Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №6
по дисциплине «Программирование и Алгоритмизация»
на тему «Структуры»

Выполнил:
студент группы БИВТ-24-5

Черных Богдан

Проверил:
Стучилин В. В.

Москва, 2024

Теоретическое введение. *Структуры* (*классы*) являются двумя основными типами в *C#*. Структура (класс) представляющая собой объектный тип данных, внешне похожа на типы данных процедурно-ориентированных языков, такие как *структура* в языке Си или *запись* в языке Паскаль. При этом элементы такой структуры (класса) могут сами быть не только данными, но и *методами* (т.е. процедурами или функциями). Такое объединение называется инкапсуляцией (см. выше).

Структуры (классы) являются типами, создаваемыми (определяемыми) пользователем, и представляют собой составной тип данных, имеющий в составе:

- *Поля данных* - параметры объекта (конечно, не все, а только необходимые в программе), задающие его состояние (свойства объекта предметной области). Иногда поля данных объекта называют свойствами объекта, из-за чего возможна путаница. Физически поля представляют собой значения (переменные, константы), объявленные как принадлежащие структуре (классу);
- *Методы* - процедуры и функции, связанные с классом. Они определяют действия, которые можно выполнять над объектом такого типа, и которые сам объект может выполнять.

Структура является типом значения. Класс является ссылочным типом (см. выше). Далее рассматривается работа со структурами и с классами (определение, создание объектов соответствующего типа, различные способы инициализации полей и т. п.). Для решения многих задач можно использовать как классы, так и структуры. Однако классы имеют более широкое применение. В частности, классы допускают наследование, что позволяет на базе одного (базового) класса без особых затрат создавать различные производные классы (см. ниже).

Сам код заданий 1, 2, 3 уровня:

```
//svg does precious
using System;                    // Аналог <iostream> для работы с консолью и основными функциями
using System.Collections.Generic;       // Аналог <vector>, <list>, <map>, <set>, <unordered_map>, <unordered_set>, <stack>, <queue>
using System.Text;                  // Аналог <string>, <cstring> (для работы со строками и StringBuilder)
using System.Linq;                  // Аналог <algorithm> (для работы с LINQ, сортировок, поиска и т.д.)
using System.IO;                   // Аналог <cstdio>, <fstream> (работа с файлами)
using System.Globalization;            // Аналог <iomanip> (для форматирования)
```

```
using System.Collections;              // Работа с различными коллекциями (например, ArrayList)
using System.Threading;                // Потоки и многопоточность
using System.Runtime.Serialization;    // Аналог <stdexcept> (работа с исключениями)
using System.Reflection;               // Аналог <typeinfo> (информация о типах, рефлексия)
using System.Diagnostics;              // Аналог <utility>, <std::pair> (вспомогательные функции и классы)
using System.ComponentModel;           // Дополнительные утилиты и атрибуты
using System.Numerics;                 // Работа с большими числами и математическими операциями
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
// Для работы с потоками данных:
using System.Threading.Tasks;          // Асинхронные задачи

// Для работы с датами и временем:
using System.Timers;                   // Для работы с таймерами и временем
using System.Collections.Generic;
using System.Text;
using System.Linq;
using System.IO; //important
using System.Globalization;
using System.Collections;
using System.Threading;
using System.Runtime.Serialization;
using System.Reflection;
using System.Diagnostics;
using System.ComponentModel;
using System.Numerics;
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
```

```
using System.Xml.Linq;
using System.Runtime.InteropServices;
using System.Security;
using System.Web;
using System.Media;
using System.Drawing;
using System.Configuration;
using System.Timers;
using System.Runtime.Remoting;
using System.Runtime.CompilerServices;
using System.Runtime.Versioning;
using System.CodeDom;
using System.CodeDom.Compiler;
using System.Collections.Concurrent;
using System.Runtime;
using System.Windows;
using System.Windows.Input;
using System.Security.Principal;
using System.Security.Permissions;
using System.Resources;
using C = System.Console; //console
using dl = System.Decimal;//decimal
using str = System.String;//string
using l = System.Int64;   //long
using u = System.UInt64;  //Ulong
using db = System.Double; //Double

/*
Izzspot - 19 years
Boogie B - 20 years
SJ - 21 years
Bandokay - life sentence

Youngest in the charge
OFB, we don't window shop
Bro caught him an opp and tried turn him off (Bow, bow)
In this X3, man's swervin' off (Skrr, skrr)
Free Boogie Bando, he got birded off (Free my bro, free my bro)
Whenever we get a burner loss
We just cop a next one and go burst it off (Ay)
Lil bro's tellin' me he got his earnings wrong
We just took him OT, now his trapline's gone (Ring, ring)
```

```
Hashtag
Bro backed this ting and just started squeezin' (Clarted)
When it broad day, it was freezin'
Hashtag fuckery, hashtag screamin'
One on the hand ting woi, left man leanin', leanin' (Fucker)
Show us cause it's good to feel it
Shortie's cooze and she must be dreamin'
*/
/* ¡Adelante Barcelona, adelante Cataluña! Visca el Barça! Visca Catalunya!
   ¡Al diablo con todos los demás, porque lo más importante en la vida es el fútbol!
*/
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------
//--------------------------------------------------------------------------------


class Program
{
    static void Main()
    {
        Console.WriteLine("1 Уровень Задача 1");
        task11.Run();

        Console.WriteLine("\n1 Уровень Задача 2");
        task12.Run();

        Console.WriteLine("\n1 Уровень Задача 3");
        task13.Run();

        Console.WriteLine("\n2 Уровень Задача 7");
        task27.Run();

        Console.WriteLine("\n2 Уровень Задача 8");
        task28.Run();

        Console.WriteLine("\n2 Уровень Задача 9");
        task29.Run();

        Console.WriteLine("3 Уровень Задача 3");
        task33.Run();

        Console.WriteLine("\n3 Уровень Задача 4");
```

```
        task34.Run();

        Console.WriteLine("\n3 Уровень Задача 5");
        task35.Run();
    }
}

// Задача 1: Соревнования по прыжкам в длину
class task11
{
    struct Participant
    {
        public string LastName;
        public string Club;
        public double Attempt1;
        public double Attempt2;
        public double TotalScore;

        public Participant(string lastName, string club, double attempt1, double
attempt2)
        {
            LastName = lastName;
            Club = club;
            Attempt1 = attempt1;
            Attempt2 = attempt2;
            TotalScore = attempt1 + attempt2;
        }
    }

    public static void Run()
    {
        var participants = new List<Participant>
        {
            new Participant("Lewandowski", "Spartak", 6.5, 7.1),
            new Participant("Yamal", "Dynamo", 7.2, 6.9),
            new Participant("Raphinha", "Labor", 6.8, 7.3)
        };

        var sortedParticipants = participants.OrderByDescending(p =>
p.TotalScore).ToList();

        Console.WriteLine("Результаты соревнований по прыжкам в длину:");
```

```
        Console.WriteLine("Место\tФамилия\t\tОбщество\tПопытка 1\tПопытка
2\tСумма");
        for (int i = 0; i < sortedParticipants.Count; i++)
        {
            var p = sortedParticipants[i];
            Console.WriteLine($"{i +
1}\t{p.LastName}\t\t{p.Club}\t\t{p.Attempt1:F2}\t\t{p.Attempt2:F2}\t\t{p.TotalS
core:F2}");
        }
    }
}

// Задача 2: Результаты кросса на 500 м для женщин
class task12
{
    struct Runner
    {
        public string LastName;
        public string Group;
        public string Coach;
        public double Result;
        public bool NormativeMet;

        public Runner(string lastName, string group, string coach, double result,
double normative)
        {
            LastName = lastName;
            Group = group;
            Coach = coach;
            Result = result;
            NormativeMet = result <= normative;
        }
    }

    public static void Run()
    {
        double normative = 90.0; // Время в секундах для выполнения норматива
        var runners = new List<Runner>
        {
            new Runner("Lewandowski", "Group 1", "Smirnov", 85.4, normative),
            new Runner("Yamal", "Group 2", "Petrov", 92.1, normative),
            new Runner("Raphinha", "Group 1", "Smirnov", 88.7, normative)
```

```
        };

        var sortedRunners = runners.OrderBy(r => r.Result).ToList();
        int normativeCount = sortedRunners.Count(r => r.NormativeMet);

        Console.WriteLine("Результаты кросса на 500 м для женщин:");

Console.WriteLine("Фамилия\t\tГруппа\t\tПреподаватель\tРезультат\tНормати
в");
        foreach (var r in sortedRunners)
        {

Console.WriteLine($"{r.LastName}\t\t{r.Group}\t{r.Coach}\t\t{r.Result:F1}\t\t{(r
.NormativeMet ? "Выполнен" : "Не выполнен")}");
        }
        Console.WriteLine($"Всего участниц, выполнивших норматив:
{normativeCount}");
    }
}

// Задача 3: Опрос радиокомпании «Человек года»
class task13
{
    struct Vote
    {
        public string Candidate;
        public int Count;

        public Vote(string candidate, int count)
        {
            Candidate = candidate;
            Count = count;
        }
    }

    public static void Run()
    {
        var votes = new List<string> { "Lewandowski", "Yamal", "Raphinha",
"Lewandowski", "Raphinha", "Lewandowski", "Yamal", "Yamal", "Raphinha",
"Raphinha" };
        var voteCounts = votes.GroupBy(v => v)
                              .Select(g => new Vote(g.Key, g.Count()))
```

```
                    .OrderByDescending(v => v.Count)
                    .Take(5)
                    .ToList();

        int totalVotes = votes.Count;
        Console.WriteLine("Результаты опроса 'Человек года':");
        Console.WriteLine("Кандидат\tГолосов\t\tДоля (%)");
        foreach (var v in voteCounts)
        {
            double percentage = (double)v.Count / totalVotes * 100;
            Console.WriteLine($"{v.Candidate}\t\t{v.Count}\t\t{percentage:F2}");
        }
    }
}


// Задача 7: Турнирная таблица по шахматам
class task27
{
    struct Participant
    {
        public string LastName;
        public double[] Results;
        public double TotalScore => Results.Sum();

        public Participant(string lastName, double[] results)
        {
            LastName = lastName;
            Results = results;
        }
    }

    public static void Run()
    {
        var participants = new List<Participant>
        {
            new Participant("Lewandowski", new double[] { 1, 0.5, 1, 0 }),
            new Participant("Yamal", new double[] { 0.5, 1, 1, 1 }),
            new Participant("Raphinha", new double[] { 1, 1, 0, 0.5 })
        };
```

```
        var sortedParticipants = participants.OrderByDescending(p =>
p.TotalScore).ToList();

        Console.WriteLine("Итоговая таблица шахматного турнира:");
        Console.WriteLine("Фамилия\t\tОчки");
        foreach (var p in sortedParticipants)
        {
            Console.WriteLine($"{p.LastName}\t\t{p.TotalScore:F1}");
        }
    }
}

// Задача 8: Список кандидатов в хоккейную сборную
class task28
{
    struct Player
    {
        public string LastName;
        public int[] PenaltyTimes;
        public int TotalPenaltyTime => PenaltyTimes.Sum();

        public Player(string lastName, int[] penaltyTimes)
        {
            LastName = lastName;
            PenaltyTimes = penaltyTimes;
        }

        public bool IsEligible => !PenaltyTimes.Contains(10);
    }

    public static void Run()
    {
        var players = new List<Player>
        {
            new Player("Lewandowski", new int[] { 2, 5, 2 }),
            new Player("Yamal", new int[] { 10, 2, 5 }),
            new Player("Raphinha", new int[] { 5, 2, 5 }),
            new Player("Gavi", new int[] { 2, 2, 2 }),
            new Player("Pedri", new int[] { 2, 5, 2 })
        };

        var eligiblePlayers = players
```

```
            .Where(p => p.IsEligible)
            .OrderBy(p => p.TotalPenaltyTime)
            .ToList();

        Console.WriteLine("Список кандидатов в хоккейную сборную:");
        Console.WriteLine("Фамилия\t\tСуммарное штрафное время");
        foreach (var p in eligiblePlayers)
        {
            Console.WriteLine($"{p.LastName}\t\t{p.TotalPenaltyTime}");
        }
    }
}

// Задача 9: Итоговая таблица соревнований фигуристов
class task29
{
    struct Skater
    {
        public string LastName;
        public double[] Scores;
        public int[] Places;
        public int TotalPlace => Places.Sum();

        public Skater(string lastName, double[] scores)
        {
            LastName = lastName;
            Scores = scores;
            Places = new int[scores.Length];
        }

        public void CalculatePlaces(List<Skater> skaters)
        {
            var currentSkater = this; // Копируем текущий экземпляр в локальную
переменную
            for (int j = 0; j < Scores.Length; j++)
            {
                Places[j] = skaters
                    .OrderByDescending(s => s.Scores[j])
                    .Select((s, index) => new { s, index = index + 1 })
                    .First(x => x.s.Equals(currentSkater)).index;
            }
        }
```

```
    }

    public static void Run()
    {
        var skaters = new List<Skater>
        {
            new Skater("Lewandowski", new double[] { 5.8, 5.9, 6.0, 5.7, 5.8, 6.0, 5.9
}),
            new Skater("Yamal", new double[] { 5.5, 5.6, 5.4, 5.6, 5.5, 5.4, 5.5 }),
            new Skater("Raphinha", new double[] { 5.7, 5.8, 5.9, 5.7, 5.8, 5.9, 5.8 }),
            new Skater("Gavi", new double[] { 5.6, 5.7, 5.8, 5.5, 5.6, 5.7, 5.6 }),
            new Skater("Pedri", new double[] { 5.4, 5.3, 5.5, 5.2, 5.4, 5.3, 5.5 })
        };

        foreach (var skater in skaters)
        {
            skater.CalculatePlaces(skaters);
        }

        var sortedSkaters = skaters.OrderBy(s => s.TotalPlace).ToList();

        Console.WriteLine("Итоговая таблица соревнований фигуристов:");
        Console.WriteLine("Фамилия\t\tСумма мест");
        foreach (var skater in sortedSkaters)
        {
            Console.WriteLine($"{skater.LastName}\t\t{skater.TotalPlace}");
        }
    }
}

// Задача 3: Определение команды-победителя на соревнованиях
class task33
{
    struct Team
    {
        public string Name;
        public int[] Places;
        public int TotalScore;

        public Team(string name, int[] places)
        {
```

```
        Name = name;
        Places = places;
        TotalScore = CalculateScore(places);
    }

    private static int CalculateScore(int[] places)
    {
        int score = 0;
        foreach (var place in places)
        {
            score += place switch
            {
                1 => 5,
                2 => 4,
                3 => 3,
                4 => 2,
                5 => 1,
                _ => 0
            };
        }
        return score;
    }
}

public static void Run()
{
    var teams = new List<Team>
    {
        new Team("Team A", new int[] { 1, 6, 3, 8, 5, 10 }),
        new Team("Team B", new int[] { 2, 7, 4, 9, 12, 11 }),
        new Team("Team C", new int[] { 1, 2, 3, 4, 5, 6 })
    };

    var winningTeam = teams.OrderByDescending(t => t.TotalScore)
                           .ThenBy(t => t.Places.Min())
                           .First();

    Console.WriteLine("Команда-победитель:");
    Console.WriteLine($"Название: {winningTeam.Name}, Очки:
{winningTeam.TotalScore}");
}
}
```

```
// Задача 4: Лыжные гонки для двух групп участников
class task34
{
    struct Participant
    {
        public string LastName;
        public double Result;

        public Participant(string lastName, double result)
        {
            LastName = lastName;
            Result = result;
        }
    }

    public static void Run()
    {
        var group1 = new List<Participant>
        {
            new Participant("Lewandowski", 21.5),
            new Participant("Yamal", 19.7),
            new Participant("Raphinha", 22.8)
        };

        var group2 = new List<Participant>
        {
            new Participant("Gavi", 20.3),
            new Participant("Pedri", 23.1),
            new Participant("Kounde", 21.2)
        };

        var sortedGroup1 = group1.OrderBy(p => p.Result).ToList();
        var sortedGroup2 = group2.OrderBy(p => p.Result).ToList();

        var combinedResults = sortedGroup1.Concat(sortedGroup2).OrderBy(p =>
p.Result).ToList();

        Console.WriteLine("Результаты лыжных гонок:");
        Console.WriteLine("Фамилия\t\tРезультат (секунды)");
        foreach (var participant in combinedResults)
        {
```

```
            Console.WriteLine($"{participant.LastName}\t\t{participant.Result}");
        }
    }
}

// Задача 5: Первенство по футболу
class task35
{
    struct GameResult
    {
        public string TeamName;
        public int GoalsScored;
        public int GoalsConceded;

        public GameResult(string teamName, int goalsScored, int goalsConceded)
        {
            TeamName = teamName;
            GoalsScored = goalsScored;
            GoalsConceded = goalsConceded;
        }
    }

    struct TeamStanding
    {
        public string TeamName;
        public int Points;
        public int GoalDifference;

        public TeamStanding(string teamName, int points, int goalDifference)
        {
            TeamName = teamName;
            Points = points;
            GoalDifference = goalDifference;
        }
    }

    public static void Run()
    {
        var games = new List<GameResult>
        {
            new GameResult("Team A", 2, 1),
            new GameResult("Team B", 0, 0),
```

```
            new GameResult("Team A", 1, 3),
            new GameResult("Team B", 2, 2),
            new GameResult("Team C", 3, 1)
        };

        var standings = CalculateStandings(games);

        var sortedStandings = standings.OrderByDescending(s => s.Points)
                                       .ThenByDescending(s => s.GoalDifference)
                                       .ToList();

        Console.WriteLine("Таблица очков первенства по футболу:");
        Console.WriteLine("Место\tКоманда\t\tОчки\tРазница мячей");
        for (int i = 0; i < sortedStandings.Count; i++)
        {
            var s = sortedStandings[i];
            Console.WriteLine($"{i +
1}\t{s.TeamName}\t\t{s.Points}\t{s.GoalDifference}");
        }
    }

    private static List<TeamStanding> CalculateStandings(List<GameResult>
games)
    {
        var standingsDict = new Dictionary<string, TeamStanding>();

        foreach (var game in games)
        {
            int pointsA = game.GoalsScored > game.GoalsConceded ? 3 :
(game.GoalsScored == game.GoalsConceded ? 1 : 0);
            int pointsB = game.GoalsScored < game.GoalsConceded ? 3 :
(game.GoalsScored == game.GoalsConceded ? 1 : 0);

            UpdateStanding(standingsDict, game.TeamName, pointsA,
game.GoalsScored - game.GoalsConceded);
            UpdateStanding(standingsDict, game.TeamName, pointsB,
game.GoalsConceded - game.GoalsScored);
        }

        return standingsDict.Values.ToList();
    }
```

```
    private static void UpdateStanding(Dictionary<string, TeamStanding>
standingsDict, string teamName, int points, int goalDifference)
    {
        if (standingsDict.ContainsKey(teamName))
        {
            var standing = standingsDict[teamName];
            standing.Points += points;
            standing.GoalDifference += goalDifference;
            standingsDict[teamName] = standing;
        }
        else
        {
            standingsDict[teamName] = new TeamStanding(teamName, points,
goalDifference);
        }
    }
}
```

