Лабораторная работа №6:

№1:

using System;

using static System.Console;

namespace L1\_task01

{

class Program

{

struct 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;

}

}

static string[] \_names = new string[]

{

"Михаил ",

"Татьяна ",

"Алексей ",

"Виктория",

"Виталий ",

"Азамат ",

"Любава ",

};

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 studends = RandomUserStudents();

PrintStudents(studends, "Сформированный массив студентов:");

var goodStudents = FindAllGoodStudents(studends);

PrintStudents(goodStudents, "Студенты, чей средный балл не менее 4:");

SortByAverageMark(ref goodStudents);

PrintStudents(goodStudents, "Упорядоченный список студентов по убыванию:");

}

static void SortByAverageMark(ref Student[] students)

{

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

{

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

{

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

{

var temp = students[j + 1];

students[j + 1] = students[j];

students[j] = temp;

}

}

}

}

static Student[] FindAllGoodStudents(Student[] allStudents)

{

int goodStudentCount = 0;

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

{

if (allStudents[i].AverageMark() >= 4)

{

goodStudentCount++;

}

}

var goodStudents = new Student[goodStudentCount];

int index = -1;

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

{

if (allStudents[i].AverageMark() >= 4)

{

index++;

goodStudents[index] = allStudents[i];

}

}

return goodStudents;

}

static Student[] RandomUserStudents()

{

Write("Введите кол-во студентов: ");

int studentCount = int.Parse(ReadLine());

var students = new Student[studentCount];

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

{

students[i] = RandomStudent();

}

return students;

}

static Student RandomStudent()

{

var name = \_names[\_random.Next(\_names.Length)];

var marks = new int[4];

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

{

marks[i] = \_random.Next(MinValue, MaxValue);

}

return new Student(name, marks);

}

static void PrintStudents(Student[] students, string message = "")

{

if (message != "")

{

WriteLine(message);

}

if (students.Length == 0)

{

WriteLine("\* Массив пуст \*");

return;

}

WriteLine($"Всего {students.Length} студентов:");

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

{

WriteLine($"{i + 1}-й студент: {students[i].Info()}");

}

}

}

}

№2:

using System;

using static System.Console;

namespace L2\_task02

{

class Program

{

struct Preparatoriest

{

public string Name;

public int[] Marks;

public Preparatoriest(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 int LowestMark()

{

int lowestMark = Marks[0];

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

{

if (lowestMark > Marks[i])

{

lowestMark = Marks[i];

}

}

return lowestMark;

}

public string Info()

{

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

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

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

{

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

}

return info;

}

}

static string[] \_names = new string[]

{

"Михаил ",

"Татьяна ",

"Алексей ",

"Виктория",

"Виталий ",

"Азамат ",

"Любава ",

};

static Random \_random = new Random();

static int MinValue = 2;

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

static void Main(string[] args)

{

var studends = RandomUserPreparatoriests();

PrintPreparatoriests(studends, "Сформированный массив учащихся:");

var goodStudents = FindAllSuccessPreparatoriests(studends);

PrintPreparatoriests(goodStudents, "Ученики успешно сдавшие экзамены:");

SortByAverageMark(ref goodStudents);

PrintPreparatoriests(goodStudents, "Упорядоченный список учеников по убыванию:");

}

static void SortByAverageMark(ref Preparatoriest[] students)

{

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

{

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

{

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

{

var temp = students[j + 1];

students[j + 1] = students[j];

students[j] = temp;

}

}

}

}

static Preparatoriest[] FindAllSuccessPreparatoriests(Preparatoriest[] allPreparatoriests)

{

int successPreparatoriestCount = 0;

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

{

if (allPreparatoriests[i].LowestMark() > 2)

{

successPreparatoriestCount++;

}

}

var goodPreparatoriests = new Preparatoriest[successPreparatoriestCount];

int index = -1;

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

{

if (allPreparatoriests[i].LowestMark() > 2)

{

index++;

goodPreparatoriests[index] = allPreparatoriests[i];

}

}

return goodPreparatoriests;

}

static Preparatoriest[] RandomUserPreparatoriests()

{

Write("Введите кол-во учащихся: ");

int studentCount = int.Parse(ReadLine());

var students = new Preparatoriest[studentCount];

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

{

students[i] = RandomPreparatoriest(3);

}

return students;

}

static Preparatoriest RandomPreparatoriest(int markCount)

{

var name = \_names[\_random.Next(\_names.Length)];

var marks = new int[markCount];

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

{

marks[i] = \_random.Next(MinValue, MaxValue);

}

return new Preparatoriest(name, marks);

}

static void PrintPreparatoriests(Preparatoriest[] students, string message = "")

{

if (message != "")

{

WriteLine(message);

}

if (students.Length == 0)

{

WriteLine("\* Массив пуст \*");

return;

}

WriteLine($"Всего {students.Length} учеников:");

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

{

WriteLine($"{i + 1}-й ученик: {students[i].Info()}");

