

**«Разработка ОО программы моделирования заданной
предметной области»**

Содержание

1. Задание	3
2. Описание предметной области	3
3. Диаграмма классов	5
4. Описание полей и методов классов.....	6
5. Описание формул	13
6. Описание пользовательского интерфейса	18
7. Тестирование программы	19
8. Выводы	24
Приложение А. Код иерархии классов.	25
Приложение Б. Код основной программы.....	36

1. Задание

Разработать ОО программу моделирования заданной предметной области на языке программирования C#. Реализовать для иерархии механизм интерфейсов, один из классов должен реализовывать как минимум два интерфейса. Использовать для проверки всех методов данного класса многоадресный делегат. Реализовать обработку ошибок (переопределив с помощью наследования одно из событий: StackOverflowException, ArrayTypeMismatchException, DivideByZeroException, IndexOutOfRangeException, InvalidCastException, OutOfMemoryException, OverflowException). Показать пример использования полиморфизма.

2. Описание предметной области

Предметная область: Благотворительный мини-футбольный матч.

В рамках благотворительной инициативы две футбольные команды собираются на поле в мини-футбольном матче. Это событие не только объединяет любителей спорта, но и направлено на сбор средств для благотворительных программ, поддерживающих нуждающихся в помощи.

Предматчевая студия: Перед началом матча болельщики собираются в предматчевой студии, где им предоставляется возможность погрузиться в атмосферу ожидания и подготовки к игре. Они наблюдают за разминкой команд, изучают составы команд. Интервью с игроками и тренерами позволяют болельщикам узнать больше о матче.

Репортаж с матча: Когда первый свисток судьи открывает игру, репортаж с матча погружает зрителей в атмосферу борьбы на поле. Фанаты делают прогноз на матч, сделав пожертвование на благотворительность. После игры болельщики изучают результаты матча.

Благотворительный мини-футбольный матч объединяет спорт, эмоции и добroе дело, создавая незабываемый опыт как для участников, так и для болельщиков.

Расстановка команд имеет следующий вид, представленный на рис. 1.



Рис. 1. Расстановка команд

Мини-футбольная схема 2-0-2(1) с вратарём (GK, 1), двумя защитниками (DEF, 2, 3), оттянутым форвардом (CF, 4) и нападающим (ST, 5) представляет собой тактическую расстановку игроков на поле. Вратарь (GK) занимает ворота и защищает их от голов соперника. Два защитника (DEF) располагаются по краям поля и обеспечивают поддержку вратарю, а также контролируют зоны вблизи своих ворот. Оттянутый форвард (CF) находится в центре поля и помогает нападающему (ST) атаковать и создавать моменты для взятия ворот. Нападающий (ST) располагается на острие атаки и стремится забить гол, используя возможности, предоставленные ему оттянутым форвардом (CF) и защитниками (DEF).

3. Диаграмма классов

Диаграмма классов представлена на рис. 2 и рис. 3.

