МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

Лабораторна робота № 5

з дисципліни “Основи програмування”
тема “Формат даних XML”

Виконав	Перевірив
студент І курсу	“____” “_____” 20___ р.
групи КП-01	викладач
Беліцький Олександр Сергійович	Гадиняк Руслан Анатолійович
(*прізвище, ім’я, по батькові*)	(*прізвище, ім’я, по батькові*)

варіант № 3

Київ 2021

Мета роботи

Навчитись виконувати серіалізацію і десеріалізацію даних у форматі XML.

Виконати генерацію зображення з графіком на основі вхідних даних.

Постановка завдання

Задано файл з даними у форматі XML.
Створити консольну програму, що дозволяє користувачу виконувати операції над даними із файлів у форматі XML заданої структури: десеріалізувати набір даних із файлу, згенерувати і зберегти частину даних у новий XML файл, обчислити і вивести дані за варіантом, а також вивести задані дані на зображення з графіком та зберегти його у файл.

Користувач керує програмою за допомогою командного інтерфейсу у консолі.
Консольні команди користувача:

- **`load {filename}` -** десеріалізувати XML із заданого файлу у об'єкти в процесі.
- **`print {pageNum}` -** вивести загальну кількість сторінок і дані сторінки (за номером) десеріалізованих даних з об'єктів у консоль. Розмір сторінки довільний.
- **`save {filename}` -** серіалізувати всі дані у заданий XML файл (з відступами).
- **`export {N} {filename}` -** зберегти у XML перші N курсів з найвищою кількістю кредитів (units).
- *три команди на отримання даних за варіантом.* Отримані дані виводити в консоль.
- **`image {filename}` -** створити і зберегти зображення з графіком за варіантом у файл.

Отримання даних:

1. **`subjects` -** Список назв всіх унікальних навчальних предметів (subj)
2. **`subject {subj}` -** Список назв курсів (title) обраного предмету (subj)
3. **`instructors` -** Список всіх унікальних викладачів (instructor)

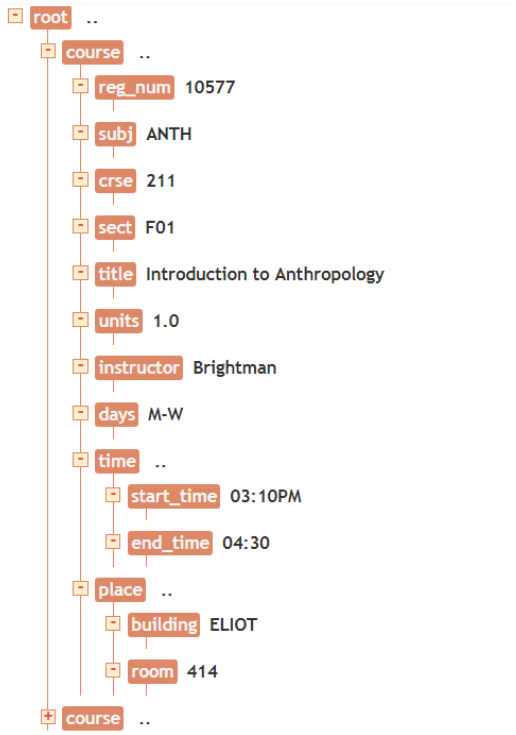
Графік: сумарна кількість кредитів (units) по кожному предмету (subj).
Тип графіка: horizontal bars

Аналіз вимог і проектування

Відповідно до варіанту частина Xml-файлу:

```
▼<root>
  ▼<course>
     <reg_num>10577</reg_num>
     <subj>ANTH</subj>
     <crse>211</crse>
     <sect>F01</sect>
     <title>Introduction to Anthropology</title>
     <units>1.0</units>
     <instructor>Brightman</instructor>
     <days>M-W</days>
    ▼<time>
       <start_time>03:10PM</start_time>
       <end_time>04:30</end_time>
     </time>
    ▼<place>
       <building>ELIOT</building>
       <room>414</room>
     </place>
   </course>
  ▶<course>
   ...
```

Для даного Xml-файлу відповідне DOM-дерево буде мати вигляд:



