Лабораторная работа №9:

№1:

using System;

using System.Diagnostics;

using System.IO;

using static System.Console;

namespace Task01

{

class Program

{

class StudentGroup

{

public string CodeName;

public Student[] Students;

public StudentGroup(string name, Student[] students)

{

CodeName = name;

Students = students;

}

public double AverageMark()

{

double markSum = 0;

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

{

markSum += Students[i].AverageMark();

}

double average = markSum / Students.Length;

return average;

}

public string Info()

{

var info = $"Группа {CodeName}: Ср. балл = {AverageMark()}\n";

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

{

info += $"Студент №{i + 1}: {Students[i].Info()}\n";

}

return info;

}

}

class Student

{

public string Name;

public int[] Marks;

public Student(string name, int[] marks)

{

Name = name;

Marks = marks;

}

public double AverageMark()

{

int markSum = 0;

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

{

markSum += Marks[i];

}

double averageMark = (double)markSum / Marks.Length;

return averageMark;

}

public string Info()

{

var info = $"{Name}: ср. оц. = {AverageMark()}; все оценки: ";

info += Marks[0].ToString();

for (int i = 1; i < Marks.Length; i++)

{

info += $", {Marks[i]}";

}

return info;

}

}

class MasterDegreeStudent : Student

{

public string GraduationWorkName;

public MasterDegreeStudent(string name, int[] marks, string graduationWorkName) : base(name, marks)

{

GraduationWorkName = graduationWorkName;

}

public string Info()

{

return base.Info() + $"; Вып. работа: {GraduationWorkName}";

}

}

static string[] \_names = new string[]

{

"Михаил ",

"Татьяна ",

"Алексей ",

"Виктория",

"Виталий ",

"Азамат ",

"Любава ",

};

static string[] \_works = new string[]

{

"Новые методы ввода текста",

"Ускорение обучения школьников",

"Сравнительный анализ раскладок",

"Новое решение тепловых задач",

"Язык программирования фотонных ВМ",

};

static Random \_random = new Random();

static int MinValue = 1;

static int MaxValue = 5 + 1; //Random.Next не включает последее число

static void Main(string[] args)

{

var studentGroups = ReadGroupsFromFile("input.txt");

PrintGroups(studentGroups, "\n\nГруппы, извлечённые из файла input.txt:");

SortByAverageMark(ref studentGroups);

PrintGroups(studentGroups, "\n\nГруппы, упорядоченные по успеваемости");

WriteGroupsToFile(studentGroups, "output.txt");

Process.Start("input.txt");

Process.Start("output.txt");

}

static void SortByAverageMark(ref StudentGroup[] groups)

{

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

{

for (int j = 0; j < groups.Length - 1; j++)

{

if (groups[j].AverageMark() < groups[j + 1].AverageMark())

{

var temp = groups[j + 1];

groups[j + 1] = groups[j];

groups[j] = temp;

}

}

}

}

static void PrintGroups(StudentGroup[] groups, string message = "")

{

if (message != "")

{

WriteLine(message);

}

if (groups.Length == 0)

{

WriteLine("\* Массив пуст \*");

return;

}

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

{

WriteLine($"{i + 1}: {groups[i].Info()}");

}

}

static StudentGroup[] ReadGroupsFromFile(string filename)

{

var reader = new StreamReader(filename);

var line = reader.ReadLine();

int groupCount = int.Parse(line.Split('=')[1]);

var groups = new StudentGroup[groupCount];

int index = -1;

while ((line = reader.ReadLine()) != null)

{

if (line == "")

{

index++;

continue;

}

if (line.Contains("group="))

{

var name = line.Split('=')[1];

line = reader.ReadLine();

int studentCount = int.Parse(line.Split('=')[1]);

var students = new Student[studentCount];

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

{

line = reader.ReadLine();

var info = line.Split(' ');

var marks = new int[info.Length - 1];

for (int j = 0; j < marks.Length; j++)

{

marks[j] = int.Parse(info[j + 1]);

}

students[i] = new Student(info[0], marks);

}

groups[index] = new StudentGroup(name, students);

}

}

reader.Close();

return groups;

}

static void WriteGroupsToFile(StudentGroup[] groups, string filename)

{

//false - вместо добавления новых данных в старый файл, файл будет перезаписан

var writer = new StreamWriter(filename, false);

writer.WriteLine($"count={groups.Length}\n");

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

{

writer.WriteLine($"group={groups[i].CodeName}");

writer.WriteLine($"count={groups[i].Students.Length}");

for (int j = 0; j < groups[i].Students.Length; j++)

{

writer.Write(groups[i].Students[j].Name);

for (int k = 0; k < groups[i].Students[j].Marks.Length; k++)

{

writer.Write($" {groups[i].Students[j].Marks[k]}");

}

writer.Write('\n');

}

if (i != groups.Length - 1)

{

writer.Write('\n');

}

}

writer.Close();

}

}

}

№2:

using System;

using System.Diagnostics;

using System.IO;

using static System.Console;

namespace Task02

{

class Program

{

class MatchResult

{

public string FirstTeam;

public string SecondTeam;

public int FirstTeamGoals;

public int SecondTeamGoals;

public MatchResult(string firstTeam, string secondTeam, int firstTeamGoals, int secondTeamGoals)

{

FirstTeam = firstTeam;

SecondTeam = secondTeam;

FirstTeamGoals = firstTeamGoals;

SecondTeamGoals = secondTeamGoals;

}

public string Info()

{

return $" {FirstTeam}\t| {FirstTeamGoals} : {SecondTeamGoals} |\t{SecondTeam} ";