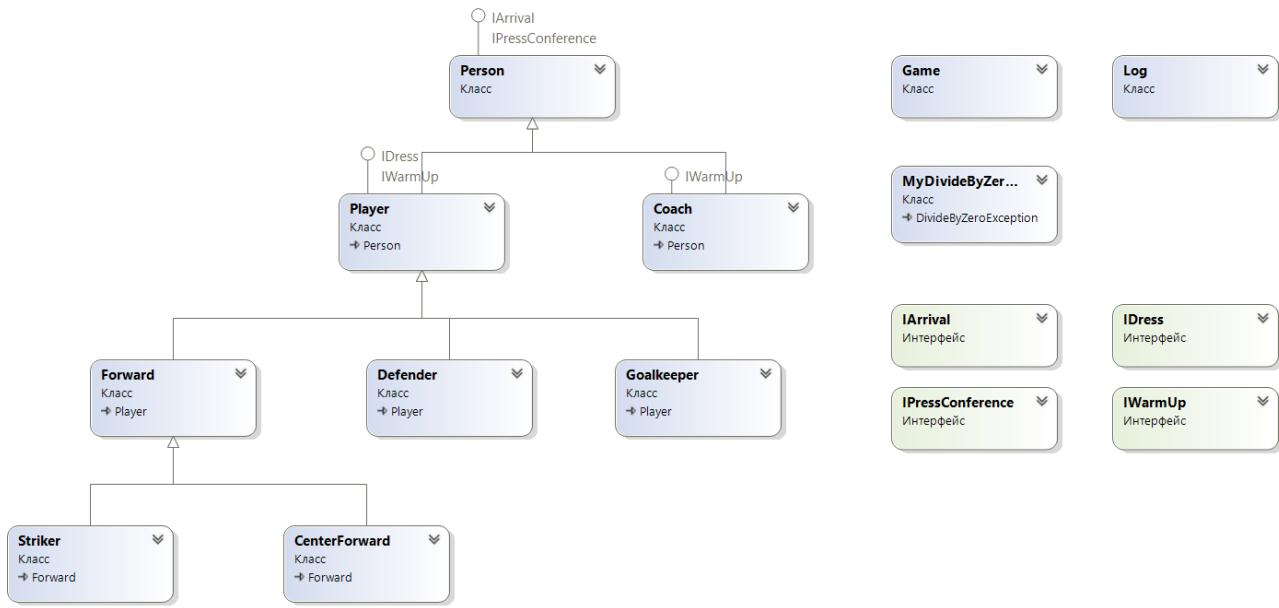


Рис. 2. Диаграмма классов

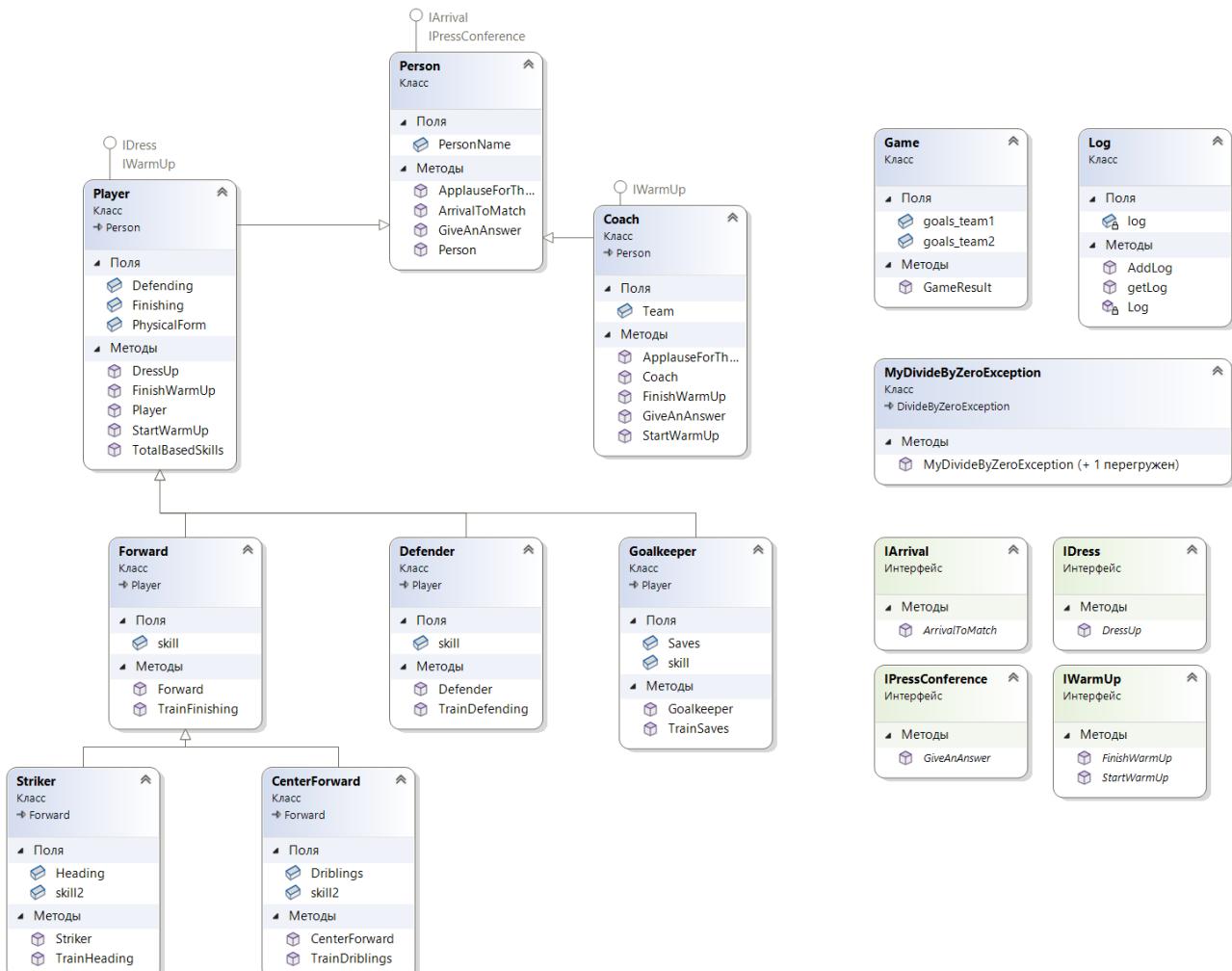


Рис. 3. Диаграмма классов в развернутом виде

4. Описание полей и методов классов

Описание полей и методов класса Person приведено в табл. 1.

Таблица 1. Поля и методы класса Person

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
PersonName	string	Имя участника матча	public	-	-
Методы					
Person	-	Конструктор класса	public	string name	-
ApplauseForTheFans	void	Аплодисменты фанатам. Результат фиксируется в логе программы	public	-	-
ArrivalToMatch	void	Прибытие матч. Результат фиксируется в логе программы	public	-	-
GiveAnAnswer	void	Ответы на вопросы журналиста после матча. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса Player приведено в табл. 2. Данный класс является дочерним к классу Person, поэтому наследует все его поля и методы.

Таблица 2. Поля и методы класса Player

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
PhysicalForm	int	Показатель физической формы игрока	public	-	-
Finishing	int	Показатель завершающего удара игрока	public	-	-
Defending	int	Показатель оборонительных навыков игрока	public	-	-

Таблица 2. Продолжение

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Методы					
Player	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending	-
DressUp	void	Надеть спортивную экипировку. Результат фиксируется в логе программы	public	-	-
TotalBasedSkills	int	Возвращает сумму показателей базовых навыков игрока (PhysicalForm, Finishing, Defending)	public	-	int sum
StartWarmUp	void	Начало разминки игрока перед матчем. Результат фиксируется в логе программы	public	-	-
FinishWarmUp	void	Конец разминки игрока перед матчем. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса Coach приведено в табл. 3. Данный класс является дочерним к классу Person, поэтому наследует все его поля и методы.