Тексти коду програм

Program.cs

```
using System;
using System.Collections.Generic;
using System.IO;


/*
SerializeProcess - Serialize (запис у файл)  !_ save {filename}
DeserializeProcess - Deserialize (читка з файлу)  -- ОБОВ'ЯЗКОВО ДОДАТИ НА ІНШІ ДІЇ ПЕРЕВІРКУ ЗАГРУЗКИ
ФАЙЛУ !- load {filename}


GetTitles - IsExist subject
Export - n > 0 AND n < root.courses.count
GetPage - перевірку на >= 1 AND < pages
*/

class Program
{
    static void Main(string[] args)
    {
        Root root = new Root();
        List<string> list = new List<string>();
        while (true)
        {
            Console.WriteLine("Enter command: ");
            string command = Console.ReadLine();
            string[] subcommands = command.Split(" ");
            if (command == "exit")
            {
                Console.WriteLine("Ending processing...");
                break;
            }
            else if (subcommands[0] == "subjects")
            {
                if (subcommands.Length != 1)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have only name of
command");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    var subjects = GetData.GetSubjects(root);
                    for (int i = 1; i <= subjects.Count; i++)
                    {
                        Console.Write("[{0}]\t ", subjects[i - 1]);
                        if (i % 8 == 0)
                        {
                            Console.WriteLine();
                        }
                    }
                    Console.WriteLine();
                }
            }
            else if (subcommands[0] == "instructors")
            {
                if (subcommands.Length != 1)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have only name of
command");
                }
```

```
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    var instructors = GetData.GetInstructors(root);
                    for (int i = 1; i <= instructors.Count; i++)
                    {
                        Console.Write("{0, -13} ", instructors[i - 1]);
                        if (i % 8 == 0)
                        {
                            Console.WriteLine();
                        }
                    }
                    Console.WriteLine();
                }
            }
            else if (subcommands[0] == "load")
            {
                if (subcommands.Length != 2)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have additional
argument");
                }
                else
                {
                    if (File.Exists(subcommands[1]))
                    {
                        if (subcommands[1].EndsWith(".xml"))
                        {
                            root = XmlProcess.Deserialize(subcommands[1]);
                            Console.WriteLine("Tip: Xml was upload");
                        }
                        else
                        {
                            Console.WriteLine("Error: check type of input file. Must be xml");
                        }
                    }
                    else
                    {
                        Console.WriteLine($"Error: file `{subcommands[1]}` does not exist");
                    }
                }
            }
            else if (subcommands[0] == "print")
            {
                if (subcommands.Length != 2)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have additional
argument");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    if (int.TryParse(subcommands[1], out int page))
                    {
                        if (page < 1 || page > 141)
                        {
                            Console.WriteLine("Error: Wrong page number. Must be 1 - 141, but have
`{0}`", page);
                        }
                        else
                        {
                            GetData.PrintPage(root, page);
                        }
                    }
                    else
```

```
                    {
                        Console.WriteLine("Error: argument must be integer number");
                    }
                }
            }
            else if (subcommands[0] == "save")
            {
                if (subcommands.Length != 2)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have additional
argument");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    if (subcommands[1].EndsWith(".xml"))
                    {
                        XmlProcess.Serialize(subcommands[1], root);
                        Console.WriteLine("Tip: Xml was saved");
                    }
                    else
                    {
                        Console.WriteLine("Error: check type of output file. Must be xml");
                    }
                }
            }
            else if (subcommands[0] == "image")
            {
                if (subcommands.Length != 2)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have additional
argument");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    if (subcommands[1].EndsWith(".png"))
                    {
                        ImageProcess.CreateImage(subcommands[1], root);
                        Console.WriteLine("Tip: Diagram was saved");
                    }
                    else
                    {
                        Console.WriteLine("Error: check type of output file. Must be png");
                    }
                }
            }
            else if (subcommands[0] == "subject")
            {
                if (subcommands.Length != 2)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have additional
argument");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    string subject = "";
                    for (int i = 0; i < root.courses.Count; i++)
                    {
                        if (root.courses[i].subject == subcommands[1])
                        {
```

```
                            subject = subcommands[1];
                            break;
                        }
                    }
                    if (subject == "")
                    {
                        Console.WriteLine($"Error: Subject {subcommands[1]} does not found");
                    }
                    else
                    {
                        var titles = GetData.GetTitles(root, subject);
                        for (int i = 1; i <= titles.Count; i++)
                        {
                            Console.Write("[{0}]\t ", titles[i - 1]);
                            if (i % 8 == 0)
                            {
                                Console.WriteLine();
                            }
                        }
                        Console.WriteLine();
                    }
                }
            }
            else if (subcommands[0] == "export")
            {
                if (subcommands.Length != 3)
                {
                    Console.WriteLine($"Error: Command `{subcommands[0]}` must have 2 additional
argument");
                }
                else if (root.courses == null)
                {
                    Console.WriteLine("Error: firstly, upload xml");
                }
                else
                {
                    if (int.TryParse(subcommands[1], out int n))
                    {
                        if (n < 1 || n > root.courses.Count)
                        {
                            Console.WriteLine($"Error: Wrong number. Must be 1 - {root.courses.Count},
but have `{0}`", n);
                        }
                        else
                        {
                            if (subcommands[2].EndsWith(".xml"))
                            {
                                XmlProcess.Export(n,subcommands[2], root);
                                Console.WriteLine("Tip: Data was exported");
                            }
                            else
                            {
                                Console.WriteLine("Error: check type of output file. Must be xml");
                            }
                        }
                    }
                    else
                    {
                        Console.WriteLine("Error: argument must be integer number");
                    }
                }
            }
            else
            {
                Console.WriteLine($"Error: Unknown command {command}");
            }
        }
    }


}
```

