Отчет по практической работе №9

Наименование работы: перечисления. Структуры.

Индивидуальные задания (вариант №9):

1. "Жених". Юноша хочет найти невесту. Кандидатов достаточно много. Известен список бинарных свойств, которые молодой человек хочет знать о претендентках. Часть из этих свойств юноша относит к положительным, часть - к отрицательным. Юноше требуется из списка претенденток составить список предпочтительных невест, упорядоченный по степени предпочтения. Предложите разумный алгоритм и реализуйте его. Указание: список свойств следует задать перечислением, представляющим шкалу. Следует определить структура Bride, среди полей которого будет поле properties, заданное перечислением. Необходимо создать проект, моделирующий решение поставленной задачи.
2. На основе данных входного файла составить список студентов, включающий фамилию, факультет, курс, группу, 5 оценок. Вывести в новый файл информацию о тех студентах, которые имеют хотя бы одну двойку, отсортировав их по курсу.

Вопросы выходного контроля:

1. Что такое перечисление?

Ответ:

Перечисления C# – enum C# — это набор именованных констант, позволяющих сделать исходный код программы более удобочитаемым.

1. Что такое интерфейс?

Ответ:

Интерфейс представляет ссылочный тип, который может определять некоторый функционал - набор методов и свойств без реализации. Затем этот функционал реализуют классы и структуры, которые применяют данные интерфейсы.

1. Какие спецификаторы могут быть указаны для интерфейса?

Ответ:

Метода, свойства, индексаторы, события, статистические поля и константы.

Ход работы:

1. Разработал программу в соответствии с заданием №1. Были реализованы класс, метод, математические функции.

Листинг программы:

using selBride.Enums;

using System;

using System.Linq;

namespace selBride

{

class Program

{

private static Bride[] Brides = Settings.GetBrides();

static void Main(string[] args)

{

var bride = new PropBride();

try

{

OutputOnDisplay("Введите от скольки лет желаете выбрать невесту:");

bride.Age = Convert.ToInt32(Console.ReadLine());

}

catch

{

bride.Age = 0;

}

Console.Clear();

try

{

OutputOnDisplay("Введите от какого роста желаете выбрать невесту:");

bride.Growth = Convert.ToInt32(Console.ReadLine());

}

catch

{

bride.Growth = 0;

}

Console.Clear();

try

{

OutputOnDisplay("Введите от скакого веса желаете выбрать невесту:");

bride.Weight = Convert.ToInt32(Console.ReadLine());

}

catch

{

bride.Weight = 0;

}

Console.Clear();

try

{

OutputOnDisplay("Выберете отношение к курению:");

OutputOnDisplay($"1) {nameof(Smoking.Smokes)}");

OutputOnDisplay($"2) {nameof(Smoking.Negative)}");

OutputOnDisplay($"3) {nameof(Smoking.Neutral)}");

int i = Convert.ToInt32(Console.ReadLine());

bride.Smoking = i < 4 && i > 0

? (Smoking)i

: Smoking.NaN;

}

catch

{

bride.Smoking = 0;

}

Console.Clear();

try

{

OutputOnDisplay("Выберете отношение к алкаголю:");

OutputOnDisplay($"1) {nameof(Alcohol.Alcoholic)}");

OutputOnDisplay($"2) {nameof(Alcohol.Moderately)}");

OutputOnDisplay($"3) {nameof(Alcohol.NotDrinking)}");

int i = Convert.ToInt32(Console.ReadLine());

bride.Alcohol = i < 4 && i > 0

? (Alcohol)i

: Alcohol.NaN;

}

catch

{

bride.Alcohol = 0;

}

Console.Clear();

try

{

OutputOnDisplay("Выберете телосложение:");

OutputOnDisplay($"1) {nameof(BodyType.Fat)}");

OutputOnDisplay($"2) {nameof(BodyType.Сommon)}");

OutputOnDisplay($"3) {nameof(BodyType.Thin)}");

