

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(ла)  студент(ка) I курсу  групи КП-02  Жученко Андрій Сергійович  (*прізвище, ім’я, по батькові*) |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2020

# Мета

Навчитись виконувати серіалізацію і десеріалізацію даних у форматі XML.

Виконати генерацію зображення з графіком на основі вхідних даних.

# 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}** - серіалізувати частину даних у XML файл за варіантом (з відступами).
* *три команди на отримання даних за варіантом.*Отримані дані виводити в консоль.
* **image {filename}** - створити і зберегти зображення з графіком за варіантом у файл.

Варіант файлу з вхідними даними та операції над даними у [**Додатку А**](https://docs.google.com/document/d/1nHM-ucXbz_Gtsqs3GRFlvXvXBWbYYh_AxOhkmu085Uo/edit#heading=h.cclohgmfifz0).

# Методичні вказівки

При вирішенні цього завдання використовувати стандартні загальні колекції даних (наприклад, List<T>, Dictionary<T1, T1>, HashTable<T>, тощо). Підібрати такі колекції, що найкраще підходять для алгоритмів, що розробляються. **LINQ використовувати заборонено**.

Створити класи для десеріалізації даних з заданого XML документа у C# об’єкти та серіалізації об’єктів у XML. При цьому слідувати правилам найменування типів і полів типів в C#. Давати полям таких класів типи, що найкраще їм підходять (мінімізувати використання **string** там, де це можливо). Зі скорочених назв типів і полів зробити повні назви (наприклад, bldg -> building). Для контролю іменування елементів у XML використати в класах спеціальні C# атрибути (наприклад, XmlType, XmlArray, XmlArrayItem, тощо).

Розбити код на модулі:

* **Командного інтерфейсу користувача** - зчитування команд користувача і вивід результатів у консоль (пагінація, форматування, тощо), зчитування і запис даних у файли.
* **Серіалізації і десеріалізації даних у XML** - на основі System.Xml.
* **Обробки даних** - з використанням стандартних загальних колекцій даних з System.Collections.Generic.
* **Генерації зображення** - для його реалізації використати [пакет ScottPlot](https://swharden.com/scottplot/cookbooks/4.0.47/), або будь-який аналог.

За бажання, можна додати інші модулі та/або створити власні бібліотеки класів.

## Додаток А. XML дані і варіант

**n** - номер в списку групи - 4

Варіант: **n % 3**

|  |  |  |
| --- | --- | --- |
| 1 | [Курси WSU](http://aiweb.cs.washington.edu/research/projects/xmltk/xmldata/data/courses/wsu.xml) | Зберегти у XML перші N курсів з найвищою кількістю зарахованих (enrolled).  Отримання даних:   1. **prefixes** - Список назв всіх унікальних предметів (prefix) 2. **titles {bldg}** Список назв (title) всіх курсів, що проводяться у заданому будинку (за назвою, place.bldg). 3. **instructors** - Список всіх унікальних викладачів (instructor)   Графік: відсортувати курси по значенню enrolled і вивести на графіку limit та enrolled. Якщо курсів забагато - обмежити їх кількість на графіку певним значенням.  Тип графіка: **stacked bars** показує заповненість кожного курсу студентами, верхнє значення - limit (кількість місць), нижнє - enrolled (кількість студентів). |