Course.cs

```
using System.Collections.Generic;
using System.Xml.Serialization;


public class Course
{
    [XmlElement("reg_num")]
    public int registerNumber;
    [XmlElement("subj")]
    public string subject;
    [XmlElement("crse")]
    public string course;
    [XmlElement("sect")]
    public string sector;
    public string title;
    public double units;
    public string instructor;
    public string days;
    public Time time;
    public Place place;
}
[XmlType(TypeName = "time")]
public class Time
{
    [XmlElement("start_time")]
    public string startTime;
    [XmlElement("end_time")]
    public string endTime;
}
[XmlType(TypeName = "place")]
public class Place
{
    public string building;
    public string room;
}

[XmlRoot("root")]
public class Root
{
    [XmlElement("course")]
    public List<Course> courses;
}
```

XmlProcess.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Xml.Serialization;



public static class XmlProcess
{
    public static void Serialize(string outFile, Root root)
    {
        XmlSerializer ser = new XmlSerializer(typeof(Root));
        System.IO.StreamWriter writer = new System.IO.StreamWriter(outFile);
        ser.Serialize(writer, root);
        writer.Close();
    }

    public static Root Deserialize(string inputFile)
    {
        XmlSerializer ser = new XmlSerializer(typeof(Root));
        StreamReader reader = new StreamReader(inputFile);
        Root value = (Root)ser.Deserialize(reader);
        reader.Close();
        return value;
```

```
    }

    public static void Export(int n, string filePath, Root root)
    {
        Course[] arr = new Course[root.courses.Count];
        root.courses.CopyTo(arr);
        ///insertion sort
        for (int i = 0; i < arr.Length - 1; i++)
        {
            for (int j = i + 1; j > 0; j--)
            {
                if (arr[j - 1].units > arr[j].units)
                {
                    var temp = arr[j - 1];
                    arr[j - 1] = arr[j];
                    arr[j] = temp;
                }
            }
        }

        Array.Reverse(arr);
        Root newRoot = new Root();
        List<Course> list = new List<Course>();
        newRoot.courses = list;

        for (int i = 0; i < n; i++)
        {
            newRoot.courses.Add(arr[i]);
        }
        Serialize(filePath, newRoot);
    }