Таблица 3. Поля и методы класса Coach

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
Team	string	Название подопечной команды тренера	public	-	-
Методы					
Coach	-	Конструктор класса	public	string name, string team,	-
ApplauseForTheFans	void	Помахать фанатам. Результат фиксируется в логе программы	public override	-	-

Таблица 3. Продолжение

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
StartWarmUp	void	Начать разминку команды перед матчем. Результат фиксируется в логе программы	public	-	-
FinishWarmUp	void	Закончить разминку команды перед матчем. Результат фиксируется в логе программы	public	-	-
GiveAnAnswer	void	Ответы на вопросы журналиста до матча. Результат фиксируется в логе программы	public	string replica	-

Описание полей и методов класса Forward приведено в табл. 4. Данный класс является дочерним к классу Player, поэтому наследует все его поля и методы.

Таблица 4. Поля и методы класса Forward

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
skill	string	Название ключевого навыка, согласно амплуа	public	-	-
Методы					
Forward	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending	-
TrainFinishing	void	Упражнение на разминке перед матчем, которое может увеличить или уменьшить ключевой навык на время игры. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса Defender приведено в табл. 5. Данный класс является дочерним к классу Player, поэтому наследует все его поля и методы.

Таблица 5. Поля и методы класса Defender

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
skill	string	Название ключевого навыка, согласно амплуа	public	-	-
Методы					
Defender	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending	-
TrainDefending	void	Упражнение на разминке перед матчем, которое может увеличить или уменьшить ключевой навык на время игры. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса Goalkeeper приведено в табл. 6. Данный класс является дочерним к классу Player, поэтому наследует все его поля и методы.

Таблица 6. Поля и методы класса Goalkeeper

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
skill	string	Название ключевого навыка, согласно амплуа	public	-	-
Saves	int	Показатель навыка игры в воротах	public		
Методы					
Goalkeeper	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending, int Saves	-

Таблица 6. Продолжение

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
TrainSaves	void	Упражнение на разминке перед матчем, которое может увеличить или уменьшить ключевой навык на время игры. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса CenterForward приведено в табл. 7. Данный класс является дочерним к классу Forward, поэтому наследует все его поля и методы.

Таблица 7. Поля и методы класса CenterForward

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
skill2	string	Название второго ключевого навыка, согласно амплуа	public	-	-
Dribblings	int	Показатель навыка дриблинга	public		
Методы					
CenterForward	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending, int Dribblings	-
TrainDribblings	void	Упражнение на разминке перед матчем, которое может увеличить или уменьшить второй ключевой навык на время игры. Результат фиксируется в логе программы	public	-	-

Описание полей и методов класса Striker приведено в табл. 8. Данный класс является дочерним к классу Forward, поэтому наследует все его поля и методы.

Таблица 8. Поля и методы класса Striker

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
skill2	string	Название второго ключевого навыка, согласно амплуа	public	-	-
Heading	int	Показатель навыка удара головой	public	-	-
Методы					
Striker	-	Конструктор класса	public	string name, int PhysicalForm, int Finishing, int Defending, int Heading	-
TrainHeading	void	Упражнение на разминке перед матчем, которое может увеличить или уменьшить второй ключевой навык на время игры. Результат фиксируется в логе программы	public	-	-

Класс Log используется для логирования всех происходящих действий в программе Описание полей и методов данного класса приведено в табл. 9.

Таблица 9. Поля и методы класса Log

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
log	Log	Экземпляр класса Log	private	-	-
Методы					
Log	-	Конструктор класса	private	-	-
getLog	Log	Создает только один экземпляр класса.	public	-	Экземпляр класса Log -
AddLog	void	Записывает строку в лог файл	public	string msg – сообщение для записи в лог файл	-

Класс Game используется для моделирования футбольного матча. Описание полей и методов данного класса приведено в табл. 10.

Таблица 10. Поля и методы класса Game

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Поля					
goals_team1	int	Количество забитых голов первой командой	public	-	-
goals_team2	int	Количество забитых голов второй командой	public	-	-
Методы					
GameResult	string	Моделирует футбольный матч	public	int skill1, int skill2, int saves1, int saves2, int finishing11, int finishing12, int finishing21, int finishing22, int def11, int def12, int def21, int def22, int head1, int head2, int dribbling1, int dribbling2	string Авторы забитых мячей в каждой команде-

Класс MyDivideByZeroException является пользовательским исключением, которое наследуется от стандартного исключения DivideByZeroException. Этот класс используется для создания переопределенного исключения при делении на ноль. Описание полей и методов данного класса приведено в табл. 11.

Таблица 11. Поля и методы класса MyDivideByZeroException

Имя	Тип	Описание	Область видимости	Входные параметры	Выходные параметры
Методы					
MyDivideByZeroException	-	Конструктор без параметров вызывает конструктор базового класса (DivideByZeroException) с заданным сообщением об ошибке. Это сообщение будет выводиться при возникновении исключения.	public	-	-
MyDivideByZeroException		Конструктор с параметром для сообщения об ошибке позволяет задать пользовательское сообщение при создании объекта исключения.	public	string message-	-

5. Описание формул

Описание формулы, используемой в методе **TrainFinishing**:

1. Начало тренировки: Метод начинает с логирования текущего значения навыка "Finishing" (Finishing) игрока перед тренировкой.
2. Случайные параметры: Создается объект Random для генерации случайных чисел. Генерируется случайное вещественное число randomNumber в диапазоне от 0.9 до 1.1 (включительно).
3. Тренировочные попытки: Задается количество попыток тренировки (attempts), например, 50. Заводится переменная goals для отслеживания количества успешных попыток.
4. Цикл тренировки: Происходит цикл с attempts итерациями, в каждой итерации генерируется новое случайное число randomNumber. Если сумма randomNumber + PhysicalForm / 100.0 больше или равна 1.0, то попытка считается успешной (гол забит), и увеличивается счетчик goals.