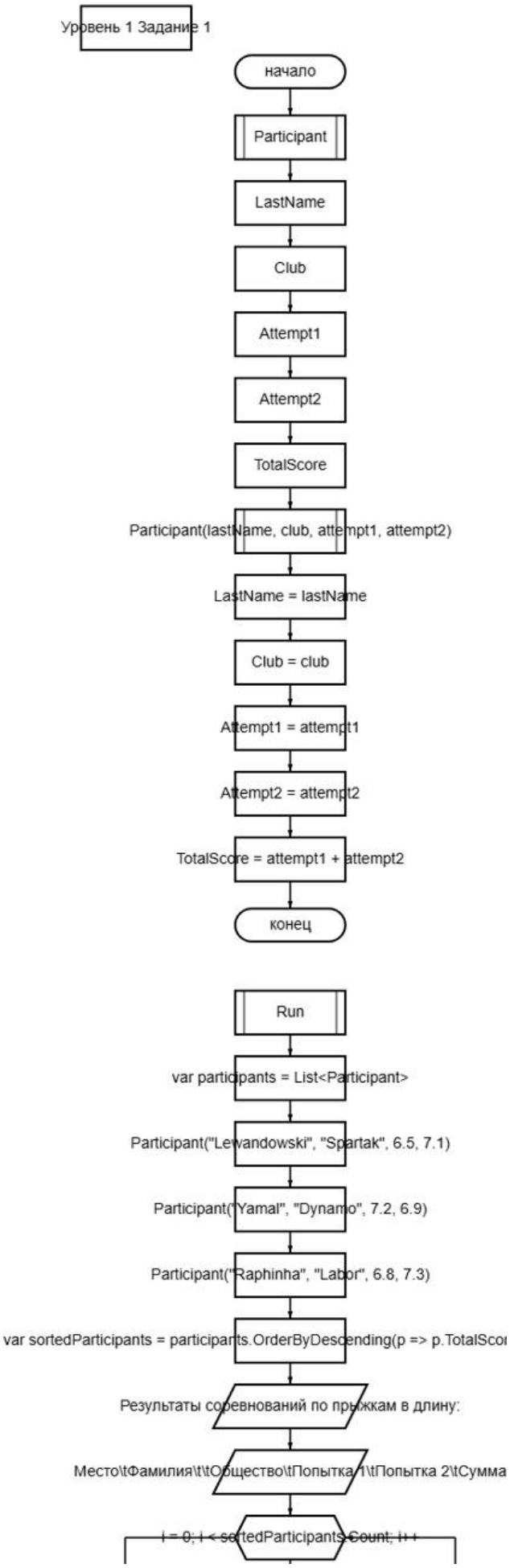
Программа выполнения каждого задания:

