

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 №** 1

з дисципліни “Основи програмування”

тема “Об'єкти і класи”

|  |  |  |
| --- | --- | --- |
| Виконав  студент I курсу  групи КП-01  Беліцький Олександр Сергійович  (*прізвище, ім’я, по батькові*)  варіант № 3 |  | Перевірив  “\_\_\_\_” “\_\_\_\_\_\_\_\_\_\_\_\_” 20\_\_\_ р.  викладач  Гадиняк Руслан Анатолійович  (*прізвище, ім’я, по батькові*) |

Київ 2021

**Мета роботи**

Створення і використання класів для об'єктів з даними без логіки та об'єктів динамічних колекцій елементів.

Генерування та порядкова обробка великої кількості даних у форматі CSV.

**Постановка завдання**

**Частина 1. Генератор CSV**

Створити консольну програму, що на основі двох заданих аргументів командного рядка генерує у текстовий файл дані у форматі CSV.

Перший аргумент задає шлях до файлу, другий - кількість рядків у CSV таблиці з даними, які будуть згенеровані. Якщо не задано одного з аргументів, або у аргументах є помилка - друкувати про це повідомлення в консолі і завершувати роботу програми. До аргументу з назвою файлу не додавати в програмі жодних додаткових рядків, не робити обмеження лише на `.csv` розширення, воно може бути будь-яким і бути відсутнім. Кількість рядків даних - невід’ємне число.

Дані, які генеруються у файл описують сутності із лабораторної роботи №5 минулого семестру. CSV файл має містити перший рядок із назвами стовпців (не входить в кількість рядків, які потрібно генерувати). Ідентифікатори генерувати унікальними в рамках файлу. Для генерації рядкових даних можна задати масив з рядками і випадковим чином вибирати звідти рядки, їх можна склеювати.

Достатньо зберігати дані без CSV екранування, якщо вони будуть генеруватись з відповідними обмеженнями на дані.

**Частина 2.**

Створити консольну програму, що зчитує CSV дані заданих за варіантом сутностей з 2-х файлів, виконує їх перетворення і записує результат як CSV у 3-ій файл.

Попередньо згенерувати вхідні файли за допомогою програми-генератора з першої частини завдання. Перший з файлів має містити близько 10-20 сутностей, другий - 100000-200000 сутностей.

Шляхи до 3-х файлів достатньо задати в коді (але 1 раз в головній функції) без аргументів командного рядка.

Перенести у код роботи структуру даних із лабораторної роботи №5 і зробити її класом. Додати у клас два конструктора: без параметрів і з параметрами для всіх полів. Одне із полів має бути цілочисельним і з назвою **id** - ідентифікатор об’єкта. Додати в клас реалізацію стандартного метода ToString (див. додаток).

Створити клас (ListEntity, Entity замінити на назву типу сутності) для списку сутностей за варіантом на основі масиву об’єктів .

Реалізувати функцію (замінити entity i entities на назву сутності в однині і множині, за варіантом):

|  |
| --- |
| static ListEntity ReadAllEntities(string filePath); |

У цій функції зчитувати текст з CSV файлу **порядково** (див додатки), у об’єкт власної реалізації списка із об’єктами типу сутності за варіантом. **Файл можна зчитати лише один раз**.

Перевіряти рядки CSV, якщо у них помилка (наприклад, кількість даних у рядку не рівна кількості стовпців) викидати помилку, що завершить роботу програми.

Використати ReadAllEntities для обох вхідних файлів і вивести кількість зчитаних рядків даних і перші 10 рядків даних (якщо є) у консоль для кожного файлу окремо.

Дії (створити на кожну по функції):

* Створити новий об’єкт ListEntity і переписати в нього всі елементи з перших двох списків так, щоби в новому списку не повторювались ідентифікатори сутностей (всі ідентифікатори були унікальні). Тобто, не переписувати у новий список сутність, якщо у списку вже є інша сутність з таким ідентифікатором. (MergeLists)
* Обрати будь-яке числове поле вашого типу сутності (але не ідентифікатор) і знайти по списку середнє арифметичне значення по цьому полю. (AverageAge)
* Видалити зі списку всі об’єкти, значення відповідного поля яких менше за знайдене середнє арифметичне. (DeleteUnderAverage)

Створити функцію (замінити entity i entities на назву сутності в однині і множині, за варіантом):

|  |
| --- |
| static void WriteAllEntities(string filePath, ListEntity entities); |

Записати результат (модифікований третій список) у вихідний файл в форматі CSV **порядково** (див додатки).