OutputOnDisplay($"4) {nameof(BodyType.Sports)}");

int i = Convert.ToInt32(Console.ReadLine());

bride.BodyType = i < 5 && i >0

? (BodyType)i

: BodyType.NaN;

}

catch

{

bride.BodyType = 0;

}

Console.Clear();

printPropBride(bride);

Console.ReadKey();

Console.Clear();

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

{

if(bride.Age > 0 && Brides[i].Age >= bride.Age)

{

Brides[i].Priority++;

}

if (bride.Weight > 0 && Brides[i].Weight >= bride.Weight)

{

Brides[i].Priority++;

}

if (bride.Growth > 0 && Brides[i].Growth >= bride.Growth)

{

Brides[i].Priority++;

}

if (bride.Alcohol > 0 && Brides[i].Alcohol == bride.Alcohol)

{

Brides[i].Priority++;

}

if (bride.BodyType > 0 && Brides[i].BodyType == bride.BodyType)

{

Brides[i].Priority++;

}

if (bride.Smoking > 0 && Brides[i].Smoking == bride.Smoking)

{

Brides[i].Priority++;

}

}

var selBrides = Brides.Where(b => b.Priority > 0).OrderByDescending(b => b.Priority);

foreach (var b in selBrides)

{

printBride(b);

}

}

private static void printPropBride(PropBride bride)

{

OutputOnDisplay($"Возрст: {bride.Age}");

OutputOnDisplay($"Вес: {bride.Weight} кг");

OutputOnDisplay($"Рост: {bride.Growth} см");

OutputOnDisplay($"Отношение к курению: {bride.Smoking}");

OutputOnDisplay($"Отношение к алкоолю: {bride.Alcohol}");

OutputOnDisplay($"Телосложение: {bride.BodyType}");

}

private static void printBride(Bride bride)

{

OutputOnDisplay($"Имя: {bride.Name}");

OutputOnDisplay($"Возрст: {bride.Age}");

OutputOnDisplay($"Вес: {bride.Weight} кг");

OutputOnDisplay($"Рост: {bride.Growth} см");

OutputOnDisplay($"Отношение к курению: {bride.Smoking}");

OutputOnDisplay($"Отношение к алкоолю: {bride.Alcohol}");

OutputOnDisplay($"Телосложение: {bride.BodyType}");

OutputOnDisplay($"Приоритет: {bride.Priority}");

OutputOnDisplay("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

private static void OutputOnDisplay(string message)

{

Console.WriteLine(message);

}

private class PropBride

{

public int Age { get; set; }

public int Weight { get; set; }

public int Growth { get; set; }

public Alcohol Alcohol { get; set; }

public Smoking Smoking { get; set; }

public BodyType BodyType { get; set; }

}

}

}

using selBride.Enums;

using System;

using System.Collections.Generic;

using System.Text;

namespace selBride

{

public class Bride

{

public string Name { get; set; }

public int Age { get; set; }

public int Weight { get; set; }

public int Growth { get; set; }

public Alcohol Alcohol { get; set; }

public Smoking Smoking { get; set; }

public BodyType BodyType { get; set; }

public int Priority { get; set; } = 0;

}

}

using selBride.Enums;

namespace selBride

{

public class Settings

{

public static Bride[] GetBrides()

{

return new Bride[] {

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Mariya",

Age = 19,

Weight = 53,

Growth = 150,

Alcohol = Alcohol.Alcoholic,

Smoking = Smoking.Smokes,

BodyType = BodyType.Fat

},

new Bride()

{

Name = "Galina",

Age = 23,

Weight = 117,

Growth = 190,

Alcohol = Alcohol.Alcoholic,

Smoking = Smoking.Smokes,

BodyType = BodyType.Fat

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

},

new Bride()

{

Name = "Anna",

Age = 18,

Weight = 63,

Growth = 170,

Alcohol = Alcohol.Moderately,

Smoking = Smoking.Negative,

BodyType = BodyType.Сommon

}

};