5. Результат тренировки: Вычисляется процент успешных попыток тренировки (result = 100.0 * goals / attempts). На основе этого результата навык "Finishing" (Finishing) увеличивается или уменьшается. Например, если результат тренировки (result) равен или больше 80%, то к навыку "Finishing" прибавляется 5 единиц. Если результат между 60% и 80%, то добавляются 2 единицы. Если результат между 50% и

60%, то добавляется 1 единица. В остальных случаях (меньше 50%) к навыку "Finishing" вычитаются 2 единицы.

6. Завершение тренировки: По окончании цикла тренировки логируется новое значение навыка "Finishing" (Finishing) после тренировки.

Эта формула моделирует тренировку навыка "Finishing" игрока на основе случайных параметров и результатов попыток, что отражает реалистичный процесс улучшения игровых навыков в футболе.

Описание формулы, применяемой в методе **TrainDefending**:

1. Начало тренировки: Метод начинает с логирования текущего значения навыка "Defending" (Defending) у игрока перед тренировкой.

2. Случайные параметры: Создается объект Random для генерации случайных чисел. Генерируется случайное вещественное число randomNumber в диапазоне от 0.9 до 1.1 (включительно).

3. Тренировочные попытки: Задается количество попыток тренировки (attempts), например, 50. Заводится переменная goals для отслеживания количества успешных попыток.

4. Цикл тренировки: Происходит цикл с attempts итерациями, в каждой итерации генерируется новое случайное число randomNumber. Если сумма randomNumber + PhysicalForm / 100.0 больше или равна 1.0, то попытка считается успешной (защита успешно справлена), и увеличивается счетчик goals.

5. Результат тренировки: Вычисляется процент успешных попыток тренировки (result = 100.0 * goals / attempts). На основе этого результата навык "Defending" (Defending) увеличивается или уменьшается. Например, если результат тренировки (result) равен или больше 75%, то к навыку "Defending" прибавляются 5 единиц. Если результат между 60% и 75%, то добавляются 2 единицы. Если результат между 40% и 60%, то добавляется 1 единица. В остальных случаях (меньше 40%) к навыку "Defending" вычитаются 2 единицы.

6. Завершение тренировки: По окончании цикла тренировки логируется новое значение навыка "Defending" (Defending) после тренировки.

Эта формула моделирует тренировку навыка "Defending" игрока на основе случайных параметров и результатов попыток, что отражает реалистичный процесс улучшения игровых навыков в области защиты в футболе.

Описание формулы, использованной в методе **TrainSaves**:

1. Начало тренировки: Метод начинает с логирования текущего значения навыка "Saves" (Saves) у вратаря перед тренировкой.

2. Случайные параметры: Создается объект Random для генерации случайных чисел. Генерируется случайное вещественное число randomNumber в диапазоне от 0.9 до 1.1 (включительно).

3. Тренировочные попытки: Задается количество попыток тренировки (attempts), например, 50. Заводится переменная goals для отслеживания количества успешных попыток.

4. Цикл тренировки: Происходит цикл с attempts итерациями, в каждой итерации генерируется новое случайное число randomNumber. Если сумма $randomNumber + PhysicalForm / 100.0$ больше или равна 1.0, то попытка считается успешной (вратарь успешно спас мяч), и увеличивается счетчик goals.

5. Результат тренировки: Вычисляется процент успешных попыток тренировки ($result = 100.0 * goals / attempts$). На основе этого результата навык "Saves" (Saves) увеличивается или уменьшается. Например, если результат тренировки (result) равен или больше 90%, то к навыку "Saves" прибавляются 5 единиц. Если результат между 60% и 90%, то добавляются 2 единицы. Если результат между 40% и 60%, то добавляется 1 единица. В остальных случаях (меньше 40%) к навыку "Saves" вычитаются 2 единицы.

6. Завершение тренировки: По окончании цикла тренировки логируется новое значение навыка "Saves" (Saves) после тренировки.

Эта формула моделирует тренировку навыка "Saves" у вратаря на основе случайных параметров и результатов попыток спасения мяча, что отражает реалистичный процесс улучшения навыков вратарской игры в футболе.

Описание формулы, использованной в методе **TrainDriblings**.

1. Начало тренировки: Метод начинает с логирования текущего значения навыка "Driblings" (Driblings) у игрока перед тренировкой.

2. Случайные параметры: Создается объект Random для генерации случайных чисел. Генерируется случайное вещественное число randomNumber в диапазоне от 0.9 до 1.1 (включительно).

3. Тренировочные попытки: Задается количество попыток тренировки (attempts), например, 50. Заводится переменная goals для отслеживания количества успешных попыток.

4. Цикл тренировки: Происходит цикл с attempts итерациями, в каждой итерации генерируется новое случайное число randomNumber. Если $randomNumber / 100.0$ больше или равно 1.0, то попытка дриблинга считается успешной, и увеличивается счетчик goals.

5. Результат тренировки: Вычисляется процент успешных попыток тренировки ($result = 100.0 * goals / attempts$). На основе этого результата навык "Dribblings" (Driblings) увеличивается или уменьшается. Например, если результат тренировки (result) равен или больше 75%, то к навыку "Dribblings" прибавляются 5 единиц. Если результат между 55% и 75%, то добавляются 2 единицы. Если результат между 45% и 55%, то добавляется 1 единица. В остальных случаях (меньше 45%) к навыку "Dribblings" вычитаются 2 единицы.

6. Завершение тренировки: По окончании цикла тренировки логируется новое значение навыка "Dribblings" (Driblings) после тренировки.