**Аналіз вимог і проектування**

**Частина 1.**

Клас Teacher (лабораторна робота №5 минулий семестр - структура Teacher)

|  |
| --- |
| class Teacher  {  public int id;  public string fullname;  public string subject;  public int age;  public Teacher()  {  id = 0;  fullname = "";  subject = "";  age = 0;  }  public Teacher(int id, string fullname, string subject, int age)  {  this.id = id;  this.fullname = fullname;  this.subject = subject;  this.age = age;  }  public override string ToString()  {  return $"{id,-8} {fullname,20} - {subject,-10} | {age,3}";  }  } |

**Частина 2.**

Клас Teacher (взятий із попередньої частини)

Клас ListTeacher

|  |
| --- |
| class ListTeachers  {  private Teacher[] \_items;  private int \_size;  public ListTeachers()  {  \_items = new Teacher[16];  \_size = 0;  }  public ListTeachers ReadAllTeachers(string filePath)  {  ListTeachers teacher = new ListTeachers();  StreamReader sr = new StreamReader(filePath);  string s = "";  while (true)  {  s = sr.ReadLine();  if (s == null)  {  break;  }  string[] str = s.Split(',');  if (str[0] == "id")  {  continue;  }  else  {  if (str.Length != 4)  {  WriteLine("Error: csv file has a problem with data");  Environment.Exit(0);  }  else  {  int str0, str3;  if (int.TryParse(str[0], out str0))  {  if (str0 <= 0)  {  WriteLine("Error: Id must be a positive number");  Environment.Exit(0);  }  if (int.TryParse(str[3], out str3))  {  if (str3 <= 0)  {  WriteLine("Error: Age must be a positive number");  Environment.Exit(0);  }  teacher.Add(new Teacher(str0, str[1], str[2], str3));  }  else  {  WriteLine("Error: Age must be a number");  Environment.Exit(0);  }  }  else  {  WriteLine("Error: Id must be a number");  Environment.Exit(0);  }  }  }  }  sr.Close();  return teacher;  }  public void PrintFirst10()  {  for (int i = 0; i < 10; i++)  {  WriteLine(\_items[i].ToString());  }  }  public void Add(Teacher newTeacher)  {  if (this.\_size == this.\_items.Length)  {  Expand();  }  this.\_items[this.\_size] = newTeacher;  this.\_size += 1;  }  private void Expand()  {  int oldCapacity = this.\_items.Length;  Teacher[] oldArray = this.\_items;  this.\_items = new Teacher[oldCapacity \* 2];  System.Array.Copy(oldArray, this.\_items, oldCapacity);  }  public void Insert(int index, Teacher teacher)  {  if ((index > (\_size)) || (index < 0))  {  WriteLine("Error: Index does not exist");  Environment.Exit(0);  }  if (this.\_size == this.\_items.Length)  {  Expand();  }  for (int i = \_size; i >= index; i--)  {  \_items[i] = \_items[i - 1];  }  \_items[index] = teacher;  \_size += 1;  }  public bool Remove(Teacher teacher)  {  for (int i = 0; i <= \_size; i++)  {  if (\_items[i] == teacher)  {  \_size -= 1;  for (int j = i; j < \_size; j++)  {  \_items[j] = \_items[j + 1];  }  return true;  }  }  return false;  }  public int GetCount()  {  return \_size;  }  public int GetCapacity()  {  return \_items.Length;  }  public Teacher GetAt(int index)  {  if ((index > (\_size - 1)) || (index < 0))  {  WriteLine("Error: Teacher under this index does not exist");  Environment.Exit(0);  }  return \_items[index];  }  public void SetAt(int index, Teacher teacher)  {  if ((index > (\_size - 1)) || (index < 0))  {  WriteLine("Error: Teacher under this index does not exist");  Environment.Exit(0);  }  \_items[index] = teacher;  }  public ListTeachers MergeLists(ListTeachers list1, ListTeachers list2)  {  ListTeachers resultList = new ListTeachers();  if (list1.\_items.Length > list2.\_items.Length)  {  var buf = list1;  list1 = list2;  list2 = buf;  }  resultList = list2;  for (int i = 0; i < list1.\_size; i++)  {  for (int j = 0; j < resultList.\_size; j++)  {  if (list1.\_items[i].id == resultList.\_items[j].id)  {  resultList.\_items[j] = list1.\_items[i];  break;  }  else if (j == resultList.\_size - 1)  {  resultList.Add(list1.\_items[i]);  }  }  }  return resultList;  }  public double AverageAge(ListTeachers list)  {  int sum = 0;  for (int i = 0; i < list.\_size; i++)  {  sum += list.\_items[i].age;  }  double avg = sum / list.\_size;  return avg;  }  public ListTeachers DeleteUnderAverage(ListTeachers list, double avg)  {  for (int i = 0; i < list.\_size; i++)  {  while (true)  {  if (list.\_items[i].age < avg)  {  list.Remove(list.\_items[i]);  }  else  {  break;  }  }  }  return list;  }  public void WriteAllTeachers(string filePath, ListTeachers list)  {  StreamWriter sw = new StreamWriter(filePath);  string s = "";  for(int i = 0; i < list.\_size; i++)  {  string[] str = {list.\_items[i].id.ToString(),list.\_items[i].fullname,list.\_items[i].subject, list.\_items[i].age.ToString() };  s = string.Join(',',str);  sw.WriteLine(s);  }  sw.Close();  }  } |