}

}

}

}

№3:

using System;

using static System.Console;

namespace L2\_task03

{

class Program

{

struct JumpSportsmen

{

public string Surname;

public int[] AllJumpResults;

public JumpSportsmen(string surname, int[] allJumps)

{

Surname = surname;

AllJumpResults = allJumps;

}

public int BestResult()

{

int bestResult = AllJumpResults[0];

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

{

if (AllJumpResults[i] > bestResult)

{

bestResult = AllJumpResults[i];

}

}

return bestResult;

}

public string Info()

{

var info = $"{Surname}: л. р. = {BestResult()}м; Все прыжки: ";

info += AllJumpResults[0];

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

{

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

}

return info;

}

}

static string[] \_surnames = new string[]

{

"Мордвинов",

"Банникова",

"Быстрых ",

"Бородач ",

"Тимошенко",

"Мельников",

"Тимофеева",

};

static Random \_random = new Random();

static int MinValue = 10;

static int MaxValue = 25;

static void Main(string[] args)

{

var sportsmens = RandomUserJumpSportsmens();

PrintJumpSportsmens(sportsmens, "Изначальный список спортсменов:");

SortByBestJump(ref sportsmens);

PrintJumpSportsmens(sportsmens, "\nПротокол соревнований: Таблица лидеров:");

}

static void SortByBestJump(ref JumpSportsmen[] sportsmens)

{

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

{

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

{

if (sportsmens[j].BestResult() < sportsmens[j + 1].BestResult())

{

var temp = sportsmens[j + 1];

sportsmens[j + 1] = sportsmens[j];

sportsmens[j] = temp;

}

}

}

}

static JumpSportsmen[] RandomUserJumpSportsmens()

{

Write("Введите кол-во спортсменов: ");

int studentCount = int.Parse(ReadLine());

var sportsmens = new JumpSportsmen[studentCount];

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

{

sportsmens[i] = RandomJumpSportsmen(3);

}

return sportsmens;

}

static JumpSportsmen RandomJumpSportsmen(int jumpCount)

{

var name = \_surnames[\_random.Next(\_surnames.Length)];

var jumps = new int[jumpCount];

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

{

jumps[i] = \_random.Next(MinValue, MaxValue);

}

return new JumpSportsmen(name, jumps);

}

static void PrintJumpSportsmens(JumpSportsmen[] sportsmens, string message = "")

{

if (message != "")

{

WriteLine(message);

}

if (sportsmens.Length == 0)

{

WriteLine("\* Массив пуст \*");

return;

}

WriteLine($"Всего {sportsmens.Length} спортсменов:");

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

{

WriteLine($"{i + 1}-й спортсмен: {sportsmens[i].Info()}");

}

}

}

}

3-й уровень:

№1:

using System;

using static System.Console;

namespace L3\_task01

{

class Program

{

struct 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;

}

}

struct 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;

}

}

static string[] \_names = new string[]

{

"Михаил ",

"Татьяна ",

"Алексей ",

"Виктория",

"Виталий ",

"Азамат ",

"Любава ",

};

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 = UserGroups(3);

SortByAverageMark(ref studentGroups);

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

}

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 StudentGroup[] UserGroups(int count)

{

var groups = new StudentGroup[count];

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

{

WriteLine($"Группа №{i + 1}:");

Write($"Название группы: ");

var groupName = ReadLine();

var students = RandomUserStudents();

groups[i] = new StudentGroup(groupName, students);

}

return groups;

}

static Student[] RandomUserStudents()

{

Write("Введите кол-во студентов: ");

int studentCount = int.Parse(ReadLine());

var students = new Student[studentCount];

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

{

students[i] = RandomStudent();

}

return students;

}

static Student RandomStudent()

{

var name = \_names[\_random.Next(\_names.Length)];

var marks = new int[4];

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

{

marks[i] = \_random.Next(MinValue, MaxValue);

}

return new Student(name, marks);

}

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()}");

}

}

}

}

№4:

using System;

using static System.Console;

namespace L3\_task04

{

class Program

{

struct 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}| {FirstTeamGoals} : {SecondTeamGoals} |{SecondTeam}";

}

}

struct FootballTeam

{

public string Name;

public int EarnedPoints;

public int GoalsDifference;

public FootballTeam(string name)

{

Name = name;

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}: Очков = {EarnedPoints}; Г-П = {GoalsDifference}";

}

}

static Random \_random = new Random();

static int MinGoals = 0;

static int MaxGoals = 10;

static string[] \_teamNames = new string[]

{

"Manchester United",

"Real Madrid ",

"Barcelona ",

"Uvensus ",

"Arsenal ",

};

static void Main(string[] args)

{

var teams = DeclareAllTeams();

CreateRandomTournament(ref teams);

SortByEarnedPoints(ref teams);

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

}

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]);

}

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());

}

}

}

}

№6:

using System;

using static System.Console;

using static System.Math;

namespace L3\_task06

{

class Program

{

struct AnswerTable

{

public Answers[] AllAnswers;

public AnswerTable(Answers[] allAnswers)