}
```

ImageProcess.cs

```
using System;
using ScottPlot;

public static class ImageProcess
{
    public static void CreateImage(string outputFile, Root root)
    {
        var subjects = GetData.GetSubjects(root);
        var plt = new ScottPlot.Plot(600, 400);
        int pointCount = 5;
        double[] xs = DataGen.Consecutive(pointCount);

        double[] ys = new double[5];
        for (int i = 0; i < 5; i++)
        {
            double units = 0;
            for (int j = 0; j < root.courses.Count; j++)
            {
                if (subjects[i] == root.courses[j].subject)
                {
                    units += root.courses[j].units;
                }
            }
            ys[i] = units;
        }
        Random rand = new Random(0);
        double[] yError = DataGen.RandomNormal(rand, pointCount, 3, 2);

        plt.PlotBar(xs, ys, yError, horizontal: true);

        plt.Grid(enableHorizontal: false, lineStyle: LineStyle.Dot);

        string[] labels = { subjects[0], subjects[1], subjects[2], subjects[3], subjects[4] };
```

```
        plt.YTicks(xs, labels);
        plt.SaveFig(outputFile);
    }
}
```

GetData.cs

```
using System;
using System.Collections.Generic;

public static class GetData
{
    public static List<string> GetSubjects(Root root)
    {
        List<string> list = new List<string>();
        for (int i = 0; i < root.courses.Count; i++)
        {
            if (list.Contains(root.courses[i].subject.ToString()) == false)
            {
                list.Add(root.courses[i].subject.ToString());
            }
        }
        return list;

    }

    public static List<string> GetTitles(Root root, string subject)
    {
        List<string> list = new List<string>();
        for (int i = 0; i < root.courses.Count; i++)
        {
            if (root.courses[i].subject == subject)
            {
                if (list.Contains(root.courses[i].course.ToString()) == false)
                {
                    list.Add(root.courses[i].course.ToString());
                }
            }
        }
        return list;
    }

    public static List<string> GetInstructors(Root root)
    {
        List<string> list = new List<string>();
        for (int i = 0; i < root.courses.Count; i++)
        {
            if (list.Contains(root.courses[i].instructor.ToString()) == false)
            {
                list.Add(root.courses[i].instructor.ToString());
            }
        }
        return list;
    }
    public static void PrintPage(Root root, int page)
    {
        Course[] arr = new Course[root.courses.Count];
        root.courses.CopyTo(arr);

        int pageSize = 5;
        int pages = GetTotalPages(arr);
        Console.WriteLine($"+----------------------------------------------------{page} / {pages}---------
-------------------------------------------+");
        for (int i = 0; i < pageSize; i++)
        {
            if (pageSize * (page - 1) + i >= arr.Length)
            {
                continue;
```

```
            }
            Console.Write("Register number: `" + arr[pageSize * (page - 1) + i].registerNumber + "` ");
            Console.Write("Subject: [" + arr[pageSize * (page - 1) + i].subject + "]; ");
            Console.Write("Course: `" + arr[pageSize * (page - 1) + i].course + "`; ");
            Console.Write("Sector: [" + arr[pageSize * (page - 1) + i].sector + "]; ");
            Console.WriteLine("Title: \"" + arr[pageSize * (page - 1) + i].title + "\" ");
            Console.Write("\tUnits: `" + arr[pageSize * (page - 1) + i].units + "`; ");
            Console.Write("Instructor: " + arr[pageSize * (page - 1) + i].instructor + "; ");
            Console.WriteLine("Days: [" + arr[pageSize * (page - 1) + i].days + "] ");
            Console.Write("\t\tStart time: `" + arr[pageSize * (page - 1) + i].time.startTime + "`; ");
            Console.WriteLine("End time: `" + arr[pageSize * (page - 1) + i].time.endTime + "` ");
            Console.Write("\t\tBuilding: [" + arr[pageSize * (page - 1) + i].place.building + "]; ");
            Console.WriteLine("Room: [" + arr[pageSize * (page - 1) + i].place.room + "] ");

Console.WriteLine("~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~");
        }

    }

    public static int GetTotalPages(Course[] arr)
    {
        const int pageSize = 5;
        return (int)Math.Ceiling(arr.Length / (double)pageSize);
    }
}
```

Приклади результатів

load {filename}

```
Enter command:
load _courses.xml
Tip: Xml was upload
```

Частина XML-файлу

_courses.xml