Реалізація функції ReadAllEntities:

|  |
| --- |
| public ListTeachers ReadAllTeachers(string filePath)  {  ListTeachers teacher = new ListTeachers();  StreamReader sr = new StreamReader(filePath);  string s = "";  while (true)  {  s = sr.ReadLine();  if (s == null)  {  break;  }  string[] str = s.Split(',');  if (str[0] == "id")  {  continue;  }  else  {  if (str.Length != 4)  {  WriteLine("Error: csv file has a problem with data");  Environment.Exit(0);  }  else  {  int str0, str3;  if (int.TryParse(str[0], out str0))  {  if (str0 <= 0)  {  WriteLine("Error: Id must be a positive number");  Environment.Exit(0);  }  if (int.TryParse(str[3], out str3))  {  if (str3 <= 0)  {  WriteLine("Error: Age must be a positive number");  Environment.Exit(0);  }  teacher.Add(new Teacher(str0, str[1], str[2], str3));  }  else  {  WriteLine("Error: Age must be a number");  Environment.Exit(0);  }  }  else  {  WriteLine("Error: Id must be a number");  Environment.Exit(0);  }  }  }  }  sr.Close();  return teacher;  } |

Реалізація функції MergeLists

|  |
| --- |
| public ListTeachers MergeLists(ListTeachers list1, ListTeachers list2)  {  ListTeachers resultList = new ListTeachers();  if (list1.\_items.Length > list2.\_items.Length)  {  var buf = list1;  list1 = list2;  list2 = buf;  }  resultList = list2;  for (int i = 0; i < list1.\_size; i++)  {  for (int j = 0; j < resultList.\_size; j++)  {  if (list1.\_items[i].id == resultList.\_items[j].id)  {  resultList.\_items[j] = list1.\_items[i];  break;  }  else if (j == resultList.\_size - 1)  {  resultList.Add(list1.\_items[i]);  }  }  }  return resultList;  } |

Реалізація функції WriteAllEntities

|  |
| --- |
| public void WriteAllTeachers(string filePath, ListTeachers list)  {  StreamWriter sw = new StreamWriter(filePath);  string s = "";  for(int i = 0; i < list.\_size; i++)  {  string[] str = {list.\_items[i].id.ToString(),list.\_items[i].fullname,list.\_items[i].subject, list.\_items[i].age.ToString() };  s = string.Join(',',str);  sw.WriteLine(s);  }  sw.Close();  } |

