Титов Павел Сергеевич, группа 04-1, 7 лабораторная работа

**Лабораторная работа №7.**

Работа со структурами и перечислениями

Разработать программу, которая выполняет следующие действия:

1. Считывает информацию в массив структур (или в связный список структур).

Поле с названием института должно быть перечисляемого типа: в программе следует создать перечисление с институтами.

2. Выводит информацию о студентах заданного института.

Выбор института должен осуществляться посредством меню. Все названия институтов из перечисления (тип Enum) выводятся на экран, и пользователь получает возможность выбора, указывая порядковый номер, при этом выбираемый институт подсвечивается красным цветом, выбор заканчивается после нажатия клавиши **Enter**.

3. Записывает в новый текстовый файл информацию о студентах в виде:

Если информации по какому-то институту нет, то в файл записывается заголовок с институтом и вместо таблицы фраза «данных нет».

Информация в каждой таблице должна быть отсортирована по невозрастанию возраста студента.

4. Реализует несколько методов:

а) Получить список институтов в порядке численности студентов.

б) По каждому институту получить средний балл и средний возраст студентов

**в) Два своих метода.**

**PROGRAM.CS**

namespace test\_lab\_7;

public class Program {

public static void Main() {

FileConverter fc = new();

string filePath = "input.data";

string savedPath = "output.data";

List<Student> students = fc.LoadStudentsFromFile(filePath);

Menu menu = new Menu();

menu.Students = students;

menu.ChoiceMenu();

AverageScoreAndAge(students);

MinimalScore(students);

MaximumScore(students);

fc.StudentsToFile(savedPath, students);

}

static void MinimalScore(List<Student> students) {

Console.WriteLine("Минимальный средний балл у:");

if (students.Any()) {

var student = students.OrderBy(el => el.AverageScore).ToArray()[0];

Console.WriteLine($"{student.Surname} - {student.AverageScore}");

}

}

static void MaximumScore(List<Student> students) {

Console.WriteLine("Максимальный средний балл у:");

if (students.Any()) {

var student = students.OrderByDescending(el => el.AverageScore).ToArray()[0];

Console.WriteLine($"{student.Surname} - {student.AverageScore}");

}

}

static void AverageScoreAndAge(List<Student> students) {

Console.WriteLine("Средний возраст и балл студентов по каждому институту:");

foreach (string institute in Enum.GetNames(typeof(Institute))) {

List<Student> studentsThis = students

.Where(x => Enum

.GetName(typeof(Institute), x.Institute) == institute)

.ToList();

Console.WriteLine($"Институт: {institute}");

if (!studentsThis.Any()) {

Console.WriteLine("\tНедостаточно данных для подсчёта");

continue;

}

Console.WriteLine($"\tСредний балл - {studentsThis.Average(x => x.AverageScore)}");

Console.WriteLine($"\tСредний возраст - {studentsThis.Average(x => x.Age)}");

}

Console.WriteLine();

}

}

**MENU.CS**

namespace test\_lab\_7;

public class Menu

{

private const ConsoleColor DefaultColor = ConsoleColor.White;

private List<InstituteMenu> \_instituteMenus = new();

private List<Student> \_students = new();

private readonly string \_fill = new(' ', Console.WindowWidth);

private bool \_isWork;

private int \_lastActiveInstituteIndex = -1;

private readonly ConsoleKey[] \_pointer =

{

ConsoleKey.D1,

ConsoleKey.D2,

ConsoleKey.D3,

ConsoleKey.D4,

ConsoleKey.D5

};

public List<Student> Students { get => \_students; set => \_students = value; }

public void Write(string text, ConsoleColor color = DefaultColor)

{

Console.ForegroundColor = color;

Console.Write(text);

Console.ForegroundColor = DefaultColor;

}

public void WriteLine(string text, ConsoleColor color = DefaultColor)

{

Write(text + "\n", color);

}

public string[] GetInstitutesOrderByStudents()

{

return Enum

.GetNames(typeof(Institute))