```xml
<?xml version='1.0' ?>
<!DOCTYPE root SYSTEM "http://www.cs.washington.edu/research/projects/xmltk/xmldata/data/courses/reed.dtd">
<root>
<course>
   <reg_num>10577</reg_num>
   <subj>ANTH</subj>
   <crse>211</crse>
   <sect>F01</sect>
   <title>Introduction to Anthropology</title>
   <units>1.0</units>
   <instructor>Brightman</instructor>
   <days>M-W</days>
   <time>
      <start_time>03:10PM</start_time>
      <end_time>04:30</end_time>
   </time>
   <place>
      <building>ELIOT</building>
      <room>414</room>
   </place>
</course>

<course>
   <reg_num>20573</reg_num>
   <subj>ANTH</subj>
```

save {filename}

```
Enter command:
save _savedCourses.xml
Tip: Xml was saved
```

Частина XML-файлу

_savedCourses.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <course>
    <reg_num>10577</reg_num>
    <subj>ANTH</subj>
    <crse>211</crse>
    <sect>F01</sect>
    <title>Introduction to Anthropology</title>
    <units>1</units>
    <instructor>Brightman</instructor>
    <days>M-W</days>
    <time>
      <start_time>03:10PM</start_time>
      <end_time>04:30</end_time>
    </time>
    <place>
      <building>ELIOT</building>
      <room>414</room>
    </place>
  </course>
  <course>
    <reg_num>20573</reg_num>
    <subj>ANTH</subj>
    <crse>344</crse>
    <sect>S01</sect>
```

export {numberOfRecords} {filename}

```
Enter command:
export 1 _export.xml
Tip: Data was exported
```

Вигляд XML-файлу

_export.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<root xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <course>
    <reg_num>10776</reg_num>
    <subj>BIOL</subj>
    <crse>388</crse>
    <sect>F</sect>
    <title>Evolutionary Biology</title>
    <units>1</units>
    <instructor>Raubeson</instructor>
    <days>M-W-F</days>
    <time>
      <start_time>09:00AM</start_time>
      <end_time>09:50</end_time>
    </time>
    <place>
      <building>PHYSIC</building>
      <room>240A</room>
    </place>
  </course>
</root>
```

print {pageNumber}

```
Enter command:
print 2
+-----------------------------------------------------2 / 141-----------------------------------------------------+
Register number: `10543` Subject: [CHEM]; Course: `101`; Sector: [F]; Title: "MolecularStructure and Properties"
       Units: `1`; Instructor: Geselbracht; Days: [M-W-F]
              Start time: `11:00AM`; End time: `11:50`
              Building: [VOLLUM]; Room: [VLH]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Register number: `10544` Subject: [CHEM]; Course: `101`; Sector: [F01]; Title: "MolecularStructure and Properties"
       Units: `0`; Instructor: Geselbracht; Days: [M]
              Start time: `01:10PM`; End time: `02:00`
              Building: [CHEM]; Room: [301]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Register number: `10545` Subject: [CHEM]; Course: `101`; Sector: [F02]; Title: "MolecularStructure and Properties"
       Units: `0`; Instructor: Geselbracht; Days: [M]
              Start time: `02:10PM`; End time: `03:00`
              Building: [CHEM]; Room: [301]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Register number: `10546` Subject: [CHEM]; Course: `101`; Sector: [F03]; Title: "MolecularStructure and Properties"
       Units: `0`; Instructor: Geselbracht; Days: [M]
              Start time: `03:10PM`; End time: `04:00`
              Building: [CHEM]; Room: [301]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Register number: `10789` Subject: [CHEM]; Course: `101`; Sector: [F08]; Title: "MolecularStructure and Properties"
       Units: `0`; Instructor: Geselbracht; Days: [T]
              Start time: `11:00AM`; End time: `11:50`
              Building: [PHYSIC]; Room: [123]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

subjects

```
Enter command:
subjects
[ANTH]   [BIOL]  [CHEM]  [CHIN]  [CLAS]  [CRWR]  [DANC]  [ECON]
[ENG]    [ART]   [FREN]  [GER]   [GRK]   [HIST]  [HUM]   [LAT]
[LING]   [LIT]   [MATH]  [MUS]   [PE]    [PHIL]  [PHYS]  [POL]
[PSY]    [REL]   [RUSS]  [SOC]   [SPAN]  [THEA]  [LBST]
```