Реалізація функції AverageAge

|  |
| --- |
| public double AverageAge(ListTeachers list)  {  int sum = 0;  for (int i = 0; i < list.\_size; i++)  {  sum += list.\_items[i].age;  }  double avg = sum / list.\_size;  return avg;  } |

Реалізація функції DeleteUnderAverage

|  |
| --- |
| public ListTeachers DeleteUnderAverage(ListTeachers list, double avg)  {  for (int i = 0; i < list.\_size; i++)  {  while (true)  {  if (list.\_items[i].age < avg)  {  list.Remove(list.\_items[i]);  }  else  {  break;  }  }  }  return list;  } |

**Обмеження до двох частин**

При реалізації завдань заборонено:

* використання стандартних типів колекцій і інших алгоритмів, що дані у завданні, їх поведінку необхідно реалізувати самостійно.
* використання статичних полів.

Функції мають мати одне призначення, без зайвих параметрів, повертати результат там, де це можливо.

Назви функцій, типів, полів і змінних мають слідувати одному стилю і відповідати їх призначенню.

Назва функції (метода) має містити хоча б одне дієслово, що описує дію, яку вона виконує.

**Тексти коду програм**

**Частина 1.**

|  |
| --- |
| **Program.cs** |
| using System;  using System.Text;  using System.IO;  using System.Diagnostics;  using static System.Console;  namespace part\_1  {  class Program  {  static void Main(string[] args)  {  CheckArgs(args);  if (File.Exists($"{args[0]}"))  {  Stopwatch sw = new Stopwatch();  sw.Start();  File.WriteAllText($"{args[0]}", String.Empty);  File.WriteAllText($"{args[0]}", GenerateFile((GiveInt(args))));  sw.Stop();  WriteLine($"Tip: Specified file was rewrited");  WriteLine(sw.Elapsed);  }  else  {  Stopwatch sw = new Stopwatch();  sw.Start();  FileStream flstream = new FileStream($"{args[0]}", FileMode.OpenOrCreate);  File.WriteAllText($"{args[0]}", GenerateFile((GiveInt(args))));  sw.Stop();  WriteLine("Tip: New file was created");  WriteLine(sw.Elapsed);  }  }  static void CheckArgs(string[] args)  {  if (args.Length != 2)  {  WriteLine("Error: Row arguments length");  Environment.Exit(0);  }  else if ((args[0][0] != '.') && (args[0][1] != '/'))  {  WriteLine("Error: Enter a relative path to the file");  Environment.Exit(0);  }  GiveInt(args);  }  static int GiveInt(string[] args)  {  int number;  if (int.TryParse(args[1], out number))  {  if (number <= 0)  {  WriteLine("Error: Enter positive integer number");  Environment.Exit(0);  }  return number;  }  else  {  WriteLine("Error: Second argument must be a number");  Environment.Exit(0);  }  return number;  }  static string GenerateFile(int strings)  {  StringBuilder sb1 = new StringBuilder();  string[] fullname = {"Prokhorov Ludwig", "Rusakov Yuri","Stepanov Gordey","Ilyin Mechislav","Lobanov Ernest",  "Bespalov Mitrofan","Boris Kulikov","Veselov Anton","Sukhanov Arsen","Komissarov Klim"};  string[] subject = { "Algebra", "Art", "Biology", "Chemistry", "English", "Geography", "Geometry", "Health", "History", "PE" };  int[] age = { 33, 23, 45, 66, 34, 78 };  sb1.Append("id").Append(',').Append("fullname").Append(',').Append("subject").Append(',').Append("age");  for (int i = 1; i < strings; i++)  {  sb1.Append("\r\n").Append(i).Append(',').Append(fullname[new Random().Next(0, fullname.Length)]).Append(',')  .Append(subject[new Random().Next(0, subject.Length)]).Append(',').Append(age[new Random().Next(0, age.Length)]);  }  return sb1.ToString();  }  }  } |