Эта формула моделирует процесс улучшения навыка дриблинга у футболиста на основе случайных параметров и результатов попыток дриблинга, что отражает реалистичный подход к тренировке виртуального футболиста.

Описание формулы, использованной в методе **TrainHeading**.

1. Начало тренировки: Метод начинает с логирования текущего значения навыка "Heading" (Heading) у игрока перед тренировкой.

2. Случайные параметры: Создается объект Random для генерации случайных чисел. Генерируется случайное вещественное число randomNumber в диапазоне от 0.9 до 1.1 (включительно).

3. Тренировочные попытки: Задается количество попыток тренировки (attempts), например, 50. Заводится переменная goals для отслеживания количества успешных попыток.

4. Цикл тренировки: Происходит цикл с attempts итерациями, в каждой итерации генерируется новое случайное число randomNumber. Если $randomNumber / 100.0$ больше или равно 1.0, то попытка удара мяча головой считается успешной, и увеличивается счетчик goals.

5. Результат тренировки: Вычисляется процент успешных попыток тренировки ($result = 100.0 * goals / attempts$). На основе этого результата навык "Heading" (Heading) увеличивается или уменьшается. Например, если результат тренировки (result) равен или больше 70%, то к навыку "Heading" прибавляются 5 единиц. Если результат между 60% и 70%, то добавляются 2 единицы. Если результат между 50% и 60%, то добавляется 1 единица. В остальных случаях (меньше 50%) к навыку "Heading" вычитаются 2 единицы.

6. Завершение тренировки: По окончании цикла тренировки логируется новое значение навыка "Heading" (Heading) после тренировки.

Эта формула моделирует процесс улучшения навыка удара мяча головой у футболиста на основе случайных параметров и результатов попыток, что отражает реалистичный подход к тренировке виртуального футболиста.

Метод **GameResult** моделирует результат игры между двумя командами (`team1` и `team2`) в футболе на основе навыков и характеристик игроков. Вот описание формулы, используемой в этом методе:

1. Начало игры: Метод принимает множество параметров, таких как навыки (`skill1`, `skill2`), количество отбитых мячей (`saves1`, `saves2`), навыки в отдельных аспектах (`finishing11`, `finishing12`, `finishing21`, `finishing22`, `def11`, `def12`, `def21`, `def22`, `head1`, `head2`, `dribbling1`, `dribbling2`). Эти параметры влияют на ход игры и результаты атак.

2. Определение атак: Рассчитывается количество атак для каждой команды на основе их общего навыка (`skill1`, `skill2`). Чем выше навык, тем больше атак создает команда, что повышает шансы на забитые голы.

3. Цикл атак: Для каждой команды (`team1` и `team2`) происходит цикл атак. В каждой атаке рассчитывается вероятность забития гола (`Pgoal`) с учетом различных факторов, таких как навыки дриблинга (`dribbling1`, `dribbling2`), навыки приема мяча головой (`head1`, `head2`), уровень защиты (`def11`, `def12`, `def21`, `def22`), количество отбитых мячей (`saves1`, `saves2`), и точность завершения атак (`finishing11`, `finishing12`, `finishing21`, `finishing22`).

4. Определение гола: Если вероятность забития гола (`Pgoal`) превышает определенный порог (в данном случае 8.0), то считается, что гол забит. Определяется автор забитого мяча (вратарь, защитник, нападающий и т. д.), и это отражается в результирующей строке (`res`), которая содержит информацию о голах каждой команды и авторах этих голов.

5. Логирование и возврат результата: В процессе игры логируются различные события, такие как вероятность забития гола, авторство голов, количество голов каждой команды, и т. д. В конце игры возвращается строка с деталями результатов.

Эта формула моделирует футбольный матч, учитывая разнообразные факторы, влияющие на исход игры, такие как навыки игроков, защитные действия и точность завершения атак.

6. Описание пользовательского интерфейса

Пользовательский интерфейс приведен на рис. 4 – 6.

