НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

*Звiт з лабораторної роботи №5*

*Тема: «Серіалізація в C#.Делегати»*

Виконав:   
ст. гр. КІТ-119б  
Шарма О.Р.

Перевірив:  
Бартош М.В.

Харків – 2021

***Індивідуальне завдання:***

1. Для автоматичного розрахунку значень при читанні використовувати властивості "тільки для

читання "(без реалізації методу set).

2. Висновок для обраного студента назви групи (абревіатура назви факультету,

номер спеціальності, рік надходження, індекс).

3. Висновок для обраного студента номера курсу і семестру на поточний момент.

4. Розрахунок і висновок для обраного студента віку на поточний момент (до дня).

5. Продемонструвати ефективне використання StringBuilder для обробки рядків.

ОПИС ПРОГРАМИ

Дана програма містить три класи з методами.

**Клас Main**

Головний клас програми, містить:

* Метод main

Послідовність виконання дій :

* 1. Створення контейнеру, який міститиме масив об’єктів типу Student.
  2. Запит у користувача чи бажає він додати студента.
  3. Якщо так, то запит користувача заповнити всі дані.
  4. Створення на основі даних об’єкта типу Student.
  5. Додання об’єкту до контейнеру.
  6. Виведення на екран вміст контейнера за допомогою foreach.
  7. Знаходження студента по індексу.
  8. Збереження введених даних у файлі для подальшого редагування.
  9. Відновлення раніше збережених даних з файлу.
  10. Пошук і редагування даних студента.
  11. Видалення даних особової справи студента.
  12. Виведення для обраного студента назви групи.
  13. Виведення для обраного студента номера курсу і семестру на поточний момент.
  14. Розрахунок і виведення для обраного студента віку на поточний момент (до дня).
  15. Виведення на екран даних про студентів(обраної групи, спеціальності, ф-ту).
  16. Групове видалення даних.
  17. Виведення середнього віку студентів(обраної групи, спеціальності, ф-ту).
  18. Виведення середньої успішності студентів(обраної групи, спеціальності, ф-ту).
  19. Серіалізація/десеріалізація об’єкту в XML файл.

**Клас Student**

Містить поля – інформація про студенів. Містить сеттери та геттери, а також метод toString та два конструктори.

**Клас Container**

Наслідується від IEnumerable та IEnumerator. Містить поля – масив студентів, розмір масиву. Також містить методи GetEnumerator, MoveNext, Reset, Current, Add, NewSize, Search, Remove, Edit, Group, GrRemove, Print, AvAge, AvProgress.

ТЕКСТ ПРОГРАМИ

**Program.cs**

using System;

using System.IO;

using System.Text;

using System.Xml.Serialization;

namespace Sharma05

{

class Program

{

delegate int Average(Container records, int number, String str);

static void Main(string[] args)

{

string path = @"C:/Users/{Dean}/source/repos/Sharma05/Sharma05/saved.txt”;

StringBuilder sb = new StringBuilder();

var records = new Container();

bool loop = true;

int number;

int choice;

Average average;

while (loop)

{

Console.WriteLine("Что Вы хотите сделать?\n 1 - Добавить данные про студента\n 2 - Вывести на экран данные\n 3 - Записать данные в файл" +

"\n 4 - Прочитать данные из файла\n 5 - Найти элемент по индексу\n 6 - Удалить данные о студенте\n 7 - Редактировать данные студента" +

"\n 8 - Вывести название группы студента\n 9 - Вывести текущий курс и семестр студента\n 10 - Вывести текущий возраст студента\n" +

" 11 - Вывести на экран данные о студентах(выбранной группы, специальности, факультета)\n 12 - Групповое удаление данных\n" +

" 13 - Средний возраст(выбранной группы, специальности, факультета)\n 14 - Средняя успеваемость(выбранной группы, специальности, факультета)\n" +

" 15 - Выход");

choice = int.Parse(Console.ReadLine());

switch (choice)

{

case 1:

Helper.AddStud(records);

break;

case 2:

Console.WriteLine("Имя\t Фамилия\t Отчество\t Дата рождения\t\t Дата поступления Индекс Ф-тет Спец-сть Усп-сть");

Console.WriteLine("------------------------------------------------------------------------------------------------------------------------");

foreach (var student in records)

{

Console.WriteLine(student + " ");

}

break;

case 3:

