)
                                .Intersect(d.Root.Descendants("student")
                                          .Where(x => Double.Parse(x.Element("score").Value) > 80))
                                .Select(s => s.Element("lName").Value + " " + s.Element("score").Value);
            PrintLINQ(sts46InterSubs, mess13);


            //-------------------------------------------------------------------------------------------
            string mess14 = "Math's students except student4";
            var mathExceptSt4 = d.Root.Descendants("sub-student")
                                 .Where(sub => sub.Attribute("subject").Value == "Math")
                                 .Except(d.Root.Descendants("sub-student")
                                         .Where(x => x.Attribute("id").Value == "4"))
                                 .Select(sub => sub.Attribute("id").Value);
            PrintLINQ(mathExceptSt4, mess14);


            //-------------------------------------------------------------------------------------------
            string mess15 = "Students with 'A..' first name";
            var hSts = from st in d.Root.Descendants("student")
                       where Regex.IsMatch(st.Element("fName").Value, @"^(A|a)")
                       select st.Element("fName").Value;
            PrintLINQ(hSts, mess15);

            Console.ReadLine();
        }

        public static int counter = 1;
        private static void PrintLINQ<T>(IEnumerable<T> list, string mess = "")
        {
            if (mess != "")
            {
                PrintMessage(mess);
                Console.WriteLine();
            }
            foreach (T item in list)
            {
                Console.WriteLine(item);
            }
            Console.WriteLine();
        }
        private static void PrintResult<T>(string mess, T result)
        {
            PrintMessage(mess);
            Console.WriteLine(result);
        }
        private static void PrintMessage(string mess)
        {
            Console.WriteLine("-------------------------------------------------------");
            Console.Write("{0}) {1}: ", counter, mess);
            counter++;
        }
    }
}
```