**Тексти коду програм**

|  |
| --- |
| **./Program.cs** |
| using static System.Console;  using System;  using System.IO;  using System.Collections.Generic;  namespace lab5\_2  {  class Program  {  static void Main(string[] args)  {  WriteLine("Command:");  string command = ReadLine();  if(command.StartsWith("load "))  {  Root root = GetCoursesFromFile(command);  while(true)  {  WriteLine("Command:");  string command1 = ReadLine();  if(command1.StartsWith("save "))  {  ProcessCommandSave(command1, root);  }  else if(command1 == "exit")  {  break;  }  else if(command1 == "prefixes")  {  ProcessPrefixes(root);  }  else if(command1 == "instructors")  {  ProcessInstructors(root);  }  else if(command1.StartsWith("titles "))  {  ProcessTitles(root, command1);  }  else if(command1.StartsWith("print "))  {  ProcessCommandPrint(root, command1);  }  else if(command1.StartsWith("export "))  {  ProcessCommandExport(command1, root);  }  else if(command1.StartsWith("image "))  {  ProcessCommandImage(root, command1);  }  else  {  WriteLine("Invalid command");  }  }  }  else  {  WriteLine("Invalid command");  }  }  static Root GetCoursesFromFile(string command)  {  string[] values = command.Split(' ');  if(values.Length != 2)  {  throw new Exception("Wrong name of file");  }  try  {  StreamReader reader = new StreamReader(values[1]);  }  catch  {  throw new Exception("Wrong name of file");  }  Root root = XMLProcessor.DeserializeCourses(values[1]);  return root;  }  static void ProcessTitles(Root root, string command)  {  string[] values = command.Split(' ');  if(values.Length != 2)  {  throw new Exception("Wrong name of file");  }  List<string> titles = new List<string>();  foreach(Course course in root.courses)  {  if(course.place.bilding == null)  {  continue;  }  else if(course.place.bilding == values[1])  {  titles.Add(course.title);  }  }  foreach(string title in titles)  {  WriteLine(title);  }  }  static void ProcessCommandSave(string command, Root root)  {  string[] values = command.Split(' ');  if(values.Length != 2)  {  throw new Exception("Wrong name of file");  }  XMLProcessor.SerializeCourses(root, values[1]);  }  static void ProcessPrefixes(Root root)  {  HashSet<string> prefixes = new HashSet<string>();  foreach(Course course in root.courses)  {  prefixes.Add(course.prefix);  }  foreach(string prefix in prefixes)  {  Console.WriteLine(prefix);  }  }  static void ProcessInstructors(Root root)  {  HashSet<string> instructors = new HashSet<string>();  foreach(Course course in root.courses)  {  instructors.Add(course.instructor);  }  foreach(string prefix in instructors)  {  Console.WriteLine(prefix);  }  }  static void ProcessCommandPrint(Root root, string command)  {  string[] values = command.Split(' ');  if(values.Length != 2)  {  throw new Exception("Wrong command");  }  int page;  bool check = int.TryParse(values[1], out page);  if(check == false)  {  throw new Exception("Wrong page number");  }  if(page > Math.Ceiling((double)root.courses.Count / 10.0))  {  throw new Exception("Wrong page number");  }  string[] array = DataProcessor.GetPages(page, root);  WriteLine($"Total pages: {Math.Ceiling((double)root.courses.Count / 10.0)}");  for(int i = 0; i < 10; i++)  {  WriteLine(array[i]);  }  }  static void ProcessCommandExport(string command, Root root)  {  string[] values = command.Split(' ');  if(values.Length != 3)  {  throw new Exception("Wrong command");  }  int n;  bool check = int.TryParse(values[1], out n);  if(check == false)  {  throw new Exception("Wrong page number");  }  Root sortedroot = DataProcessor.GetSortedByEnroll(root);  Course[] arrayForExport = new Course[n];  for(int i = 0; i < n; i++)  {  arrayForExport[i] = sortedroot.courses[i];  XMLProcessor.SerializeSortedArray(arrayForExport, values[2]);  }  XMLProcessor.SerializeSortedArray(arrayForExport, values[2]);  }  static void ProcessCommandImage(Root root, string command)  {  string[] values = command.Split(' ');  if(values.Length != 2)  {  throw new Exception("Wrong command");  }  Root sortedroot = DataProcessor.GetSortedByEnroll(root);  Course[] arrayForImage = new Course[10];  for(int i = 0; i < 10; i++)  {  arrayForImage[i] = sortedroot.courses[i];  }  ImageProcessor.DrawImage(arrayForImage, values[1]);  }  }  } |

|  |
| --- |
| ImageProcessor.cs |
| namespace lab5\_2  {  public static class ImageProcessor  {  public static void DrawImage(Course[] courses, string path)  {  ScottPlot.Plot plt = new ScottPlot.Plot(400, 300);  double[] xs = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  double[] valuesB = new double[10];  for(int i = 0; i < 10; i++)  {  valuesB[i] = (double)courses[i].limit;  }  double[] valuesA = new double[10];  for(int i = 0; i < 10; i++)  {  valuesA[i] = (double)courses[i].enrolled;  }  plt.PlotBar(xs, valuesB, label: "Series B");  plt.PlotBar(xs, valuesA, label: "Series A");  plt.Title("Stacked Bar Charts");  plt.SaveFig(path);  }  }  } |

|  |
| --- |
| ./XMLProcessor.cs |
| using System.Collections.Generic;  using System.Xml.Serialization;  using System.IO;  namespace lab5\_2  {  public static class XMLProcessor  {  public static Root DeserializeCourses(string path)  {  XmlSerializer ser = new XmlSerializer(typeof(Root));  StreamReader reader = new StreamReader(path);  Root courses = (Root)ser.Deserialize(reader);  reader.Close();  return courses;  }  public static void SerializeCourses(Root root, string path)  {  XmlSerializer ser = new XmlSerializer(typeof(Root));  StreamWriter writer = new StreamWriter(path);  ser.Serialize(writer, root);  writer.Close();  }  public static void SerializeSortedArray(Course[] array, string path)  {  XmlSerializer ser = new XmlSerializer(typeof(Course[]));  StreamWriter writer = new StreamWriter(path);  ser.Serialize(writer, array);  writer.Close();  }  }  } |