Helper.WriteFile(records, path);

break;

case 4:

Helper.ReadFile(records, path);

break;

case 5:

Console.WriteLine("Номер студента, которого хотите найти: ");

number = int.Parse(Console.ReadLine());

records.Search(records, number);

break;

case 6:

Console.WriteLine("Номер студента, данные о котором хотите удалить: ");

number = int.Parse(Console.ReadLine());

records.Remove(records, number);

break;

case 7:

Console.WriteLine("Номер студента, данные о котором хотите отредактировать: ");

number = int.Parse(Console.ReadLine());

Console.WriteLine("Что хотите отредактировать? (1-имя, 2 - фамилию, 3 - отчество, 4 - день рождения, 5 - дата поступления, " +

"6 - индекс группы, 7 - факультет, 8 - специальность, 9 - успеваемость");

int n;

string str;

n = int.Parse(Console.ReadLine());

Console.WriteLine("Введите новые данные в соответствующем формате : ");

str = Console.ReadLine();

records.Edit(records, number, n, str);

break;

case 8:

Console.WriteLine("Номер студента, чью группу хотите узнать: ");

number = int.Parse(Console.ReadLine());

Helper.Group(records, number);

break;

case 9:

Console.WriteLine("Номер студента, чей номер курса и семестра на текущий момент хотите узнать: ");

number = int.Parse(Console.ReadLine());

Helper.Course(records, number);

break;

case 10:

Console.WriteLine("Номер студента, чей текущий возраст хотите узнать: ");

number = int.Parse(Console.ReadLine());

DateTime today = DateTime.Today;

DateTime b = records.students[number - 1].Birthday;

TimeSpan old = today.Subtract(b);

var d = new DateTime(old.Ticks);

Console.WriteLine($"Возраст: {d.Year - 1} лет, {d.Month - 1} месяцев, {d.Day - 1} дней");

break;

case 11:

Console.WriteLine("По какому критерию вывести список студентов? (1 - группа, 2 - специальность, 3 - факультет)");

int num;

string str1;

num = int.Parse(Console.ReadLine());

switch (num)

{

case 1:

Console.WriteLine("Введите группу : ");

break;

case 2:

Console.WriteLine("Введите специальность : ");

break;

case 3:

Console.WriteLine("Введите факультет : ");

break;

}

str1 = Console.ReadLine();

records.Print(records, num, str1);

break;

case 12:

Console.WriteLine("По какому критерию удалить студентов? (1 - группа, 2 - специальность, 3 - факультет)");

int numb;

string str2;

numb = int.Parse(Console.ReadLine());

switch (numb)

{

case 1:

Console.WriteLine("Введите группу : ");

break;

case 2:

Console.WriteLine("Введите специальность : ");

break;

case 3:

Console.WriteLine("Введите факультет : ");

break;

}

str2 = Console.ReadLine();

records.GrRemove(records, numb, str2);

break;

case 13:

Console.WriteLine("По какому критерию средний возраст студентов? (1 - группа, 2 - специальность, 3 - факультет)");

int num2;

string str3;

num2 = int.Parse(Console.ReadLine());

switch (num2)

{

case 1:

Console.WriteLine("Введите группу : ");

break;

case 2:

Console.WriteLine("Введите специальность : ");

break;

case 3:

Console.WriteLine("Введите факультет : ");

break;

}

str3 = Console.ReadLine();

average = records.AvAge;

Console.WriteLine("Средний возраст: " + average(records, num2, str3));

break;

case 14:

Console.WriteLine("По какому критерию расчитать среднюю успеваемость студентов? (1 - группа, 2 - специальность, 3 - факультет)");

int num3;

string str4;

num3 = int.Parse(Console.ReadLine());

switch (num3)

{

case 1:

Console.WriteLine("Введите группу : ");

break;

case 2:

Console.WriteLine("Введите специальность : ");

break;

case 3:

Console.WriteLine("Введите факультет : ");

break;

}

str4 = Console.ReadLine();

average = records.AvProgress;

Console.WriteLine("Средняя успеваемость: " + average(records, num3, str4));

break;

case 15:

loop = false;

break;

}

}

XmlSerializer formatter = new XmlSerializer(typeof(Student[]));

using (FileStream fs = new FileStream("students.xml", FileMode.OpenOrCreate))

{

formatter.Serialize(fs, records.students);