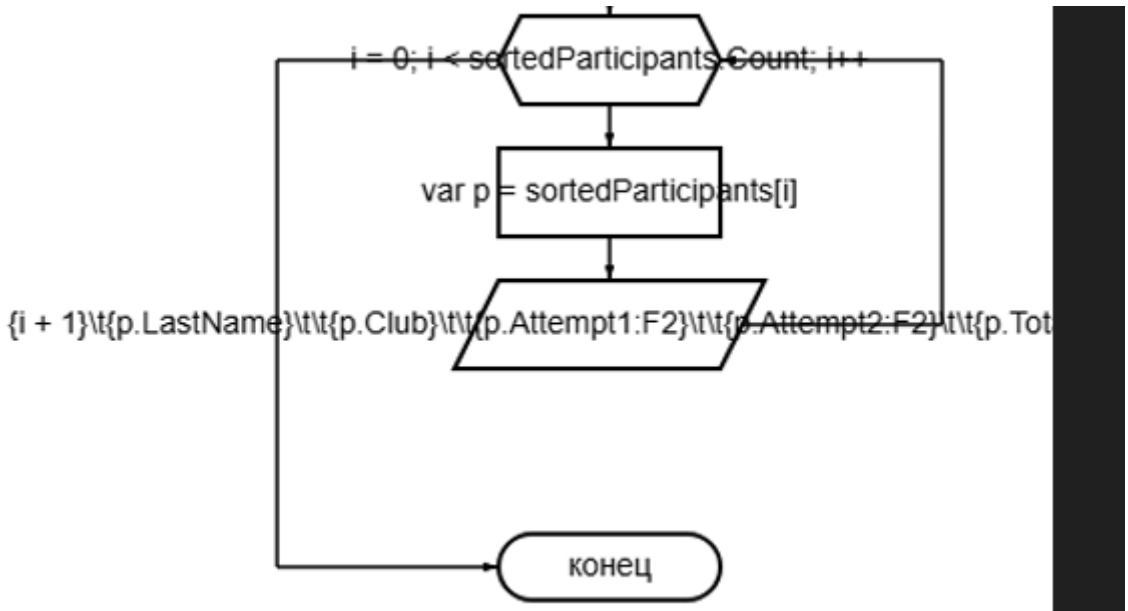
```
1 Уровень Задача 1
Результаты соревнований по прыжкам в длину:
Место   Фамилия         Общество        Попытка 1       Попытка 2       Сумма
1       Yamal           Dynamo          7,20            6,90            14,10
2       Raphinha                Labor           6,80            7,30            14,10
3       Lewandowski             Spartak         6,50            7,10            13,60

1 Уровень Задача 2
Результаты кросса на 500 м для женщин:
Фамилия         Группа          Преподаватель   Результат       Норматив
Lewandowski             Group 1 Smirnov         85,4            Выполнен
Raphinha                Group 1 Smirnov         88,7            Выполнен
Yamal           Group 2 Petrov          92,1            Не выполнен
Всего участниц, выполнивших норматив: 2

1 Уровень Задача 3
Результаты опроса 'Человек года':
Кандидат        Голосов         Доля (%)
Raphinha                4               40,00
Lewandowski             3               30,00
Yamal           3               30,00

2 Уровень Задача 7
Итоговая таблица шахматного турнира:
Фамилия         Очки
Yamal           3,5
Lewandowski             2,5
Raphinha                2,5

2 Уровень Задача 8
Список кандидатов в хоккейную сборную:
Фамилия         Суммарное штрафное время
Gavi            6
Lewandowski             9
Pedri           9
Raphinha                12

2 Уровень Задача 9
Итоговая таблица соревнований фигуристов:
Фамилия         Сумма мест
Lewandowski             7
Raphinha                14
Gavi            22
Yamal           28
Pedri           34
3 Уровень Задача 3
Команда-победитель:
Название: Team C, Очки: 15
```

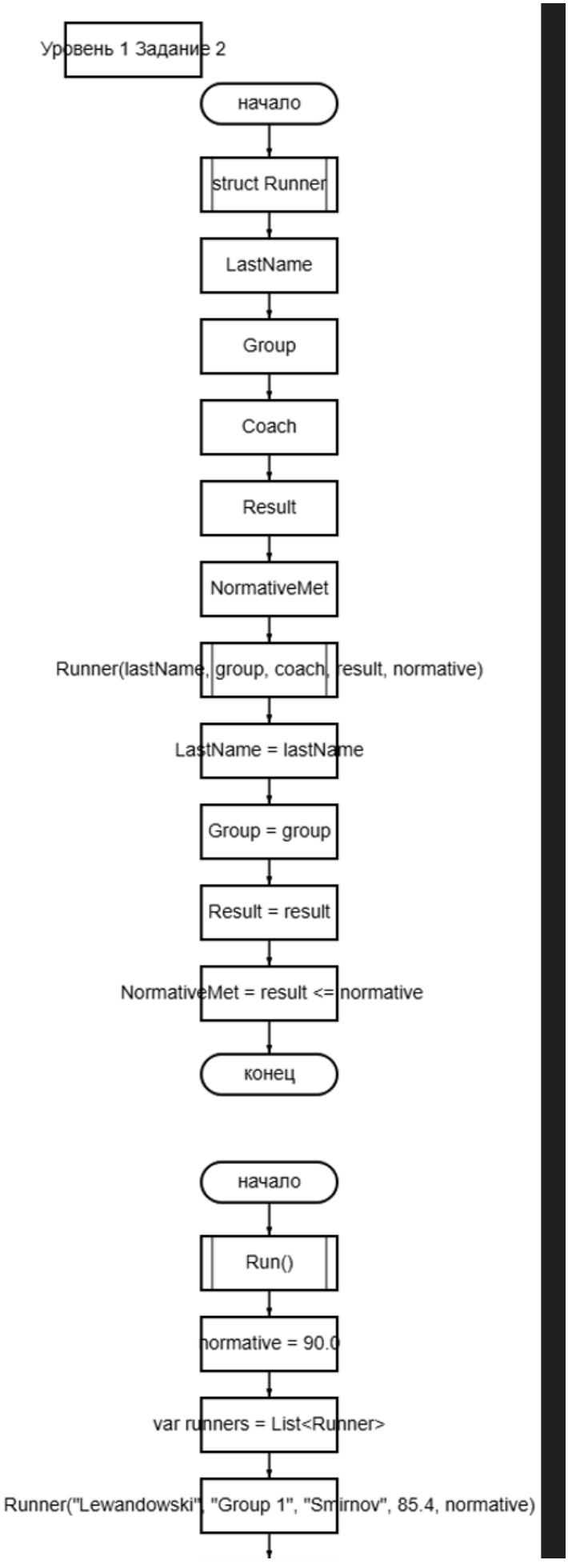
```
3 Уровень Задача 4
Результаты лыжных гонок:
Фамилия          Результат (секунды)
Yamal            19,7
Gavi             20,3
Kounde           21,2
Lewandowski              21,5
Raphinha                 22,8
Pedri            23,1

3 Уровень Задача 5
Таблица очков первенства по футболу:
Место   Команда          Очки    Разница мячей
1       Team A           6       0
2       Team B           4       0
3       Team C           3       0
```

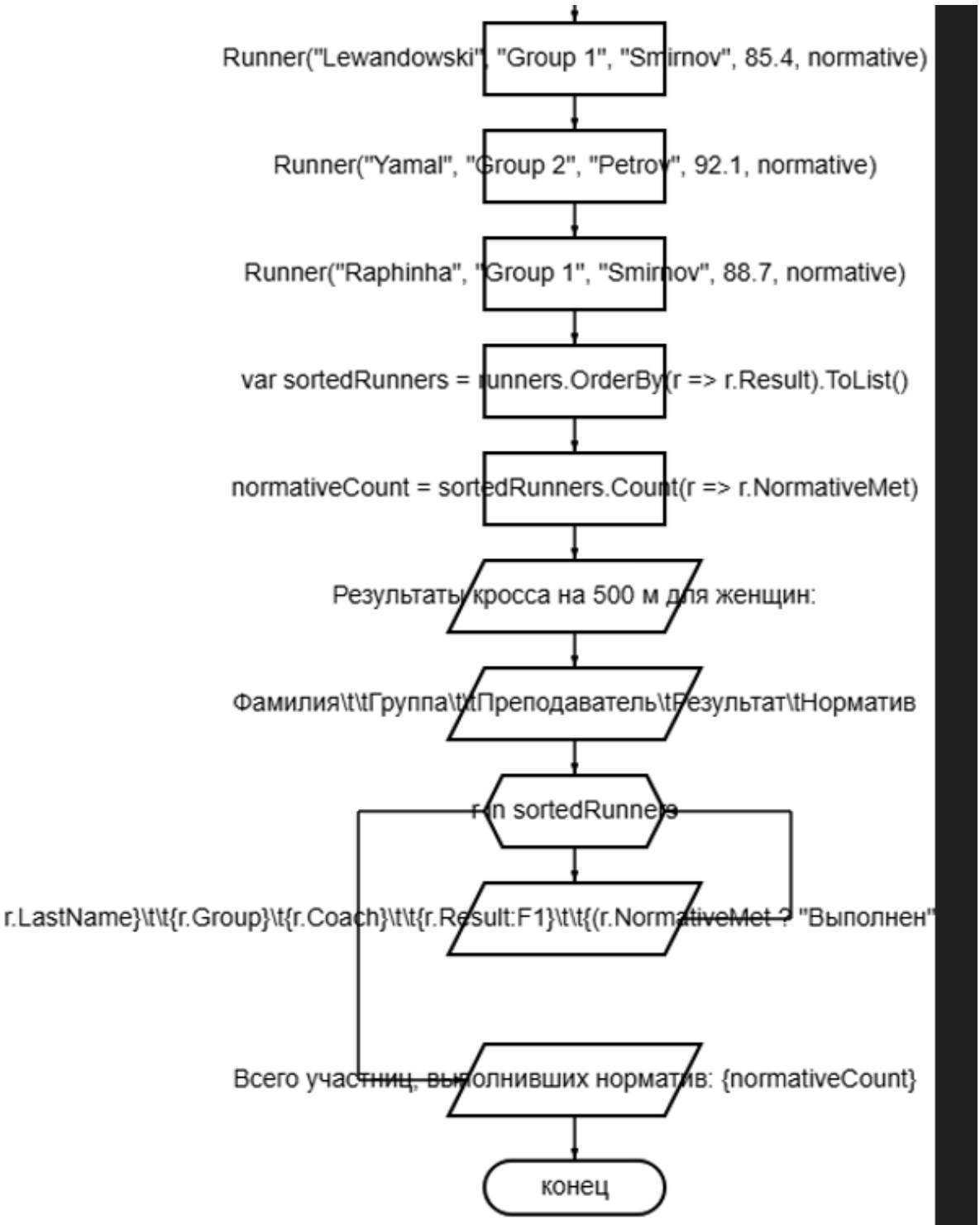
Блок-Схемы для каждого задания:


Лабораторная работа №6
начало
Main()
1 Уровень Задача 1
task11.Run()
1 Уровень Задача 2
task12.Run()
1 Уровень Задача 3
task13.Run()
2 Уровень Задача 7
task27.Run()
2 Уровень Задача 8
task28.Run()
2 Уровень Задача 9
task29.Run()
3 Уровень Задача 3
task33.Run()
3 Уровень Задача 4
task34.Run()
3 Уровень Задача 5
task35.Run()
конец