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace selBride.Enums

{

public enum Alcohol

{

NaN,

Alcoholic,

Moderately,

NotDrinking

}

}

namespace selBride.Enums

{

public enum BodyType

{

NaN,

Fat,

Сommon,

Thin,

Sports

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace selBride

{

public enum Smoking

{

NaN,

Smokes,

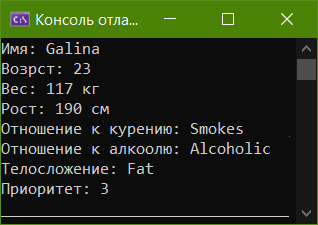
Negative,

Neutral

}

}

Результат выполнения программы Рис 1.1:



**Рисунок 1.1 – результат программы 9.1.**

1. Разработал программу в соответствии с заданием №1. Были реализованы класс, метод, математические функции.

Листинг программы:

using System;

using System.IO;

using System.Text;

namespace ConsoleApplicationTest

{

public static class Program

{

static string path = @"users.txt";

static string outs = @"output.txt";

private static void Main()

{

Auth();

string[] allLines = File.ReadAllLines(path);

Student[] students = new Student[allLines.Length];

string[] fields = new string[5];

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

{

string line = allLines[i];

fields = line.Split(';');

Student student = new Student(fields[0], fields[1], fields[2], fields[3], fields[4]);

students[i] = student;

}

Array.Sort(students);

foreach (Student student in students)

{

Console.WriteLine(student);

}

string[] linesToSave = new string[students.Length];

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

{

for(int j = 0; j < 5; j++)

{

if (students[i].Ocen[j] == 2)

{

Student student = students[i];

linesToSave[i] = student.ToString();

}

}

}

// Сохраняем в файл

FileInfo fi = new FileInfo(outs);

if (fi.Exists)

{

File.WriteAllLines(outs, linesToSave, Encoding.UTF8);

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Отсортированные данные записаны в файл {outs}");

Console.ResetColor();

}

else

{

File.Create(outs);

File.WriteAllLines(outs, linesToSave);

Console.ReadKey();

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Отсортированные данные записаны в файл {outs}");

Console.ResetColor();

}

}

public static void Auth()

{

FileInfo fn = new FileInfo(path);

if (fn.Exists)

{

Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine($"Файл {path} существует! Продолжаем работу!");

Console.ResetColor();

}

else

{

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine($"Файл {path} не найден! Проверьте расположение файла!");

Console.ResetColor();

}

}

}

public struct Student : IComparable

{

public Student(string name, string fac, string group, string cours, string nums)

: this()

{

Name = name;

Fac = fac;

Group = group;

Cours = cours;

Nums = nums;

string[] str = nums.Split(' ');

Ocen = new int[5];

for (int i = 0; i < 5; i++)

{

Ocen[i] = Convert.ToInt32(str[i]);

}

}

public string Name { get; set; }

public string Fac { get; set; }

public string Group { get; set; }

public string Cours { get; set; }

public string Nums { get; set; }

public int[] Ocen { get; set; }

public int CompareTo(object obj)

{

return Fac.CompareTo(((Student)obj).Fac);

}

public override string ToString()

{

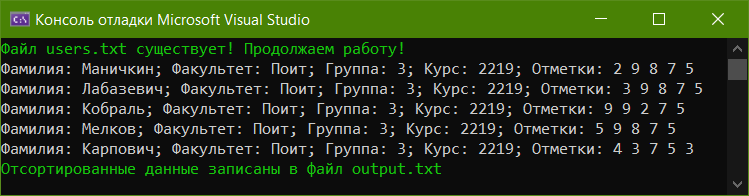
return string.Format($"Фамилия: {Name}; Факультет: {Fac}; Группа: {Group}; Курс: {Cours}; Отметки: {Nums}");

}

}

}

Результат выполнения программы Рис 2.1:



**Рисунок 2.1 – результат программы 9.2.**

Отметка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись преподавателя\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_