}

using (FileStream fs = new FileStream("students.xml", FileMode.OpenOrCreate))

{

Student[] newStud = (Student[])formatter.Deserialize(fs);

foreach (Student p in newStud)

{

Console.WriteLine(p);

}

}

}

}

}

**Helper.cs**

using System;

using System.IO;

using System.Text;

namespace Sharma05

{

class Helper

{

public static void AddStud(Container records)

{

string name, lastname, patronym, faculty;

DateTime birthday, date;

int progress, specialty;

char index;

Console.WriteLine("Имя: ");

name = Console.ReadLine();

Console.WriteLine("Фамилия: ");

lastname = Console.ReadLine();

Console.WriteLine("Отчество: ");

patronym = Console.ReadLine();

Console.WriteLine("День рождения: ");

birthday = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Дата поступления: ");

date = DateTime.Parse(Console.ReadLine());

Console.WriteLine("Индекс группы: ");

index = char.Parse(Console.ReadLine());

Console.WriteLine("Факультет: ");

faculty = Console.ReadLine();

Console.WriteLine("Специальность: ");

specialty = int.Parse(Console.ReadLine());

Console.WriteLine("Успеваемость: ");

progress = int.Parse(Console.ReadLine());

var stud = new Student(name, lastname, patronym, birthday, date, index, faculty, specialty, progress);

records.Add(stud);

}

public static void ReadFile(Container records, string path)

{

string name, lastname, patronym, faculty, text;

DateTime birthday, date;

int progress, specialty;

char index;

try

{

using (StreamReader sr = new StreamReader(path))

{

text = sr.ReadToEnd();

string[] separatingStrings = { " ", "\r", "\n", ":", "00:00:00", "\t", "Студенты", "Имя", "Фамилия", "Отчество", "День рождения",

"Дата поступления", "Индекс группы", "Факультет", "Специальность", "Успеваемость"};

string[] words = text.Split(separatingStrings, System.StringSplitOptions.RemoveEmptyEntries);

for (int i = 0; i < words.Length / 9; i++)

{

name = words[9 \* i];

lastname = words[9 \* i + 1];

patronym = words[9 \* i + 2];

birthday = DateTime.Parse(words[9 \* i + 3]);

date = DateTime.Parse(words[9 \* i + 4]);

index = char.Parse(words[9 \* i + 5]);

faculty = words[9 \* i + 6];

specialty = int.Parse(words[9 \* i + 7]);

progress = int.Parse(words[9 \* i + 8]);

var stud = new Student(name, lastname, patronym, birthday, date, index, faculty, specialty, progress);

records.Add(stud);

}

}

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void WriteFile(Container records, string path)

{

try

{

using (StreamWriter sw = new StreamWriter(path, false, System.Text.Encoding.Default))

{

sw.WriteLine("Студенты: ");

}

foreach (var student in records)

{

using StreamWriter sw = new StreamWriter(path, true, System.Text.Encoding.Default);

sw.WriteLine(student);

}

Console.WriteLine("Запись выполнена");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

public static void Course(Container records, int number)

{

int course, semester;

course = DateTime.Today.Year - records.students[number - 1].Date.Year;

if (DateTime.Today.Month >= 7 && DateTime.Today.Month <= 12)

{

semester = course \* 2 - 1;

}

else

{

semester = course \* 2;

}

Console.WriteLine($"Курс : {course}, семестр : {semester}");

}

public static void Group(Container records, int number)

{

StringBuilder sb = new StringBuilder();

sb.Append(records.students[number - 1].Faculty);

sb.Append(records.students[number - 1].Specialty);

sb.Append("-");

sb.Append(records.students[number - 1].Date.Year);

sb.Append(records.students[number - 1].Index);

sb.AppendLine();

Console.WriteLine(sb.ToString());

}

}

}

**Student.cs**

using System;

namespace Sharma05

{

[Serializable]

public class Student

{

public string Name { get; set; }

public string Lastname { get; set; }

public string Patronym { get; set; }

public DateTime Birthday { get; set; }

public DateTime Date { get; set; }

public char Index { get; set; }

public string Faculty { get; set; }

public int Specialty { get; set; }

public int Progress { get; set; }

public Student() { }

public Student(string name, string lastname, string patronym, DateTime birthday, DateTime date, char index, string faculty, int specialty, int progress)