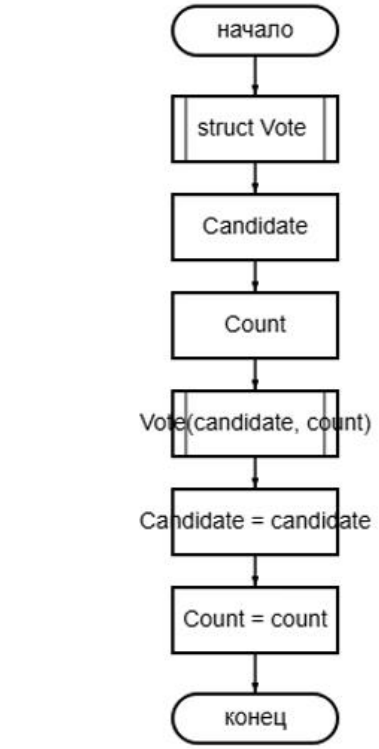

Уровень 1 Задание 1
начало
Participant
LastName
Club
Attempt1
Attempt2
TotalScore
Participant(lastName, club, attempt1, attempt2)
LastName = lastName
Club = club
Attempt1 = attempt1
Attempt2 = attempt2
TotalScore = attempt1 + attempt2
конец
Run
var participants = List<Participant>
Participant("Lewandowski", "Spartak", 6.5, 7.1)
Participant("Yamal", "Dynamo", 7.2, 6.9)
Participant("Raphinha", "Labor", 6.8, 7.3)
var sortedParticipants = participants.OrderByDescending(p => p.TotalScor
Результаты соревнований по прыжкам в длину:
Место\tФамилия\t\tОбщество\tПопытка 1\tПопытка 2\tСумма
i = 0; i < sortedParticipants.Count; i++

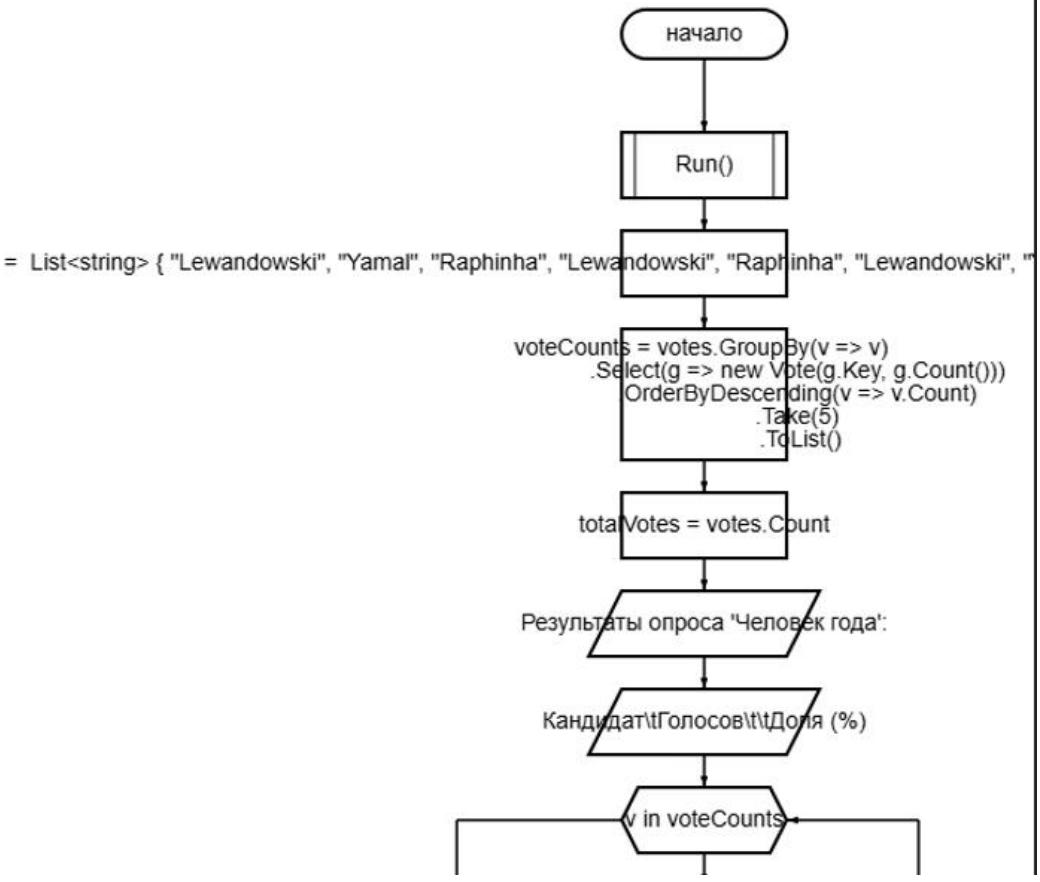

i = 0; i < sortedParticipants.Count; i++
var p = sortedParticipants[i]
{i + 1}\t{p.LastName}\t\t{p.Club}\t\t{p.Attempt1:F2}\t\t{p.Attempt2:F2}\t\t{p.Tot
конец

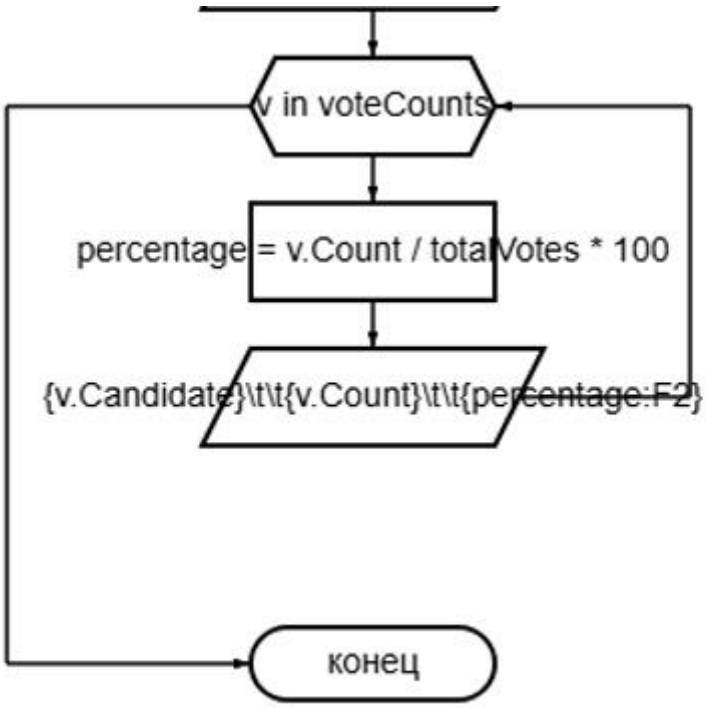

Уровень 1 Задание 2
начало
struct Runner
LastName
Group
Coach
Result
NormativeMet
Runner(lastName, group, coach, result, normative)
LastName = lastName
Group = group
Result = result
NormativeMet = result <= normative
конец
начало
Run()
normative = 90.0
var runners = List<Runner>
Runner("Lewandowski", "Group 1", "Smirnov", 85.4, normative)


Runner("Lewandowski", "Group 1", "Smirnov", 85.4, normative)
Runner("Yamal", "Group 2", "Petrov", 92.1, normative)
Runner("Raphinha", "Group 1", "Smirnov", 88.7, normative)
var sortedRunners = runners.OrderBy(r => r.Result).ToList()
normativeCount = sortedRunners.Count(r => r.NormativeMet)
Результаты кросса на 500 м для женщин:
Фамилия\t\tГруппа\t\tПреподаватель\tРезультат\tНорматив
r in sortedRunners
r.LastName}\t\t{r.Group}\t{r.Coach}\t\t{r.Result:F1}\t\t{(r.NormativeMet ? "Выполнен"
Всего участниц, выполнивших норматив: {normativeCount}
конец