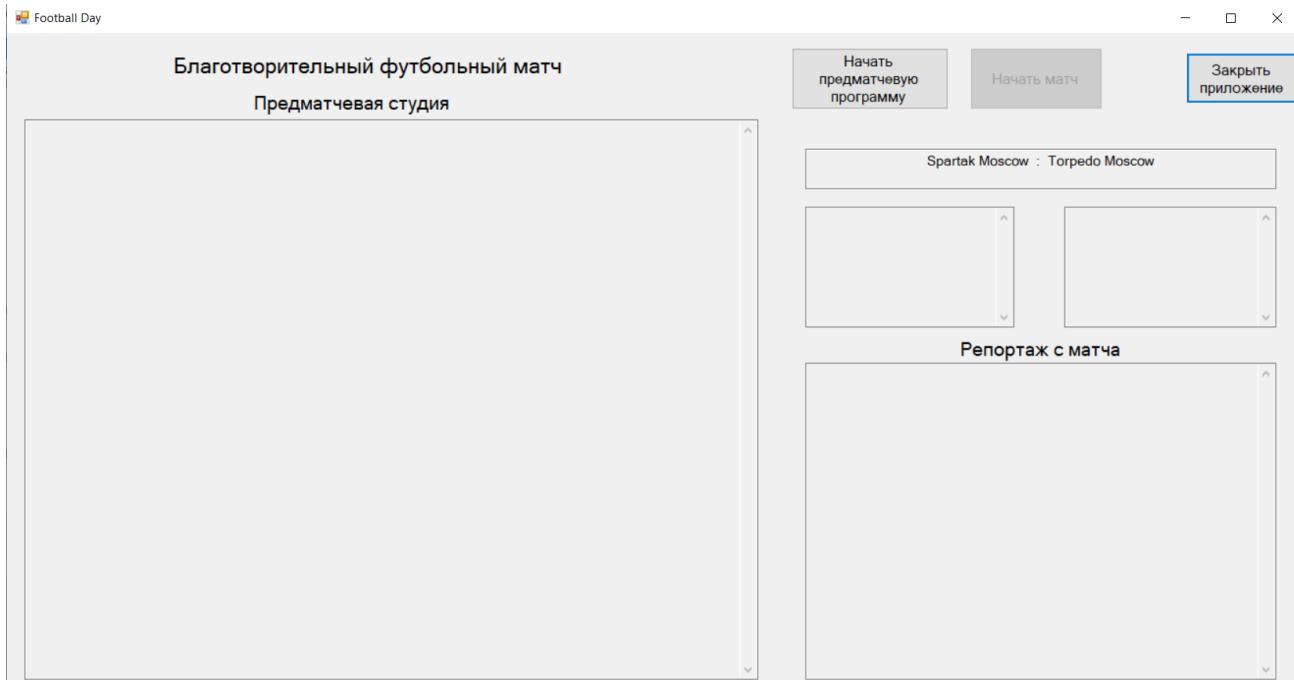


Рис. 4. Пользовательский интерфейс после запуска программы

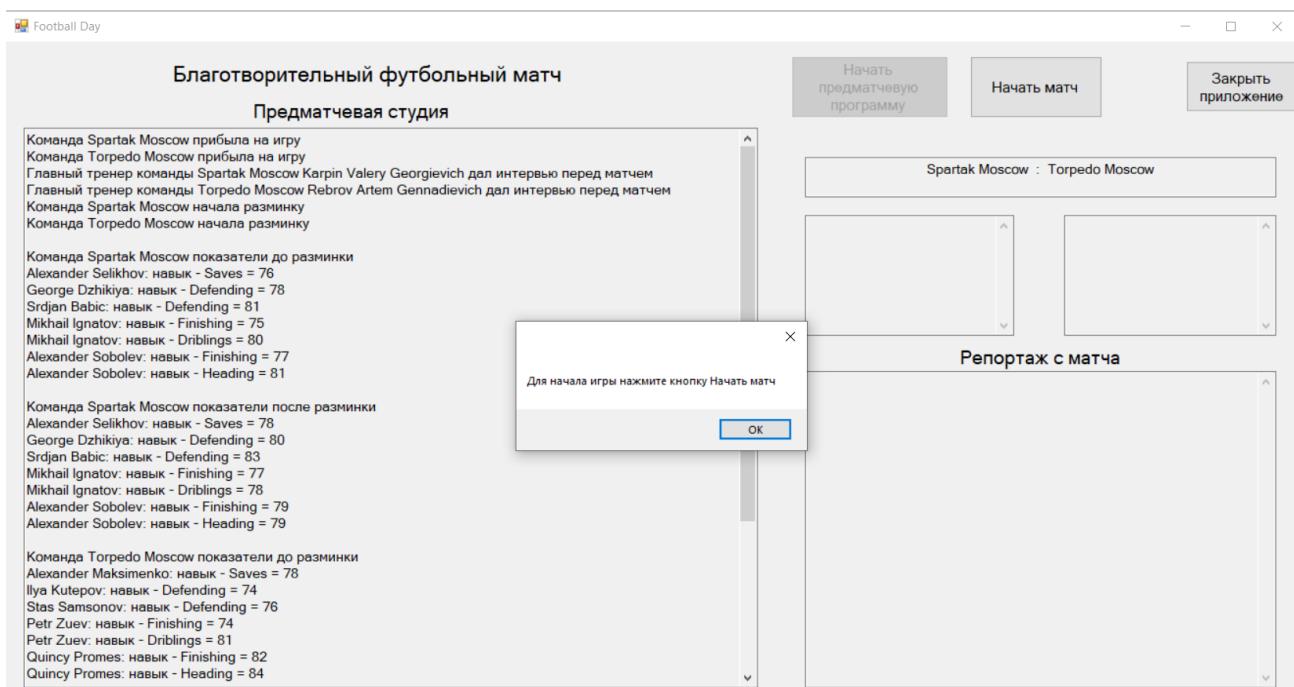


Рис. 5. Пользовательский интерфейс после завершения предматчевой программы

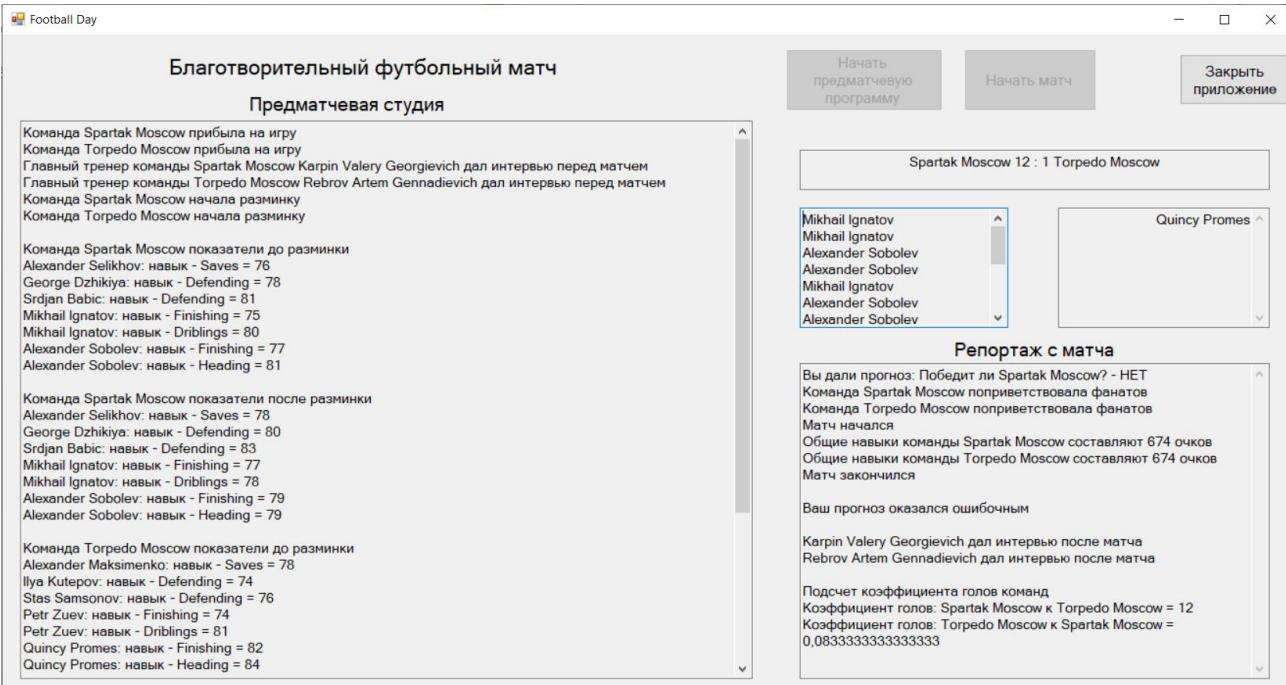


Рис. 6. Пользовательский интерфейс после завершения репортажа матча

После запуска программы пользователь может начать предматчевую программу, нажав на соответствующую кнопку. После этого можно начать репортаж с матча, нажав на кнопку «Начать матч». После нажатия на эту кнопку, можно сделать прогноз на игру, ответив «Да» или «Нет» на заданный вопрос. Закрыть программу можно с помощью нажатия на кнопку «Закрыть приложение».

В соответствующих TextBox'ах выводятся данные о предматчевой программе и самом матче.

7. Тестирование программы

Тестирование программы приведено на рис. 7 – 16. В тестах заметно, что в каждом случае результат моделирования различный, так как вносится вероятностный случайный параметр