{

this.Name = name;

this.Lastname = lastname;

this.Patronym = patronym;

this.Birthday = birthday;

this.Date = date;

this.Index = index;

this.Faculty = faculty;

this.Specialty = specialty;

this.Progress = progress;

}

public override string ToString()

{

return $"{Name}\t {Lastname}\t {Patronym}\t {Birthday}\t {Date}\t {Index}\t {Faculty}\t {Specialty}\t {Progress} \n";

}

}

}

**Container.cs**

using System;

using System.Collections;

using System.Text;

namespace Sharma05

{

public class Container : IEnumerable, IEnumerator

{

static int size = 1;

public Student[] students = new Student[size];

int index = -1;

int currentSize = 0;

// Реализуем интерфейс IEnumerable

public IEnumerator GetEnumerator()

{

return this;

}

// Реализуем интерфейс IEnumerator

public bool MoveNext()

{

if (index == students.Length - 1)

{

Reset();

return false;

}

index++;

return true;

}

public void Reset()

{

index = -1;

}

public object Current

{

get

{

return students[index];

}

}

public void Add(Student student)

{

currentSize = NewSize();

if (currentSize < size)

{

students[currentSize] = student;

}

else

{

size = currentSize + 1;

Array.Resize(ref students, size);

students[size - 1] = student;

}

}

int NewSize()

{

currentSize = 0;

for (int i = 0; i < students.Length; i++)

{

if (students[i] != null)

currentSize++;

}

return currentSize;

}

public void Search(Container records, int number)

{

Console.WriteLine("Студент по индексу :");

Console.Write(records.students[number - 1] + " ");

}

public void Remove(Container records, int number)

{

currentSize = NewSize();

students[number - 1] = null;

size--;

for (int i = number; i < currentSize; i++)

{

if (students[i] != null)

{

students[i - 1] = students[i];

}

}

Array.Resize(ref students, currentSize - 1);

}

public void Edit(Container records, int number, int n, string str)

{

switch (n)

{

case 1:

records.students[number - 1].Name = str;

break;

case 2:

records.students[number - 1].Lastname = str;

break;

case 3:

records.students[number - 1].Patronym = str;

break;

case 4:

records.students[number - 1].Birthday = DateTime.Parse(str);

break;

case 5:

records.students[number - 1].Date = DateTime.Parse(str);

break;

case 6:

records.students[number - 1].Index = char.Parse(str);

break;

case 7:

records.students[number - 1].Faculty = str;

break;

case 8:

records.students[number - 1].Specialty = int.Parse(str);

break;

case 9:

records.students[number - 1].Progress = int.Parse(str);

break;

}

Console.WriteLine("Измененный список студентов :");

foreach (var student in records)

{

Console.WriteLine(student + " ");

}

}

public string Group(Container records, int number)

{

StringBuilder sb = new StringBuilder();

sb.Append(records.students[number - 1].Faculty);

sb.Append(records.students[number - 1].Specialty);

sb.Append("-");

sb.Append(records.students[number - 1].Date.Year);

sb.Append(records.students[number - 1].Index);

sb.AppendLine();

return sb.ToString();

}

public void GrRemove(Container records, int number, String str)

{

switch (number)

{

case 1:

for (int i = 0; i < NewSize(); i++)

{

if (Group(records, i + 1).Equals(str + "\r\n"))

{

records.Remove(records, i + 1);

}

}

break;

case 2:

int spec = int.Parse(str);

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Specialty.Equals(spec))

{

records.Remove(records, i + 1);

}

}

break;

case 3:

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Faculty.Equals(str))

{

records.Remove(records, i + 1);

i--;

}

}

break;

}

}

public void Print(Container records, int number, String str)

{

switch (number)

{

case 1:

for (int i = 0; i < NewSize(); i++)

{

if (Group(records, i + 1).Equals(str + "\r\n"))

{

Console.WriteLine(records.students[i]);

}

}

break;

case 2:

int spec = int.Parse(str);

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Specialty.Equals(spec))

{

Console.WriteLine(records.students[i]);

}

}

break;

case 3:

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Faculty.Equals(str))

{

Console.WriteLine(records.students[i]);

}

}

break;

}

}

public int AvAge(Container records, int number, String str)

{

int count = 0, age = 0, av;

DateTime today = DateTime.Today;

DateTime b;

switch (number)