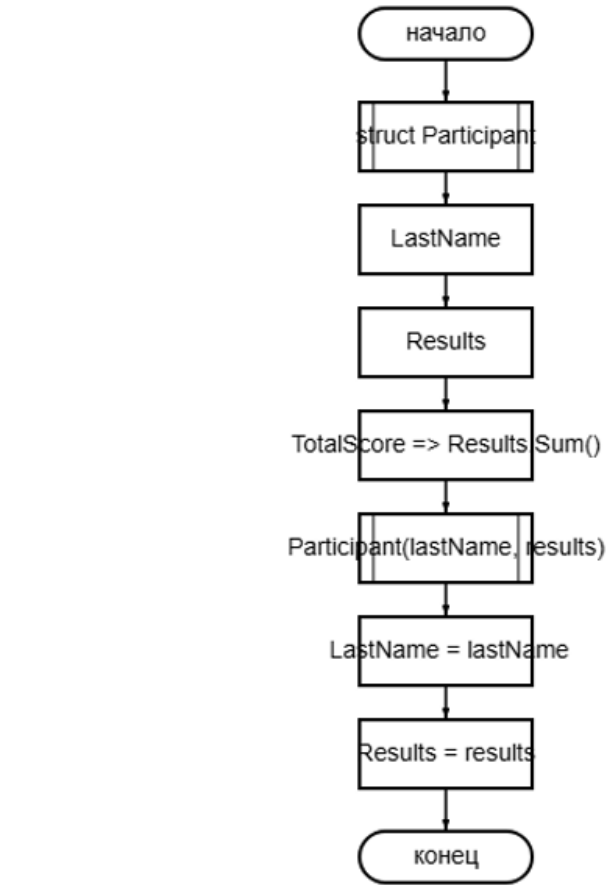
1 Уровень 3 Задание

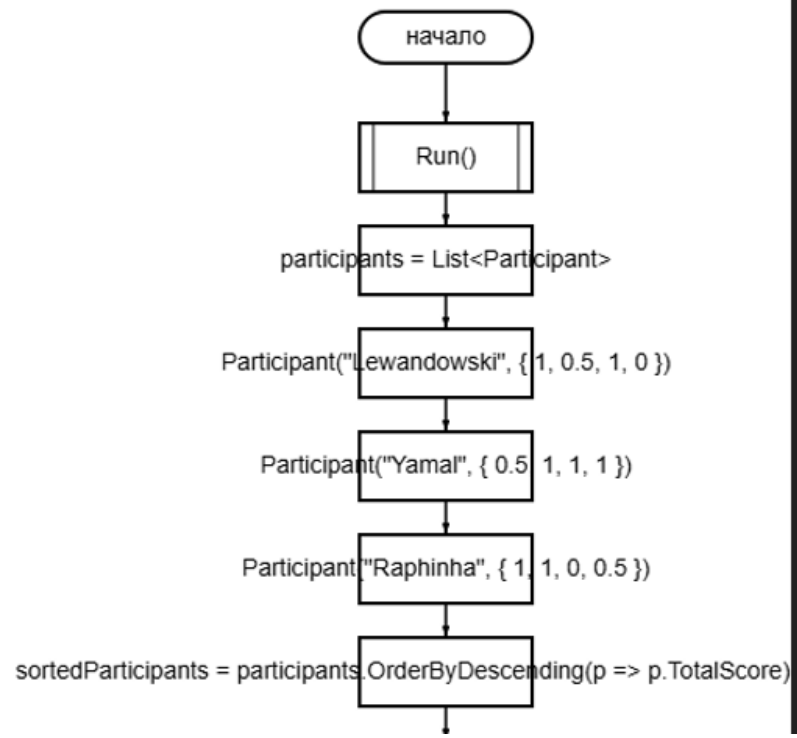


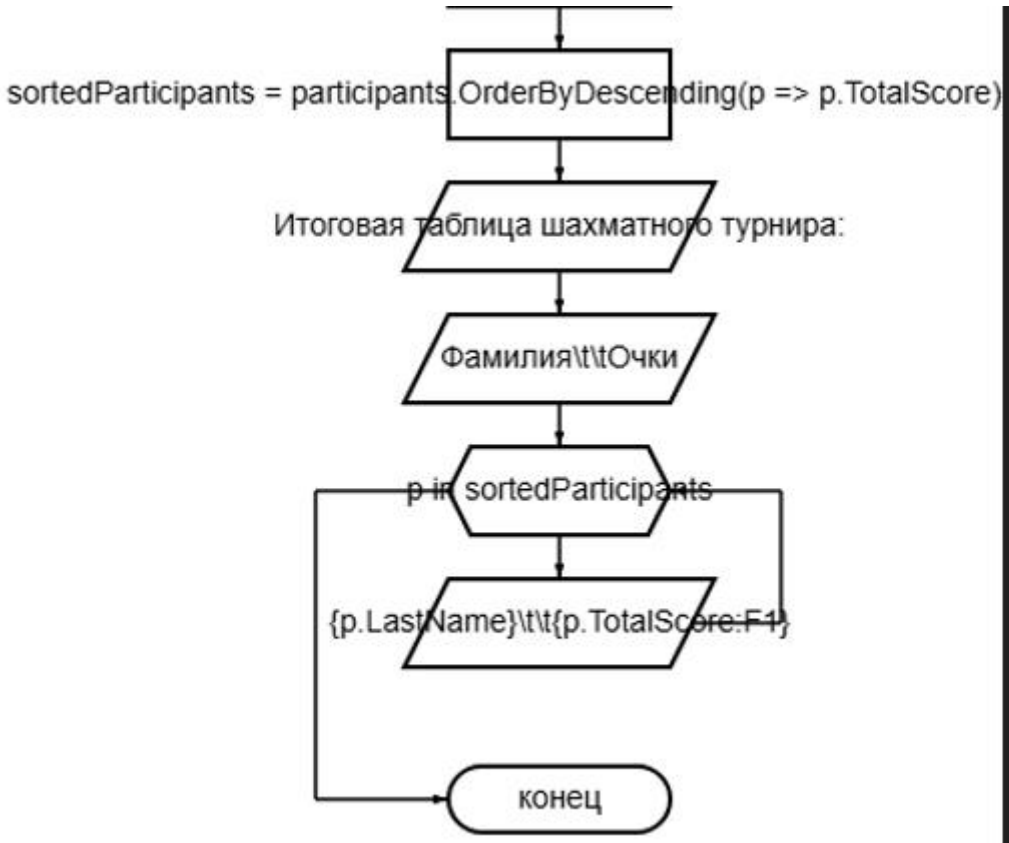



v in voteCounts
percentage = v.Count / totalVotes * 100
{v.Candidate}\t\t{v.Count}\t\t{percentage:F2}
конец

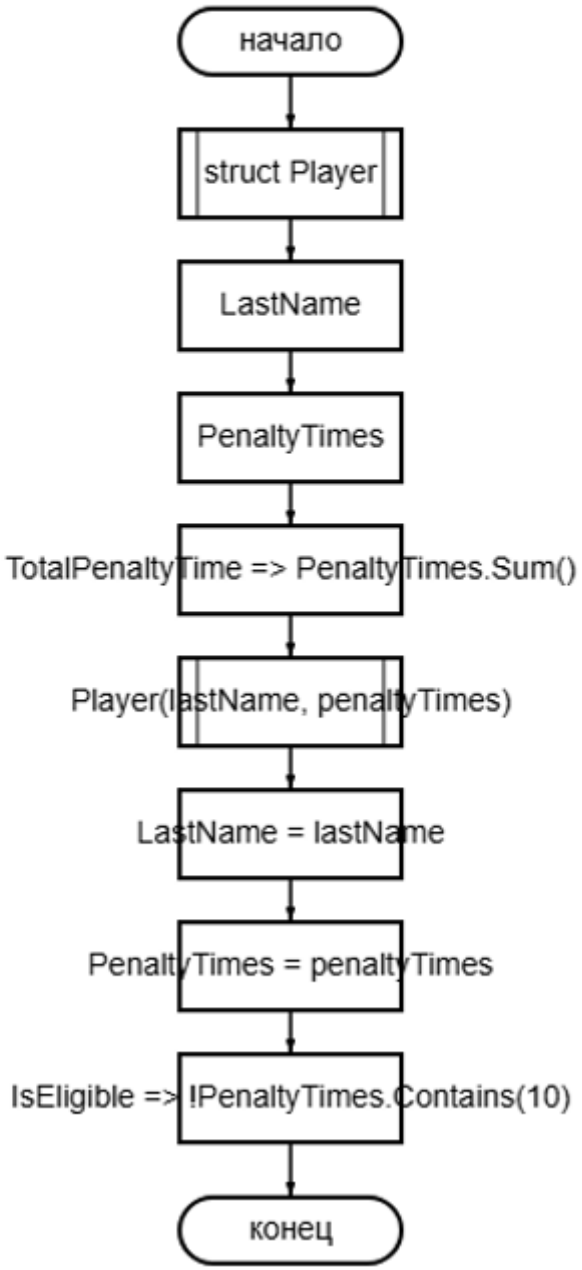
2 Уровень 7 Задание

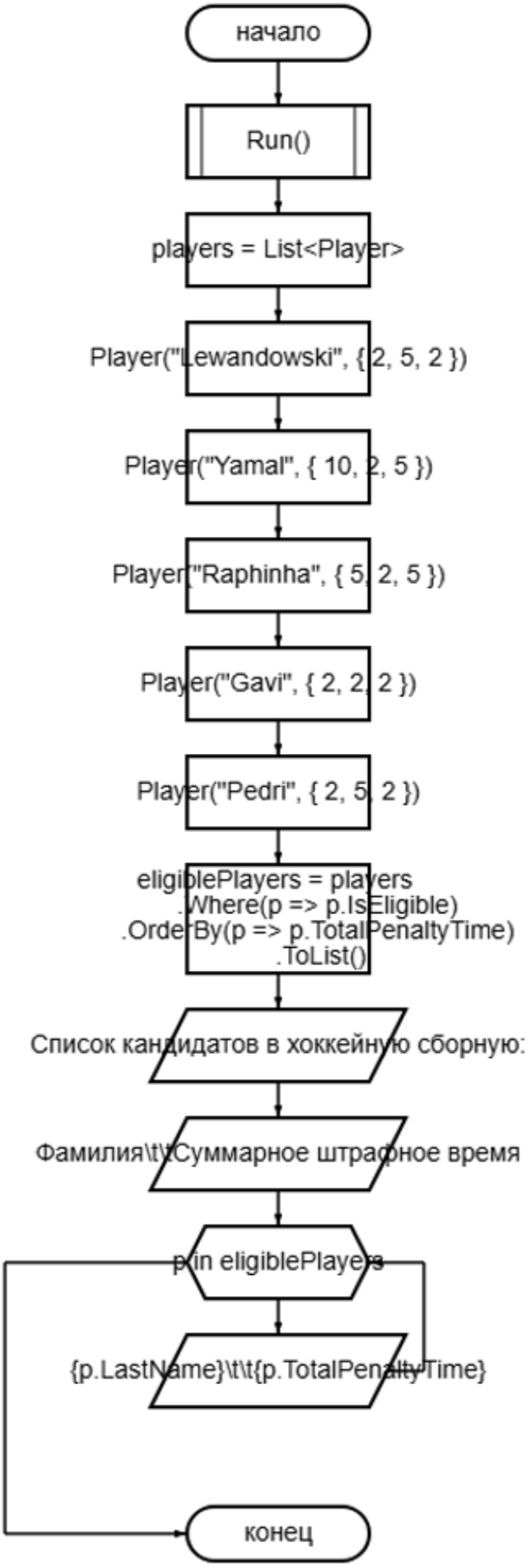




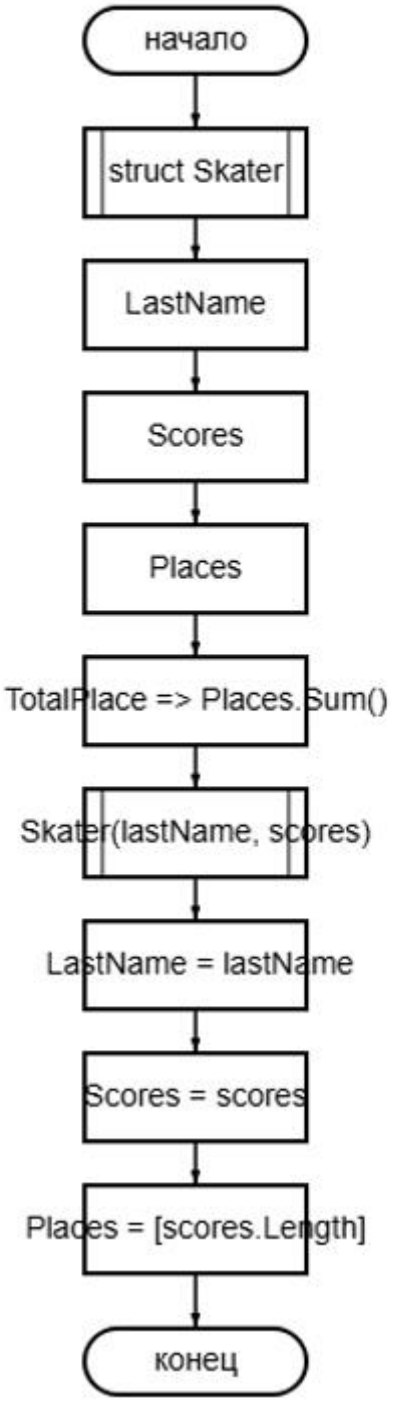

sortedParticipants = participants.OrderByDescending(p => p.TotalScore)
Итоговая таблица шахматного турнира:
Фамилия\t\tОчки
p in sortedParticipants
{p.LastName}\t\t{p.TotalScore:F1}
конец

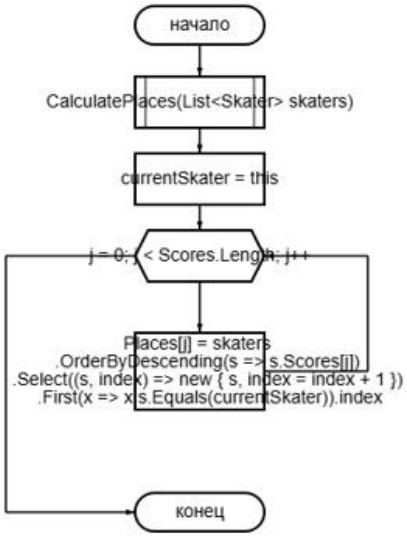
2 Уровень 8 Задание

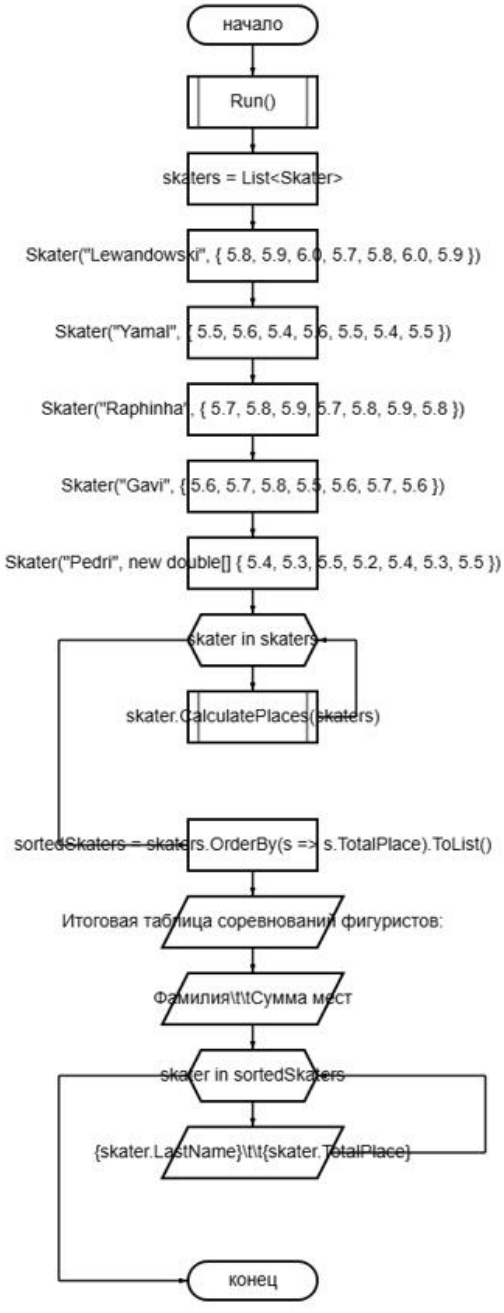



начало
Run()
players = List<Player>
Player("Lewandowski", { 2, 5, 2 })
Player("Yamal", { 10, 2, 5 })
Player("Raphinha", { 5, 2, 5 })
Player("Gavi", { 2, 2, 2 })
Player("Pedri", { 2, 5, 2 })
eligiblePlayers = players
.Where(p => p.IsEligible)
.OrderBy(p => p.TotalPenaltyTime)
.ToList()
Список кандидатов в хоккейную сборную:
Фамилия\t\tСуммарное штрафное время
p in eligiblePlayers
{p.LastName}\t\t{p.TotalPenaltyTime}
конец