**Частина 2.**

|  |
| --- |
| **Program.cs** |
| using System;  using System.Diagnostics;  using System.IO;  using static System.Console;  namespace part\_2  {  class Teacher  {  public int id;  public string fullname;  public string subject;  public int age;  public Teacher()  {  id = 0;  fullname = "";  subject = "";  age = 0;  }  public Teacher(int id, string fullname, string subject, int age)  {  this.id = id;  this.fullname = fullname;  this.subject = subject;  this.age = age;  }  public override string ToString()  {  return $"{id,-8} {fullname,20} - {subject,-10} | {age,3}";  }  }  class ListTeachers  {  private Teacher[] \_items;  private int \_size;  public ListTeachers()  {  \_items = new Teacher[16];  \_size = 0;  }  public ListTeachers ReadAllTeachers(string filePath)  {  ListTeachers teacher = new ListTeachers();  StreamReader sr = new StreamReader(filePath);  string s = "";  while (true)  {  s = sr.ReadLine();  if (s == null)  {  break;  }  string[] str = s.Split(',');  if (str[0] == "id")  {  continue;  }  else  {  if (str.Length != 4)  {  WriteLine("Error: csv file has a problem with data");  Environment.Exit(0);  }  else  {  int str0, str3;  if (int.TryParse(str[0], out str0))  {  if (str0 <= 0)  {  WriteLine("Error: Id must be a positive number");  Environment.Exit(0);  }  if (int.TryParse(str[3], out str3))  {  if (str3 <= 0)  {  WriteLine("Error: Age must be a positive number");  Environment.Exit(0);  }  teacher.Add(new Teacher(str0, str[1], str[2], str3));  }  else  {  WriteLine("Error: Age must be a number");  Environment.Exit(0);  }  }  else  {  WriteLine("Error: Id must be a number");  Environment.Exit(0);  }  }  }  }  sr.Close();  return teacher;  }  public void PrintFirst10()  {  for (int i = 0; i < 10; i++)  {  WriteLine(\_items[i].ToString());  }  }  public void Add(Teacher newTeacher)  {  if (this.\_size == this.\_items.Length)  {  Expand();  }  this.\_items[this.\_size] = newTeacher;  this.\_size += 1;  }  private void Expand()  {  int oldCapacity = this.\_items.Length;  Teacher[] oldArray = this.\_items;  this.\_items = new Teacher[oldCapacity \* 2];  System.Array.Copy(oldArray, this.\_items, oldCapacity);  }  public void Insert(int index, Teacher teacher)  {  if ((index > (\_size)) || (index < 0))  {  WriteLine("Error: Index does not exist");  Environment.Exit(0);  }  if (this.\_size == this.\_items.Length)  {  Expand();  }  for (int i = \_size; i >= index; i--)  {  \_items[i] = \_items[i - 1];  }  \_items[index] = teacher;  \_size += 1;  }  public bool Remove(Teacher teacher)  {  for (int i = 0; i <= \_size; i++)  {  if (\_items[i] == teacher)  {  \_size -= 1;  for (int j = i; j < \_size; j++)  {  \_items[j] = \_items[j + 1];  }  return true;  }  }  return false;  }  public int GetCount()  {  return \_size;  }  public int GetCapacity()  {  return \_items.Length;  }  public Teacher GetAt(int index)  {  if ((index > (\_size - 1)) || (index < 0))  {  WriteLine("Error: Teacher under this index does not exist");  Environment.Exit(0);  }  return \_items[index];  }  public void SetAt(int index, Teacher teacher)  {  if ((index > (\_size - 1)) || (index < 0))  {  WriteLine("Error: Teacher under this index does not exist");  Environment.Exit(0);  }  \_items[index] = teacher;  }  public ListTeachers MergeLists(ListTeachers list1, ListTeachers list2)  {  ListTeachers resultList = new ListTeachers();  if (list1.\_items.Length > list2.\_items.Length)  {  var buf = list1;  list1 = list2;  list2 = buf;  }  resultList = list2;  for (int i = 0; i < list1.\_size; i++)  {  for (int j = 0; j < resultList.\_size; j++)  {  if (list1.\_items[i].id == resultList.\_items[j].id)  {  resultList.\_items[j] = list1.\_items[i];  break;  }  else if (j == resultList.\_size - 1)  {  resultList.Add(list1.\_items[i]);  }  }  }  return resultList;  }  public double AverageAge(ListTeachers list)  {  int sum = 0;  for (int i = 0; i < list.\_size; i++)  {  sum += list.\_items[i].age;  }  double avg = sum / list.\_size;  return avg;  }  public ListTeachers DeleteUnderAverage(ListTeachers list, double avg)  {  for (int i = 0; i < list.\_size; i++)  {  while (true)  {  if (list.\_items[i].age < avg)  {  list.Remove(list.\_items[i]);  }  else  {  break;  }  }  }  return list;  }  public void WriteAllTeachers(string filePath, ListTeachers list)  {  StreamWriter sw = new StreamWriter(filePath);  string s = "";  for(int i = 0; i < list.\_size; i++)  {  string[] str = {list.\_items[i].id.ToString(),list.\_items[i].fullname,list.\_items[i].subject, list.\_items[i].age.ToString() };  s = string.Join(',',str);  sw.WriteLine(s);  }  sw.Close();  }  }  class Program  {  static void Main(string[] args)  {  //  ListTeachers newList = new ListTeachers();  newList = newList.ReadAllTeachers("./data2.csv");  newList.PrintFirst10();  WriteLine("Number of elements of second list: {0}", newList.GetCount());  WriteLine("Size of first list: {0}\r\n", newList.GetCapacity());  Teacher teacher = new Teacher(44, "tt", "reded", 44);  newList.Insert(14, teacher);  WriteLine(newList.GetAt(14));  newList.Remove(teacher);  WriteLine(newList.GetAt(13));  WriteLine();  //  ListTeachers newList2 = new ListTeachers();  newList2 = newList2.ReadAllTeachers("./data1.csv");  newList2.PrintFirst10();  WriteLine("\r\nNumber of elements of second list: {0}", newList2.GetCount());  WriteLine("Size of second list: {0}\r\n", newList2.GetCapacity());  WriteLine(newList2.GetAt(55));  WriteLine();  ListTeachers lastList = new ListTeachers();  lastList = lastList.MergeLists(newList, newList2);  lastList.PrintFirst10();  double avg = lastList.AverageAge(lastList);  WriteLine("Average age is {0}\r\n", avg);  Stopwatch sw = new Stopwatch();  sw.Start();  lastList = lastList.DeleteUnderAverage(lastList, avg);  sw.Stop();  lastList.PrintFirst10();  WriteLine("\r\n{0}", sw.Elapsed);  lastList.WriteAllTeachers("./dataout.csv", lastList);  }  }  } |