}

}

class Team

{

public string Name;

public string CodeName;

}

class FootballTeam : Team

{

public int EarnedPoints;

public int GoalsDifference;

public FootballTeam(string name, string codeName)

{

Name = name;

CodeName = codeName;

EarnedPoints = 0;

GoalsDifference = 0;

}

public bool IsStronger(FootballTeam otherTeam)

{

if (EarnedPoints == otherTeam.EarnedPoints)

{

return GoalsDifference > otherTeam.GoalsDifference;

}

else

{

return EarnedPoints > otherTeam.EarnedPoints;

}

}

public string Info()

{

return $"{Name} ({CodeName}): Очков = {EarnedPoints}; Г-П = {GoalsDifference}";

}

}

static Random \_random = new Random();

static int MinGoals = 0;

static int MaxGoals = 10;

static string[] \_teamNames = new string[]

{

"Manchester",

"Real Madrid",

"Barcelona",

"Uvensus",

"Arsenal",

};

static void Main(string[] args)

{

var teams = ReadNewMatchFromFile("input.txt");

SortByEarnedPoints(ref teams);

PrintTeams(teams, "\nИтоговая таблица:");

WriteTeams(teams, "output.txt");

Process.Start("input.txt");

Process.Start("output.txt");

}

static void SortByEarnedPoints(ref FootballTeam[] teams)

{

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

{

for (int j = 0; j < teams.Length - 1; j++)

{

if (!teams[j].IsStronger(teams[j + 1]))

{

var temp = teams[j + 1];

teams[j + 1] = teams[j];

teams[j] = temp;

}

}

}

}

static void CreateRandomTournament(ref FootballTeam[] teams)

{

const int matchCount = 16;

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

{

int firstTeamGoals = \_random.Next(MinGoals, MaxGoals);

int secondTeamGoals = \_random.Next(MinGoals, MaxGoals);

int firstTeam = \_random.Next(\_teamNames.Length);

int secondTeam = \_random.Next(\_teamNames.Length);

while (secondTeam == firstTeam)

{

secondTeam = \_random.Next(\_teamNames.Length);

}

WriteLine(new MatchResult(

\_teamNames[firstTeam],

\_teamNames[secondTeam],

firstTeamGoals,

secondTeamGoals).Info());

teams[firstTeam].GoalsDifference += firstTeamGoals - secondTeamGoals;

teams[secondTeam].GoalsDifference += secondTeamGoals - firstTeamGoals;

if (firstTeamGoals > secondTeamGoals)

{

teams[firstTeam].EarnedPoints += 3;

}

else if (secondTeamGoals > firstTeamGoals)

{

teams[secondTeam].EarnedPoints += 3;

}

else

{

teams[firstTeam].EarnedPoints += 1;

teams[secondTeam].EarnedPoints += 1;

}

}

}

static FootballTeam[] DeclareAllTeams()

{

var teams = new FootballTeam[\_teamNames.Length];

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

{

teams[i] = new FootballTeam(\_teamNames[i], \_random.Next(0, 100).ToString("0000"));

}

return teams;

}

static void PrintTeams(FootballTeam[] teams, string message = "")

{

if (message != "")

{

WriteLine(message);

}

if (teams.Length == 0)

{

WriteLine("\* Массив пуст \*");

}

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

{

WriteLine($"{i + 1}-е место: " + teams[i].Info());

}

}

static FootballTeam[] ReadNewMatchFromFile(string filename)

{

var reader = new StreamReader(filename);

var line = reader.ReadLine();

int teamCount = int.Parse(line.Split('=')[1]);

var teams = new FootballTeam[teamCount];

reader.ReadLine();

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

{

line = reader.ReadLine();

var tokens = line.Split('=');

var name = tokens[0];

var code = tokens[1];

teams[i] = new FootballTeam(name, code);

}

reader.ReadLine();

line = reader.ReadLine();

int matchCount = int.Parse(line.Split('=')[1]);

reader.ReadLine();

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

{

line = reader.ReadLine();

var tokens = line.Split('|');

var goals = tokens[1].Split(':');

int firstTeamGoals = int.Parse(goals[0].Trim());

int secondTeamGoals = int.Parse(goals[1].Trim());

int firstTeam = IndexByName(teams, tokens[0].Trim());

int secondTeam = IndexByName(teams, tokens[2].Trim());

WriteLine(new MatchResult(

\_teamNames[firstTeam],

\_teamNames[secondTeam],

firstTeamGoals,

secondTeamGoals).Info());

teams[firstTeam].GoalsDifference += firstTeamGoals - secondTeamGoals;

teams[secondTeam].GoalsDifference += secondTeamGoals - firstTeamGoals;

if (firstTeamGoals > secondTeamGoals)

{

teams[firstTeam].EarnedPoints += 3;

}

else if (secondTeamGoals > firstTeamGoals)

{

teams[secondTeam].EarnedPoints += 3;

}

else

{

teams[firstTeam].EarnedPoints += 1;

teams[secondTeam].EarnedPoints += 1;

}

}

return teams;

}

static void WriteTeams(FootballTeam[] teams, string fileName)

{

var writer = new StreamWriter(fileName, false);

if (teams.Length == 0)

{

writer.WriteLine("\* Массив пуст \*");

}

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

{

writer.WriteLine($"{i + 1}-е место: " + teams[i].Info());

}

writer.Close();

}

static int IndexByName(FootballTeam[] teams, string name)

{

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

{

if (name == teams[i].Name)

{

return i;

}

}

return -1;

}

}

}