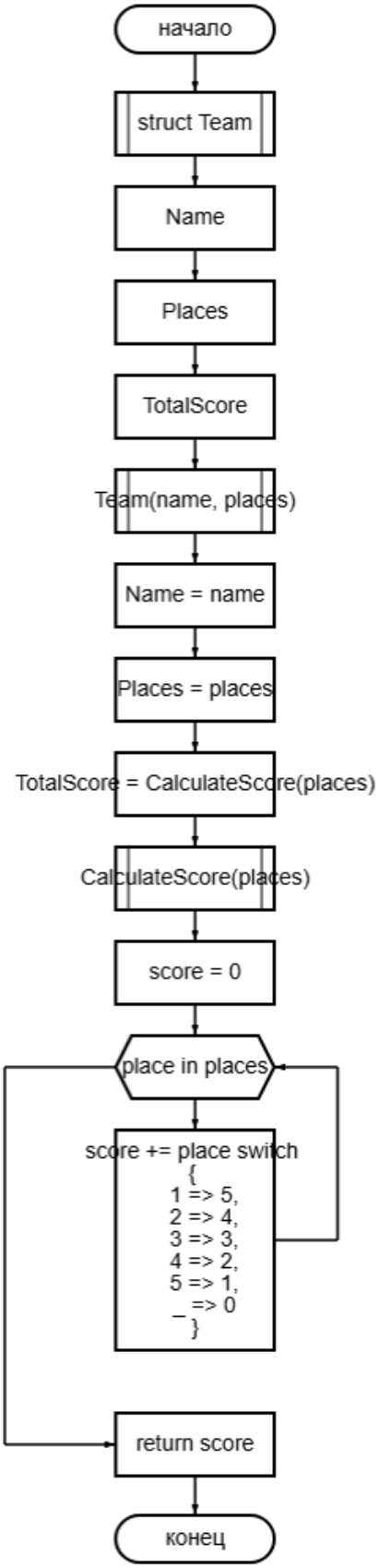
2 Уровень 9 Задание

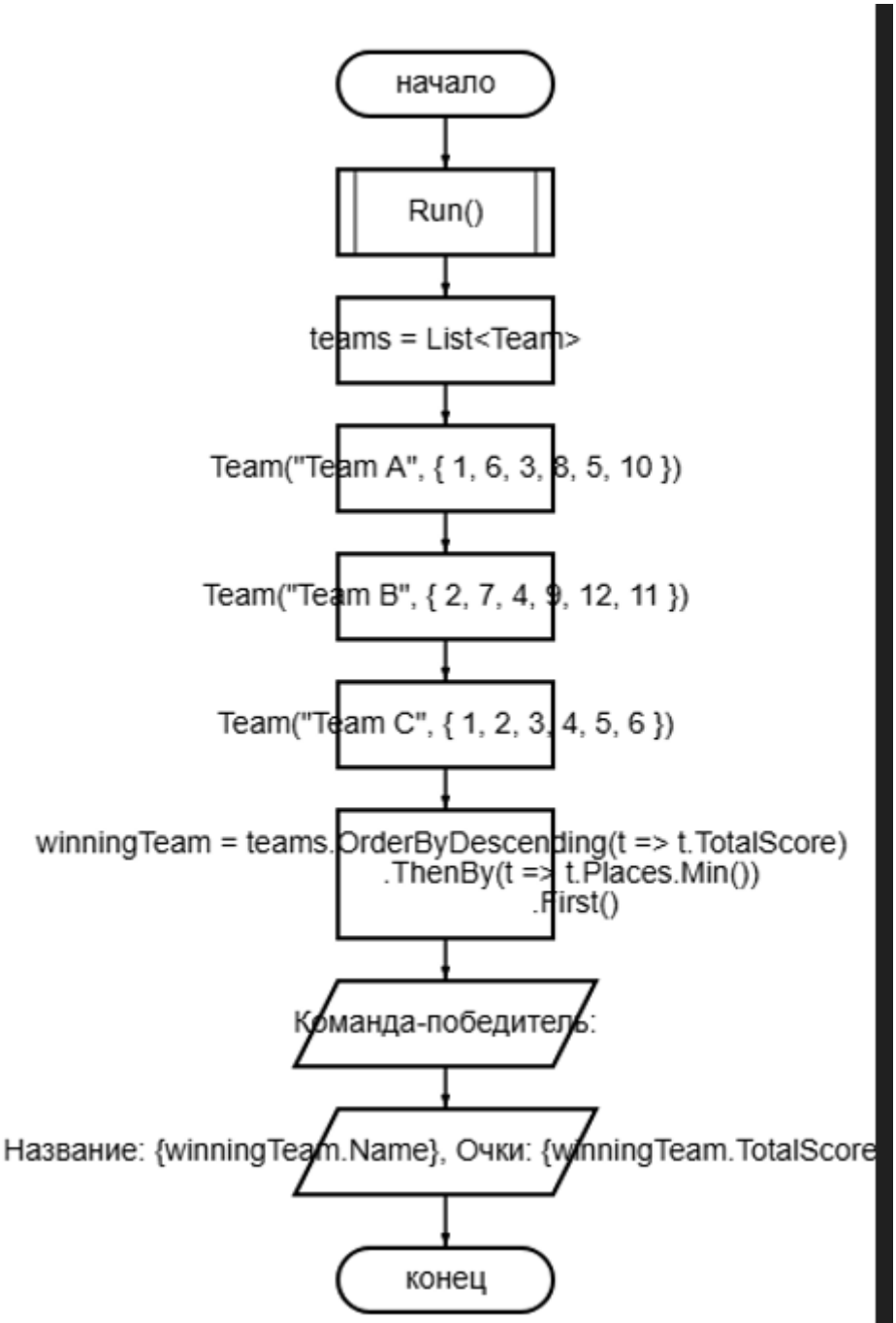



начало
CalculatePlaces(List<Skater> skaters)
currentSkater = this
j = 0; j < Scores.Length; j++
Places[j] = skaters
.OrderByDescending(s => s.Scores[j])
.Select((s, index) => new { s, index = index + 1 })
.First(x => x.s.Equals(currentSkater)).index
конец

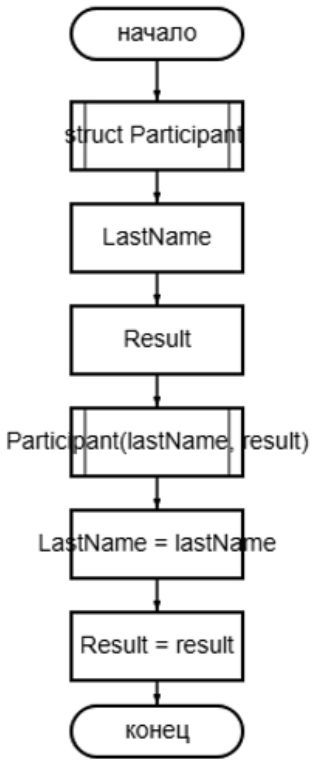

начало
Run()
skaters = List<Skater>
Skater("Lewandowski", { 5.8, 5.9, 6.0, 5.7, 5.8, 6.0, 5.9 })
Skater("Yamal", { 5.5, 5.6, 5.4, 5.6, 5.5, 5.4, 5.5 })
Skater("Raphinha", { 5.7, 5.8, 5.9, 5.7, 5.8, 5.9, 5.8 })
Skater("Gavi", { 5.6, 5.7, 5.8, 5.5, 5.6, 5.7, 5.6 })
Skater("Pedri", new double[] { 5.4, 5.3, 5.5, 5.2, 5.4, 5.3, 5.5 })
skater in skaters
skater.CalculatePlaces(skaters)
sortedSkaters = skaters.OrderBy(s => s.TotalPlace).ToList()
Итоговая таблица соревнований фигуристов:
Фамилия\t\tСумма мест
skater in sortedSkaters
{skater.LastName}\t\t{skater.TotalPlace}
конец