**Приклади результатів**

**Частина 1.**

|  |
| --- |
| Неправильний ввід аргументів командного рядка: |
|  |
|  |
| Неправильна кількість рядків у файлі: |
|  |
|  |
| Правильний ввід даних: |
|  |
| Файл "dd": |
|  |
| При вказанні існуючого файлу він буде перезаписаний: |
|  |
| Файл "dd": |
|  |

**Частина 2.**

|  |
| --- |
| public void PrintFirst10() |
|  |
| public int GetCount() |
|  |
| public int GetCapacity() |
|  |
| public Teacher GetAt(int index) |
|  |
| public void SetAt(int index, Teacher teacher) |
|  |
| public void Add(Teacher newTeacher) |
|  |
| public void Insert(int index, Teacher newTeacher) |
|  |
| public void Clear() |
|  |

|  |
| --- |
| data1.csv |
|  |
| data2.csv |
|  |
| dataout.csv |
|  |

**Висновки**

Виконавши дану лабораторну роботу було використано механізм інкапсуляції. **Інкапсуляція** - це механізм, що об’єднує дані і код, що маніпулює цими даними, а також захищає їх від зовнішнього впливу і неправильного використання.

Також при виконанні даної лабораторної роботи були розібрані основні принципи роботи з класами та об'єктами класів. **Клас** - новий тип даних, що грає роль шаблону, за допомогою якого можна створювати нові значення цього типу даних і виконувати для них поведінку, що описана в класі. **Об'єкт** - це окрема одиниця для зберігання даних із визначеною поведінкою.

При розробці програм даної лабораторної було проведене вивчення та часткова реалізація стандартних класів StringBuilder (System.Text), StringWriter та StringReader (System.IO). Для дослідження поведінки роботи програми використовувався стандартний клас Stopwatch (System.Diagnostics).

Компіляція всього коду відбувалася за допомогою утиліти dotnet.