.OrderByDescending(x => \_students

.Count(y => Enum

.GetName(typeof(Institute), y.Institute) == x))

.ToArray();

}

public void InitializationMenu()

{

string[] institutes = GetInstitutesOrderByStudents();

WriteLine("Список доступных институтов: ", ConsoleColor.DarkCyan);

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

{

int left = Console.CursorLeft;

int top = Console.CursorTop;

int studentsCount = \_students.Count(el => el.Institute.ToString() == institutes[i]);

WriteLine($"{i + 1,2} {institutes[i],-10} ({studentsCount} students)", ConsoleColor.Blue);

\_instituteMenus.Add(new InstituteMenu()

{

Index = i,

Institute = (Institute)Enum.Parse(typeof(Institute), institutes[i]),

Left = left,

Top = top,

StudentsCount = studentsCount

});

}

WriteLine("Выберите институт", ConsoleColor.DarkCyan);

}

private void EnableInstitute(int index)

{

if (\_lastActiveInstituteIndex >= 0)

DisableInstitute(\_lastActiveInstituteIndex);

var institute = \_instituteMenus[index];

Console.SetCursorPosition(institute.Left, institute.Top);

Write($"{institute.Index + 1,2} {institute.Institute,-10} ({institute.StudentsCount} students)",

ConsoleColor.Red);

LogMessage($"Выбран {institute.Institute}, для продолжения нажмите Enter", ConsoleColor.White);

\_lastActiveInstituteIndex = index;

}

private void DisableInstitute(int index)

{

var institute = \_instituteMenus[index];

Console.SetCursorPosition(institute.Left, institute.Top);

Write($"{institute.Index + 1,2} {institute.Institute,-10} ({institute.StudentsCount} students)",

ConsoleColor.Blue);

}

public void PrintCurrentStudentsInInstitute(Institute institute)

{

Console.WriteLine($"Институт: {institute}");

Console.WriteLine($"|{"№",2} | {"Фамилия",10} | {"Cредний балл",15} | {"Возраст",10} |");

var curStudents = \_students

.Where(stud => stud.Institute == institute)

.OrderByDescending(el => el.Age)

.ThenByDescending(el => el.AverageScore)

.Take(5)

.ToList();

Console.WriteLine(new string('-', 49));

for (int i = 0; i < curStudents.Count; i++)

{

Console.WriteLine(

$"|{i + 1,2} | {curStudents[i].Surname,10} | {curStudents[i].AverageScore,15} | {curStudents[i].Age,10} |");

}

}

public void ChoiceMenu()

{

Console.Clear();

InitializationMenu();

\_isWork = true;

var userHook = new Task(GetInput);

userHook.Start();

while (\_isWork)

{

Thread.Sleep(50);

}

Console.WriteLine();

userHook.Dispose();

PrintCurrentStudentsInInstitute(GetCurrentInstitute());

}

public Institute GetCurrentInstitute()

{

return \_instituteMenus[\_lastActiveInstituteIndex].Institute;

}

private void LogMessage(string message, ConsoleColor color)

{

Console.SetCursorPosition(0, 20);

Console.Write(\_fill);

Console.SetCursorPosition(0, 20);

Write($"> {message}", color);

}

private void GetInput()

{

while (\_isWork)

{

var a = Console.ReadKey(true);

if (a.Key == ConsoleKey.Escape)

{

\_isWork = false;

}

else if (\_pointer.Contains(a.Key))

{

EnableInstitute(Array.IndexOf(\_pointer, a.Key));

LogMessage($"Выбран {\_instituteMenus[Array.IndexOf(\_pointer, a.Key)].Institute}", ConsoleColor.Cyan);

}

else if (a.Key == ConsoleKey.Enter)

{

if (\_lastActiveInstituteIndex < 0)

{

LogMessage("Вы не выбрали институт. Выход ...", ConsoleColor.Red);

Environment.Exit(0);

}

var institute = \_instituteMenus[\_lastActiveInstituteIndex];