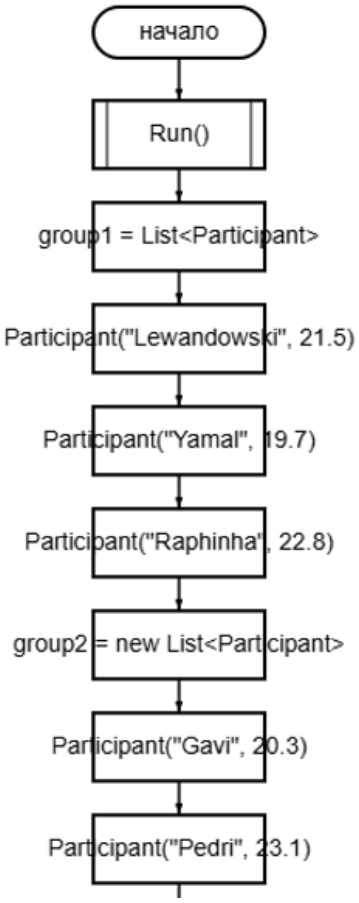
3 Уровень 3 Задание

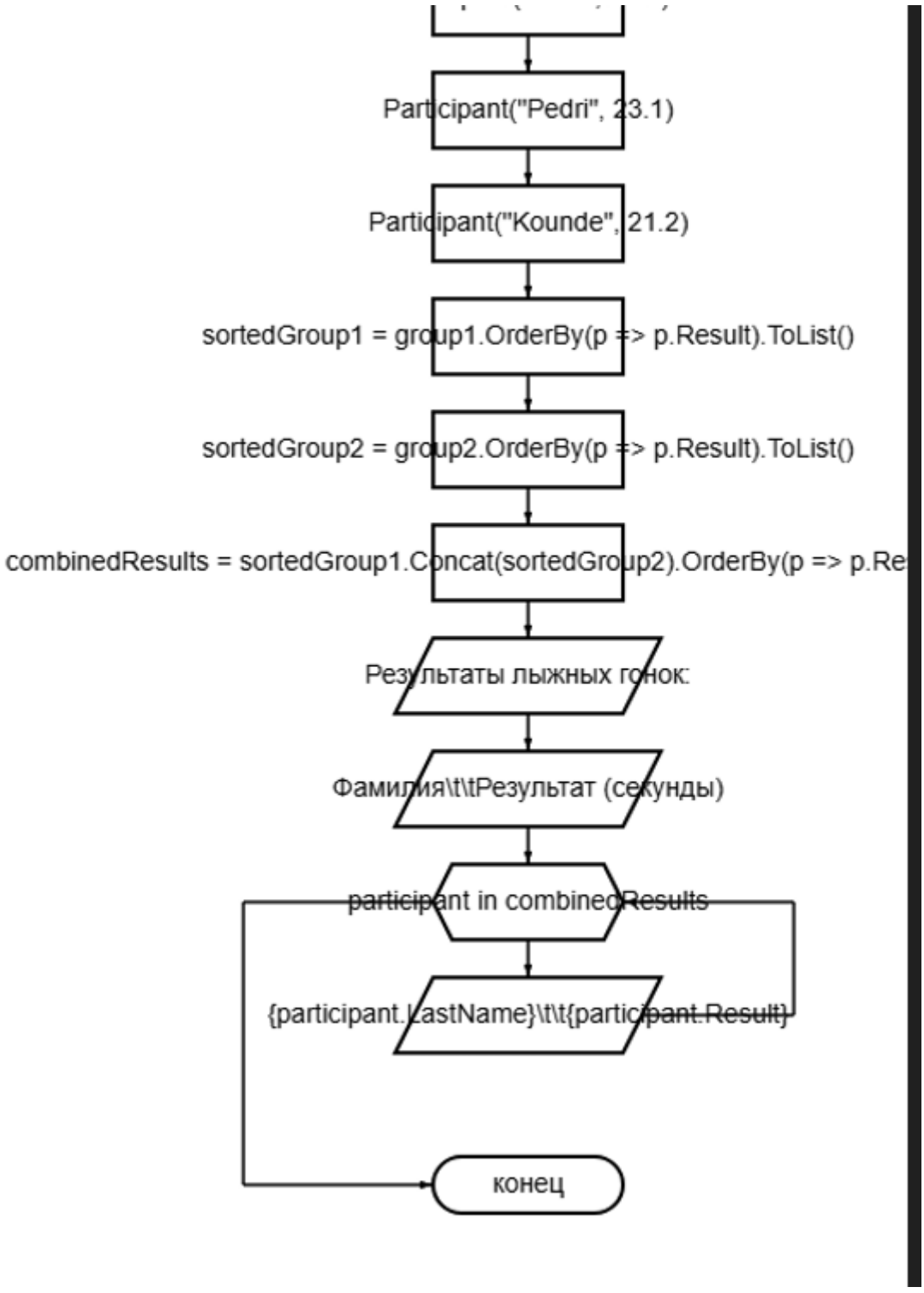



начало
Run()
teams = List<Team>
Team("Team A", { 1, 6, 3, 8, 5, 10 })
Team("Team B", { 2, 7, 4, 9, 12, 11 })
Team("Team C", { 1, 2, 3, 4, 5, 6 })
winningTeam = teams.OrderByDescending(t => t.TotalScore)
.ThenBy(t => t.Places.Min())
.First()
Команда-победитель:
Название: {winningTeam.Name}, Очки: {winningTeam.TotalScore
конец