{

AllAnswers = allAnswers;

}

public string AllAnswersInfo()

{

var info = "\n| Животное | Черта х. | Вещь |\n";

info += "======================================\n";

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

{

info += $"|{AllAnswers[i].JapanAndAnimal}|{AllAnswers[i].JapanAndPersonCharacter}|{AllAnswers[i].JapanAndThing}|\n";

}

info += "======================================\n";

return info;

}

public string[] GetMostPopularAnimalsAndPercentage(int count)

{

var allAnimals = new string[AllAnswers.Length];

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

{

allAnimals[i] = AllAnswers[i].JapanAndAnimal;

}

var animalsCounts = new int[7];

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

{

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

{

if (allAnimals[i] == \_animalAnswers[j])

{

animalsCounts[j]++;

}

}

}

Array.Sort(animalsCounts, \_animalAnswers);

Array.Sort(animalsCounts);

var mostPopularAnimals = new string[count];

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

{

double percentage = (double) animalsCounts[i] / AllAnswers.Length \* 100d;

mostPopularAnimals[mostPopularAnimals.Length - 1 - i] = $" {\_animalAnswers[i]}, {percentage:N1}% ";

}

return mostPopularAnimals;

}

public string[] GetMostPopularCharactersAndPercentage(int count)

{

var allCharacters = new string[AllAnswers.Length];

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

{

allCharacters[i] = AllAnswers[i].JapanAndPersonCharacter;

}

var charactersCounts = new int[7];

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

{

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

{

if (allCharacters[i] == \_characterAnswers[j])

{

charactersCounts[j]++;

}

}

}

Array.Sort(charactersCounts, \_characterAnswers);

Array.Sort(charactersCounts);

var mostPopularCharacters = new string[count];

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

{

double percentage = (double)charactersCounts[i] / AllAnswers.Length \* 100d;

mostPopularCharacters[mostPopularCharacters.Length - 1 - i] = $" {\_characterAnswers[i]}, {percentage:N1}% ";

}

return mostPopularCharacters;

}

public string[] GetMostPopularThingsAndPercentage(int count)

{

var allThings = new string[AllAnswers.Length];

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

{

allThings[i] = AllAnswers[i].JapanAndThing;

}

var thingsCounts = new int[7];

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

{

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

{

if (allThings[i] == \_thingAnwers[j])

{

thingsCounts[j]++;

}

}

}

Array.Sort(thingsCounts, \_thingAnwers);

Array.Sort(thingsCounts);

var mostPopularThings = new string[count];

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

{

double percentage = (double)thingsCounts[i] / AllAnswers.Length \* 100d;

mostPopularThings[mostPopularThings.Length - 1 - i] = $" {\_thingAnwers[i]}, {percentage:N1}% ";

}

return mostPopularThings;

}

}

struct Answers

{

public string JapanAndAnimal;

public string JapanAndPersonCharacter;

public string JapanAndThing;

public Answers(string japanAndAnimal, string japanAndPersonCharacter, string japanAndThing)

{

JapanAndAnimal = japanAndAnimal;

JapanAndPersonCharacter = japanAndPersonCharacter;

JapanAndThing = japanAndThing;

}

}

static string[] \_animalAnswers =

{

" Журавль ",

" Выдра ",

" Шиба Ину ",

" Лисица ",

" Заяц ",

" Барсук ",

" -------- "

};

static string[] \_characterAnswers =

{

" Трудолюбие ",

" Уважение ",

" Честность ",

" Синтоизм ",

" Традиции ",

" Скромность ",

" ---------- "

};

static string[] \_thingAnwers =

{

" Катана ",

" Панасоник ",

" Рикша ",

" Магнитофон ",

" Сюрикен ",

" Манга ",

" ---------- "

};

static Random \_random = new Random();

static int MaxIndex = \_characterAnswers.Length;

static void Main(string[] args)

{

var table = new AnswerTable(RandomAnswers(20));

WriteLine("Все сгенерированные ответы:");

WriteLine(table.AllAnswersInfo());

WriteLine("5 самых популярных ответов:");

PrintMostPopularAnswers(

table.GetMostPopularAnimalsAndPercentage(5),

table.GetMostPopularCharactersAndPercentage(5),

table.GetMostPopularThingsAndPercentage(5));

}

static Answers[] RandomAnswers(int count)

{

var answers = new Answers[count];

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

{

answers[i] = new Answers(

\_animalAnswers[\_random.Next(MaxIndex)],

\_characterAnswers[\_random.Next(MaxIndex)],

\_thingAnwers[\_random.Next(MaxIndex)]);

}

return answers;

}

static void PrintMostPopularAnswers(string[] animals, string[] characters, string[] things)

{

var length = animals.Length;

var info = "| СП животные | СП черты хар. | СП вещи |\n";

info += "=========================================================================\n";

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

{

info += $"|{animals[i]}\t|{characters[i]}\t|{things[i]}\t|\n";

}

info += "=========================================================================\n";

WriteLine($"\n{info}\n");

}

}

}