LogMessage($"Выбран {institute.Institute}, вывод первых элементов ...", ConsoleColor.White);

\_isWork = false;

}

else

{

LogMessage($"Pressed {a.Key} key", ConsoleColor.Gray);

}

}

}

}

**INSTITUTE.CS**

namespace test\_lab\_7;

public enum Institute

{

Фэи,

Ифк,

Имикн,

ИнЗем,

ИФиЖ

};

**Structs.cs**

namespace test\_lab\_7;

public struct Student : IComparable

{

public string Surname;

public string DateOfBirth;

public Institute Institute;

public double AverageScore;

public double Age => (int)Math.Floor((DateTime.Now - Convert.ToDateTime(DateOfBirth)).TotalDays / 365.25D);

public int CompareTo(object obj) => Age.CompareTo(((Student)obj).Age);

}

public struct InstituteMenu

{

public Institute Institute;

public int Top;

public int Left;

public int Index;

public int StudentsCount;

}

**FileConverter.cs**

namespace test\_lab\_7;

public class FileConverter

{

public List<Student> LoadStudentsFromFile(string filePath)

{

string[][] data = File.ReadAllLines(filePath).Select(x => x.Split(';')).ToArray();

List<Student> students = new();

foreach (string[] line in data)

{

students.Add(new Student()

{

Surname = line[0],

DateOfBirth = line[1],

Institute = (Institute)Enum.Parse(typeof(Institute), line[2]),

AverageScore = Convert.ToDouble(line[3])

});

}

return students;

}

public void StudentsToFile(string savePath, List<Student> students)

{

*//Подготовка информации для записи в файл*

using var file = new StreamWriter(savePath);

string[] instituteNames = Enum.GetNames(typeof(Institute));

foreach (var institute in instituteNames)

{

file.WriteLine($"Институт: {institute}");

var curStudents = students

.Where(stud => stud.Institute.ToString() == institute)

.OrderByDescending(el => el.Age)

.ThenByDescending(el => el.AverageScore).ToList();

if (curStudents.Count == 0)

{

file.WriteLine("Данных нет");

continue;

}

file.WriteLine($"|{"№",2} | {"Фамилия",10} | {"Cредний балл",15} | {"Возраст",10} |");

file.WriteLine(new string('-', 49));

for (int i = 0; i < curStudents.Count; i++)

{

file.WriteLine(

$"|{i + 1,2} | {curStudents[i].Surname,10} | {curStudents[i].AverageScore,15} | {curStudents[i].Age,10} |");

}

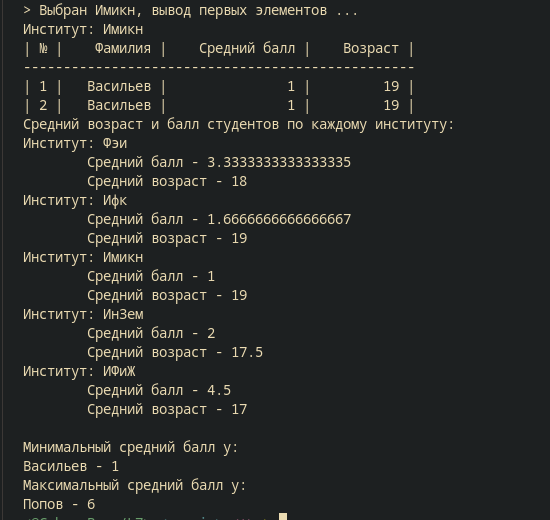
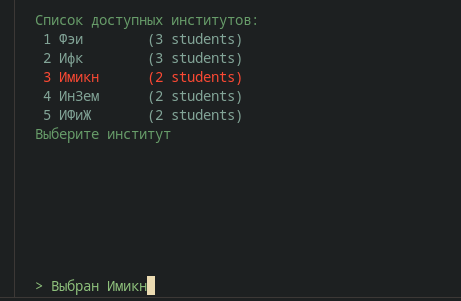
file.WriteLine();

}

}

}

**Вывод:**

****