|  |
| --- |
| ./DataProcessor.cs |
| namespace lab5\_2  {  public static class DataProcessor  {  public static string[] GetPages(int numberOfPage, Root root)  {  string[] array = new string[10];  int counter = 0;  for(int i = (numberOfPage \* 10 - 10); i < numberOfPage \* 10; i++)  {  array[counter] = root.courses[i].ConvertToSting();  counter++;  }  return array;  }  public static Root GetSortedByEnroll(Root root)  {  root.courses.Sort(CompareTwoCourses);  return root;  }  private static int CompareTwoCourses(Course x, Course y)  {  if(x.enrolled == y.enrolled)  {  return 0;  }  else if(x.enrolled > y.enrolled)  {  return -1;  }  else  {  return 1;  }  }  }  } |

|  |
| --- |
| ./Root.cs |
| using System.Collections.Generic;  using System.Xml.Serialization;  namespace lab5\_2  {  [XmlRoot("root")]  public class Root  {  [XmlElement("course")]  public List<Course> courses;  }  } |

|  |
| --- |
| ./Course.cs |
| using System.Xml.Serialization;  namespace lab5\_2  {  public class Course  {  public string footnote;  public string sln; // I have no idea what is the decryption of this abbreviation:(  public string prefix;  public int crs; // I have no idea what is the decryption of this abbreviation:(  public string lab;  [XmlElement("sect")]  public int section;  public string title;  public string credit;  public string days;  public Times times;  public Place place;  public string instructor;  public int limit;  public int enrolled;  public string ConvertToSting()  {  return $"Title: {this.title}, section: {this.section}, enrolled: {this.enrolled}";  }  }  } |

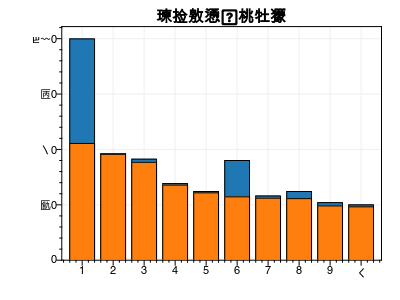
|  |
| --- |
| ./Times.cs |
| namespace lab5\_2  {  public class Times  {  public string start;  public string end;  }  } |

|  |
| --- |
| ./Place.cs |
| using System.Xml.Serialization;  namespace lab5\_2  {  public class Place  {  [XmlElement("bldg")]  public string bilding;  public string room;  }  } |

# 

# Результати роботи програми

**Графік:**

****

**Частина вхідного XML-файлу:**

**<?xml version='1.0' ?>**

**<!DOCTYPE root SYSTEM "http://www.cs.washington.edu/research/projects/xmltk/xmldata/data/courses/wsu.dtd">**

**<root>**

**<course>**

**<footnote></footnote>**

**<sln>10637</sln>**

**<prefix>ACCTG</prefix>**

**<crs>230</crs>**

**<lab></lab>**

**<sect>01</sect>**

**<title>INT FIN ACCT</title>**

**<credit>3.0</credit>**

**<days>TU,TH</days>**

**<times>**

**<start>7:45</start>**

**<end>9</end>**

**</times>**

**<place>**

**<bldg>TODD</bldg>**

**<room>230</room>**

**</place>**

**<instructor>B. MCELDOWNEY</instructor>**

**<limit>0112</limit>**

**<enrolled>0108</enrolled>**

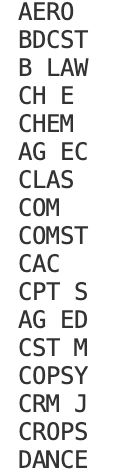
**</course>**

**Дерево:**

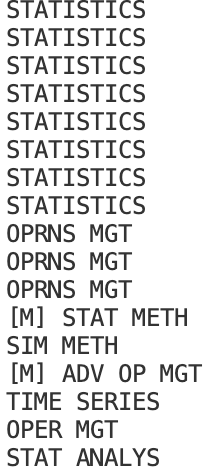
****

**Приклади виводу для деяких команд:**

Частина виводу для команди “prefixes”



Частина виводу для команди “titles TODD”



Перший за кількістю enrolled курс:

<Course>

<footnote />

<sln>27425</sln>

<prefix>ENGL</prefix>

<crs>102</crs>

<lab />

<sect>1</sect>

<title>WRT TUTORIAL</title>

<credit>V</credit>

<days />

<times>

<start />

<end />

</times>

<place>

<bldg>ARR</bldg>

<room>ARR</room>

</place>

<instructor />

<limit>999</limit>

<enrolled>527</enrolled>

</Course>

# Висновки

В результаті виконання цієї лабораторної роботи була виконана серіалізація і десеріалізація даних у форматі XML.

Виконана генерація зображення з графіком на основі вхідних даних.