{

case 1:

for (int i = 0; i < NewSize(); i++)

{

if (Group(records, i + 1).Equals(str + "\r\n"))

{

b = records.students[i].Birthday;

TimeSpan old = today.Subtract(b);

var d = new DateTime(old.Ticks);

count++;

age += d.Year;

}

}

break;

case 2:

int spec = int.Parse(str);

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Specialty.Equals(spec))

{

b = records.students[i].Birthday;

TimeSpan old = today.Subtract(b);

var d = new DateTime(old.Ticks);

count++;

age += d.Year;

}

}

break;

case 3:

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Faculty.Equals(str))

{

b = records.students[i].Birthday;

TimeSpan old = today.Subtract(b);

var d = new DateTime(old.Ticks);

count++;

age += d.Year;

}

}

break;

}

av = age / count;

return av;

}

public int AvProgress(Container records, int number, String str)

{

int count = 0, progress = 0, av;

switch (number)

{

case 1:

for (int i = 0; i < NewSize(); i++)

{

if (Group(records, i + 1).Equals(str + "\r\n"))

{

count++;

progress += records.students[i].Progress;

}

}

break;

case 2:

int spec = int.Parse(str);

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Specialty.Equals(spec))

{

count++;

progress += records.students[i].Progress;

}

}

break;

case 3:

for (int i = 0; i < NewSize(); i++)

{

if (records.students[i].Faculty.Equals(str))

{

count++;

progress += records.students[i].Progress;

}

}

break;

}

av = progress / count;

return av;

}

}

}

РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ

Результати роботи програми приведено нижче на скріншотах:

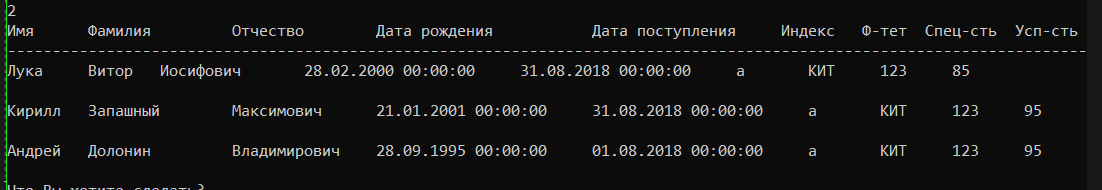


Рисунок 1 –вхідні дані

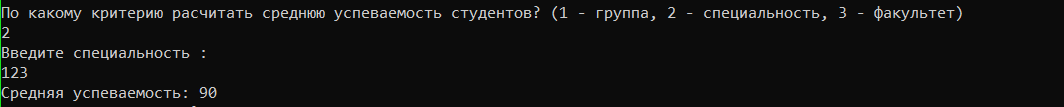
ъ 

Рисунок 2 – ередня успішність

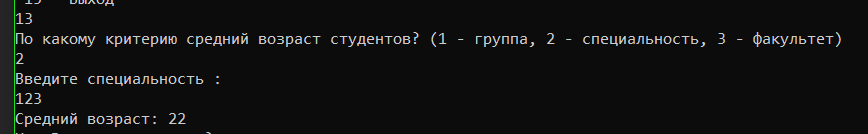


Рисунок 3 – середній вік

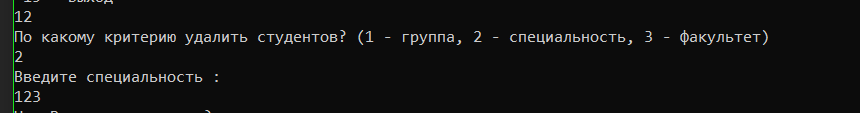
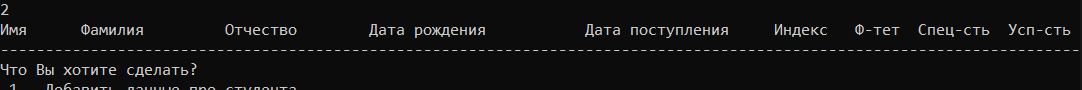


Рисунок 4 – запит на групове видалення даних

Рисунок 5 – успішне видалення

Висновки:

В ході данної лабораторної роботи було доповнено програму з попередньої роботи, було додано делегати для групування даних, реалізовано видалення та пошук за допомогою делегатів, а також підрахунок середнього віку за критеріями.