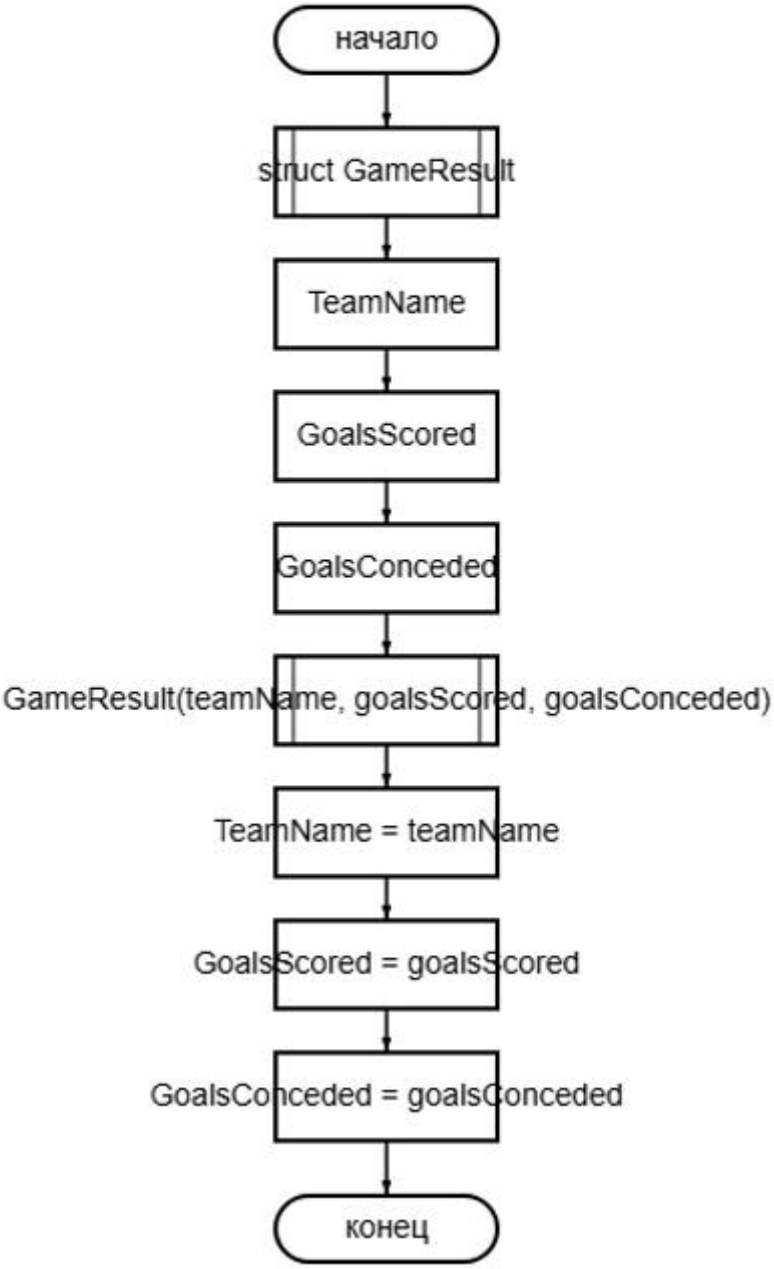
3 Уровень 4 Задание

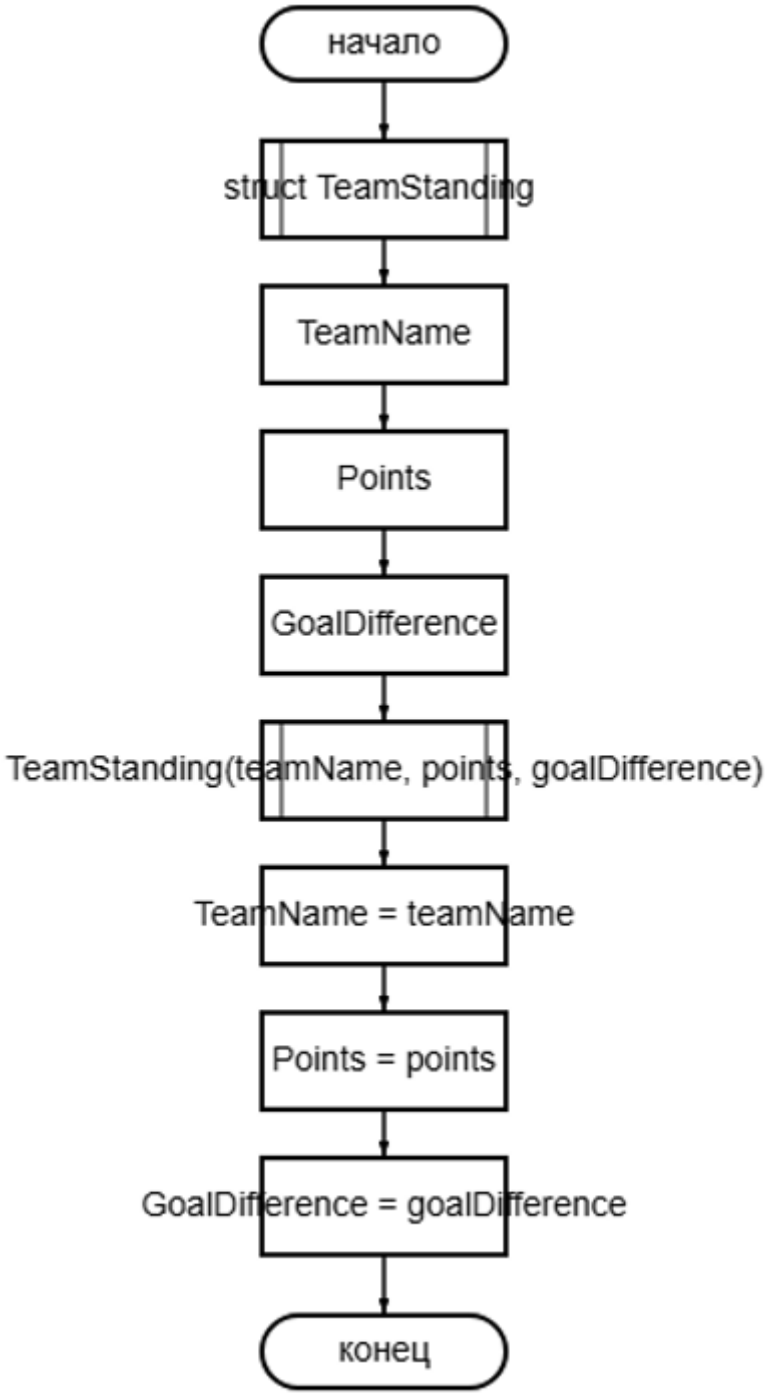


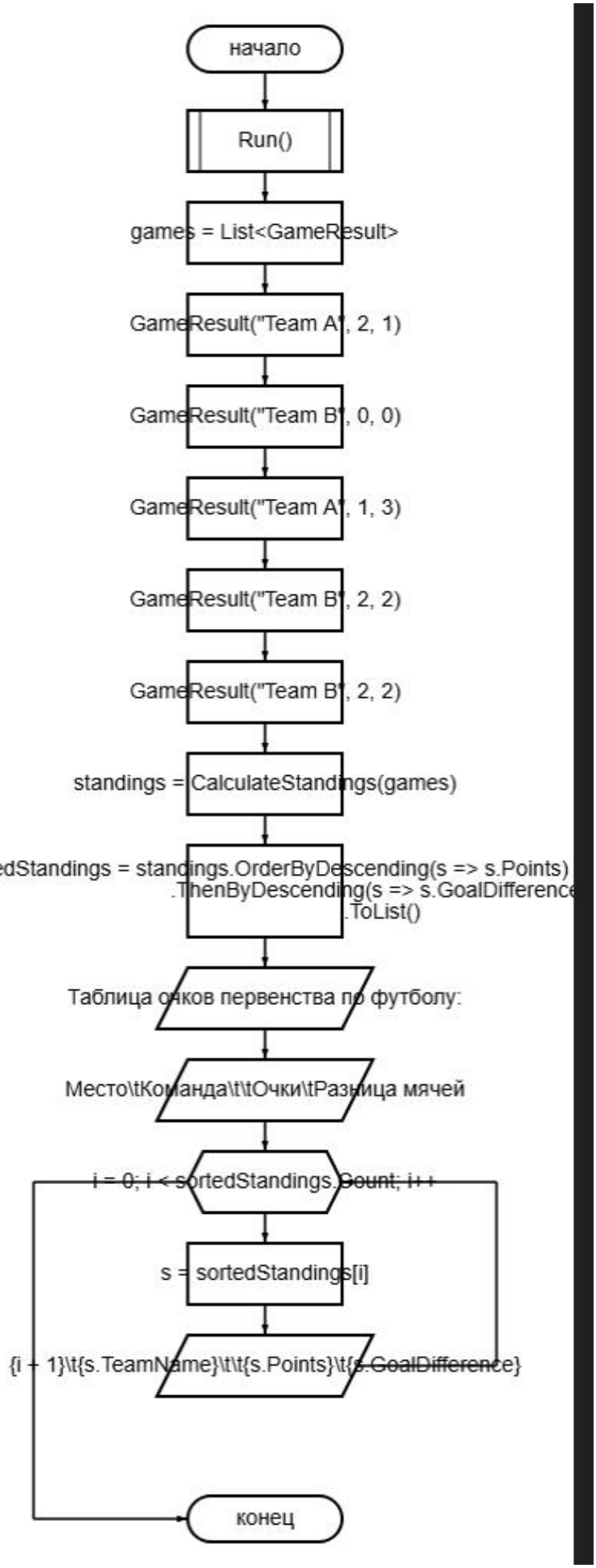


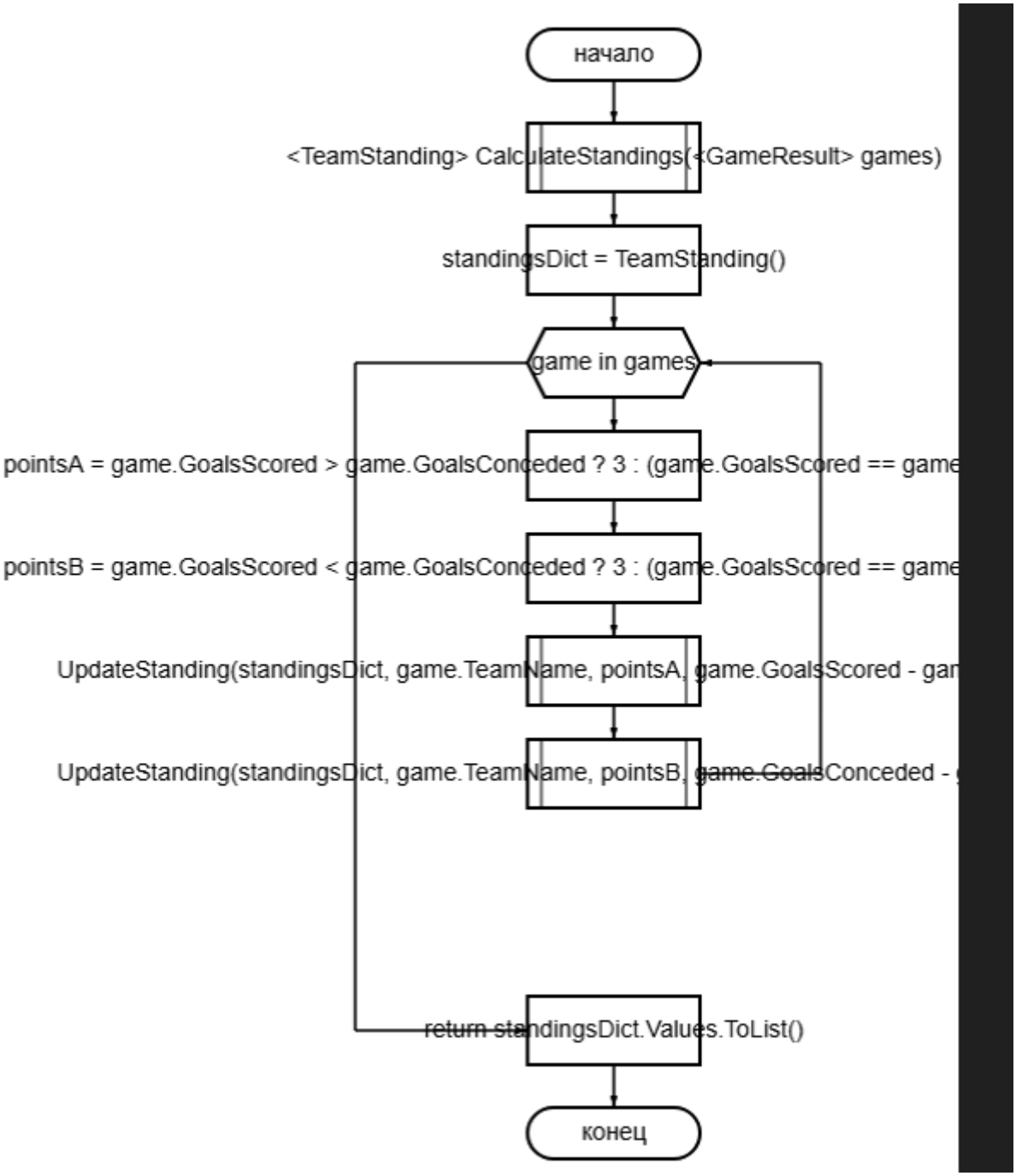

Participant("Pedri", 23.1)
Participant("Kounde", 21.2)
sortedGroup1 = group1.OrderBy(p => p.Result).ToList()
sortedGroup2 = group2.OrderBy(p => p.Result).ToList()
combinedResults = sortedGroup1.Concat(sortedGroup2).OrderBy(p => p.Re
Результаты лыжных гонок:
Фамилия\t\tРезультат (секунды)
participant in combinedResults
{participant.LastName}\t\t{participant.Result}
конец

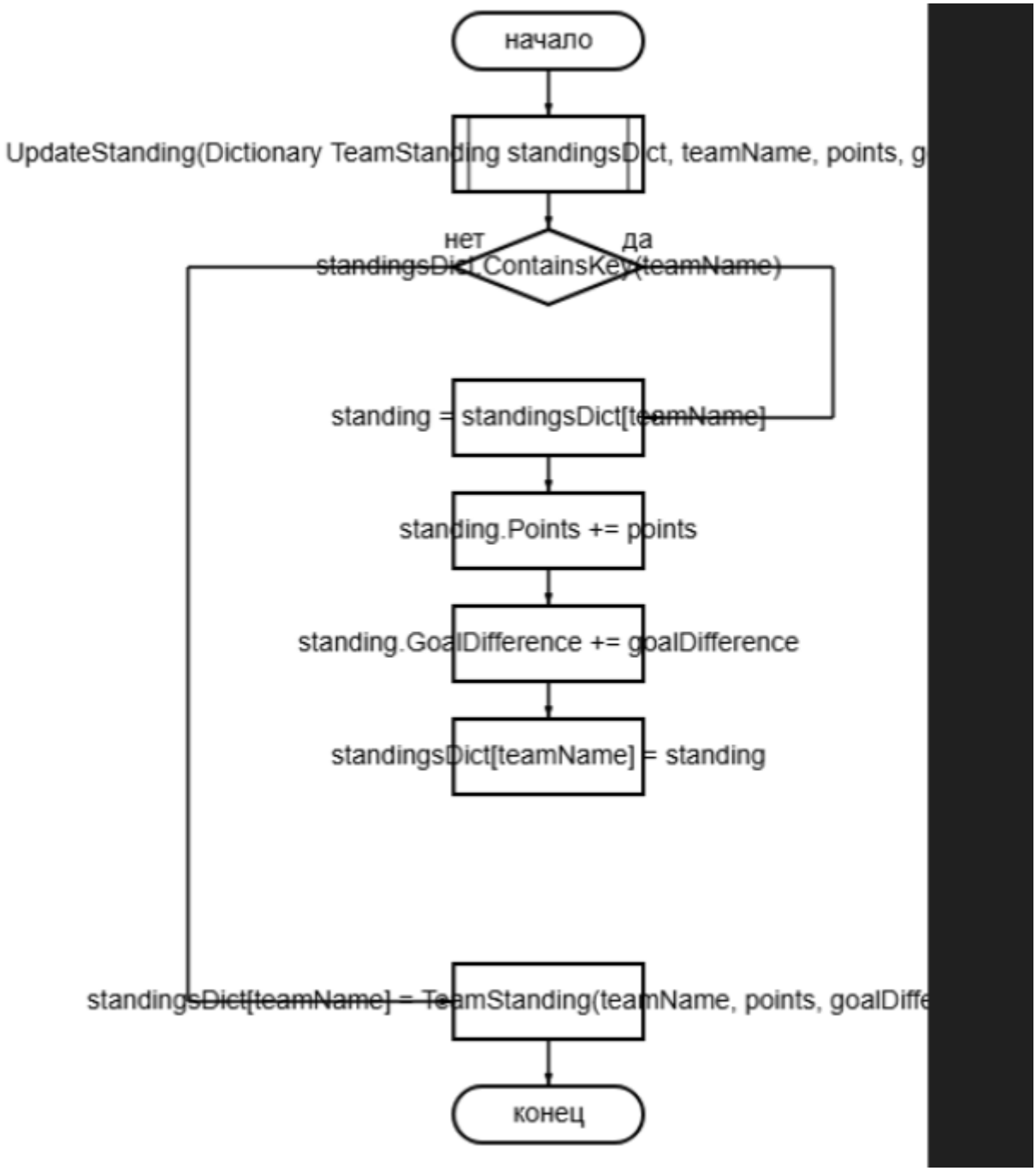
3 Уровень 5 Задание




начало
struct TeamStanding
TeamName
Points
GoalDifference
TeamStanding(teamName, points, goalDifference)
TeamName = teamName
Points = points
GoalDifference = goalDifference
конец


начало
Run()
games = List<GameResult>
GameResult("Team A", 2, 1)
GameResult("Team B", 0, 0)
GameResult("Team A", 1, 3)
GameResult("Team B", 2, 2)
GameResult("Team B", 2, 2)
standings = CalculateStandings(games)
sortedStandings = standings.OrderByDescending(s => s.Points)
.ThenByDescending(s => s.GoalDifference
.ToList()
Таблица очков первенства по футболу:
Место\tКоманда\t\tОчки\tРазница мячей
i = 0; i < sortedStandings.Count; i++
s = sortedStandings[i]
{i + 1}\t{s.TeamName}\t\t{s.Points}\t{s.GoalDifference}
конец


начало
<TeamStanding> CalculateStandings(<GameResult> games)
standingsDict = TeamStanding()
game in games
pointsA = game.GoalsScored > game.GoalsConceded ? 3 : (game.GoalsScored == game
pointsB = game.GoalsScored < game.GoalsConceded ? 3 : (game.GoalsScored == game
UpdateStanding(standingsDict, game.TeamName, pointsA, game.GoalsScored - gam
UpdateStanding(standingsDict, game.TeamName, pointsB, game.GoalsConceded -
return standingsDict.Values.ToList()
конец



Вывод: В ходе выполнения лабораторной работы были приобретены навыки использования структур и методов для решения задач. Были разработаны программы для обработки результатов соревнований, упорядочивания данных и выполнения расчетов по заданным правилам. Работа показала, как эффективно комбинировать структуры и методы для хранения и обработки данных, улучшая читаемость и структурированность кода. Полученные результаты продемонстрировали возможности C# для решения реальных задач с использованием программных подходов.