intructors

```
Enter command:
instructors
Brightman     Makley        Kaplan                Yezerinac     Geselbracht   Haviland      Glasfeld
Silverstein   Shusterman    McDougal      Stasch  Dunne         McClard       Ahmadi        Wu
Rhew          Englert       Kelly         Hillis  Williams      Mann          Wong          Clausing
Netusil       Parker        Despins       Arnold  Leveen        Duquette      Gillcrist     King
Baker         Steinman      Stauder       Savery  Porter        Sherman       Knapp         Knutson
Delehanty     Berkvam       Ondrizek      Khan    Hochman       Gokberk       Mieszkowski   Irmscher
Rudolf        Wolfe         Tron          Nicholson Fix         Dirks         McCalla       Segel
Diebold       Sacks         Mueller       Smith   Breen         Iaccarino     Garrett       Drumm
Arkonovich    Feener        Foat          Mandell David         Latham        Hendrickson   Kierstead
van Dyke      Brashier      Canseco-Gonzale Lencek  Bershtein     Mayer         Shurman       Potluri
Perkinson     Jones         Wieting       Bernhard Fillin-Yeh   Schiff        Hancock       Falk
Casey         Shampay       Hinchliff     Reeve   Taschek       Peck          Bonfim        Reynolds
Staff         Griffiths     Black         James   Mitchell      Essick        Wheeler       Crandall
Rejali        Gronke        Steinberger   Andreas Tudor         Teske         Kapsch        Reisberg
Neuringer     Oleson        Herman        Bulkley Dickinson     Rhodes        Raubeson      Hrycak
Schneiberg    Moret         Schulz        Fischer Alonso        Larisch       Garcia-Bryce  Worley
Carr          Muller        Clinton       Lowe    Mellies       Russell       McClellan     Arch
```

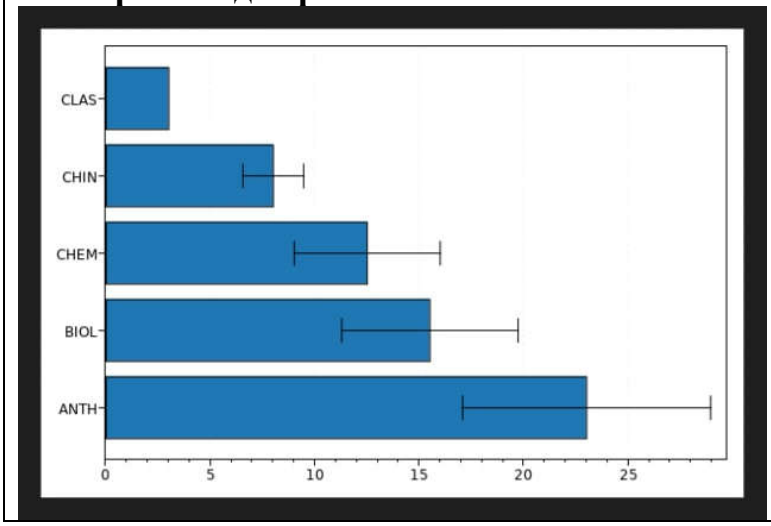
subjects {subjectName}

```
Enter command:
subject BIOL
[431]    [101]    [102]    [332]    [342]    [351]    [356]    [358]
[361]    [366]    [367]    [372]    [381]    [388]
```

image {filename}

```
Enter command:
image Plot.png
Tip: Diagram was saved
```

PNG-файл із діаграмою:



exit

```
Enter command:
exit
Ending processing...
```

Висновки

Виконавши дану лабораторну роботу було вивчено одна із мов розмітки - XML. XML(англ. Extensible Markup Language) - мова розмітки, що базується на ієрархічних зв'язках. Програмна модель XML є деревом, кожен елемент має ім’я, дітей і є словником атрибутів (рядок-рядок).

Лінійні колекції подаються як дерево із коренем (колекція) і дітьми (елементами колекції).

Також було використано пакет для програмної генерації графіків - ScottPlot. ScottPlot - це безкоштовна бібліотека для побудови графіків із відкритим кодом для .NET, яка полегшує інтерактивне відображення великих наборів даних. Лінійні графіки, гістограми, кругові діаграми, діаграми розсіювання та багато іншого можна створити лише за допомогою декількох рядків коду.

Компіляція всього коду відбувалася за допомогою утиліти dotnet.