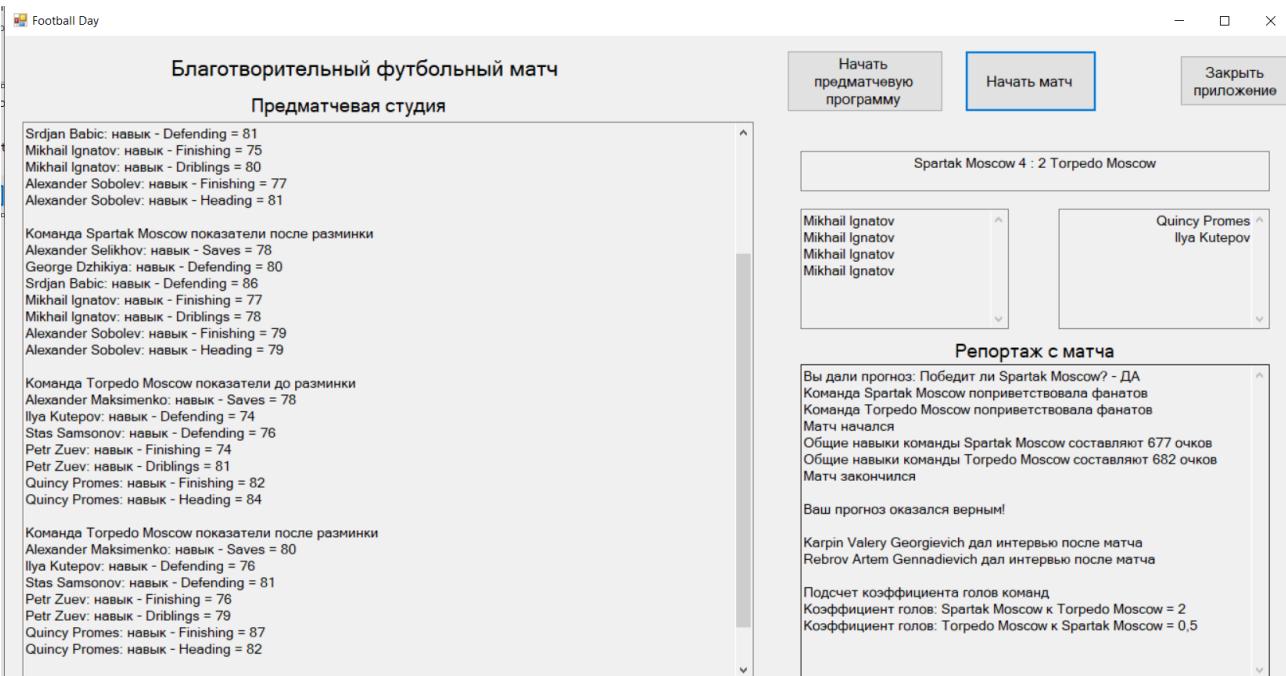


Рис. 7. Тест 1. Победа Спартака, верный прогноз, первый тип прогноза

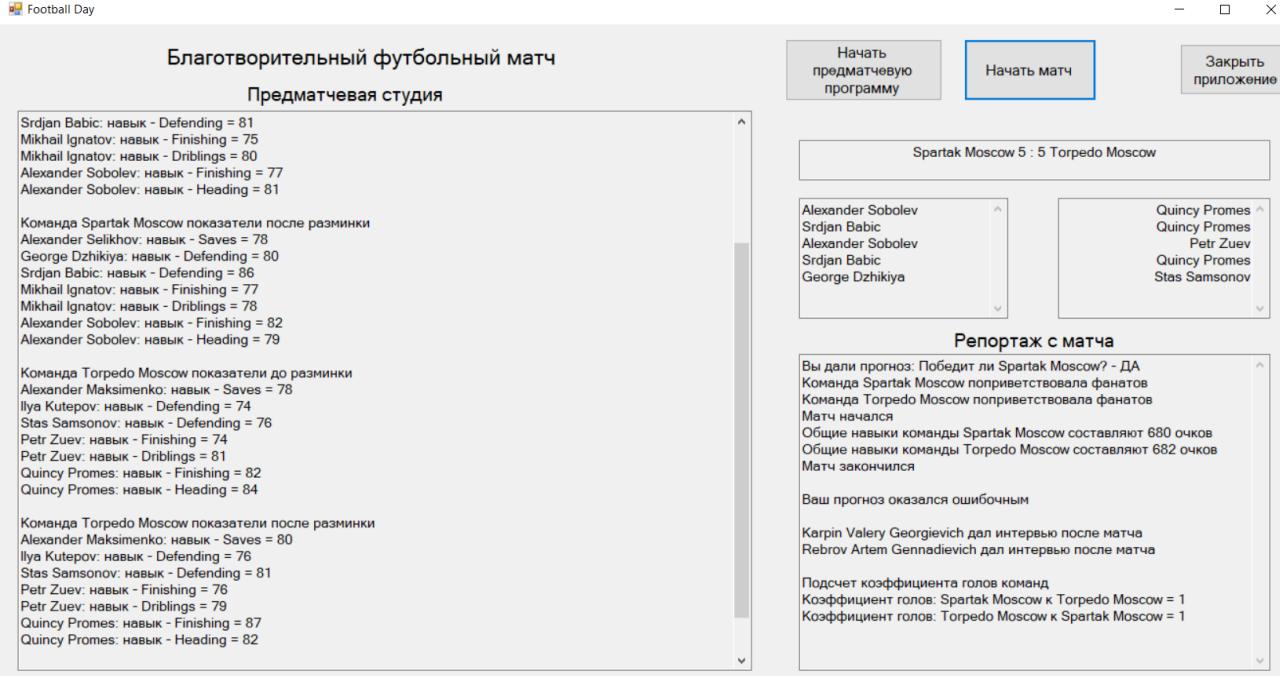


Рис. 8. Тест 2. Ничья, ошибочный прогноз

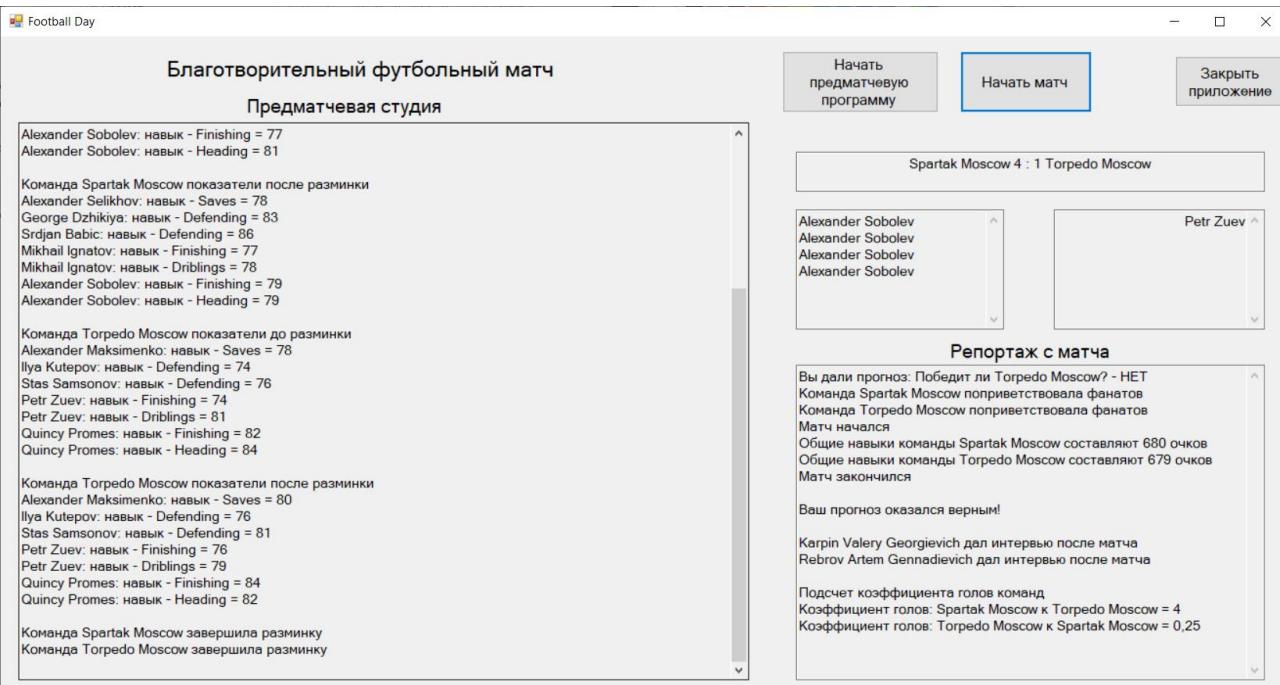


Рис. 9. Тест 3. Ничья, Победа Спартака, верный прогноз. Отличный от теста счет и авторы голов.
Второй тип прогноза

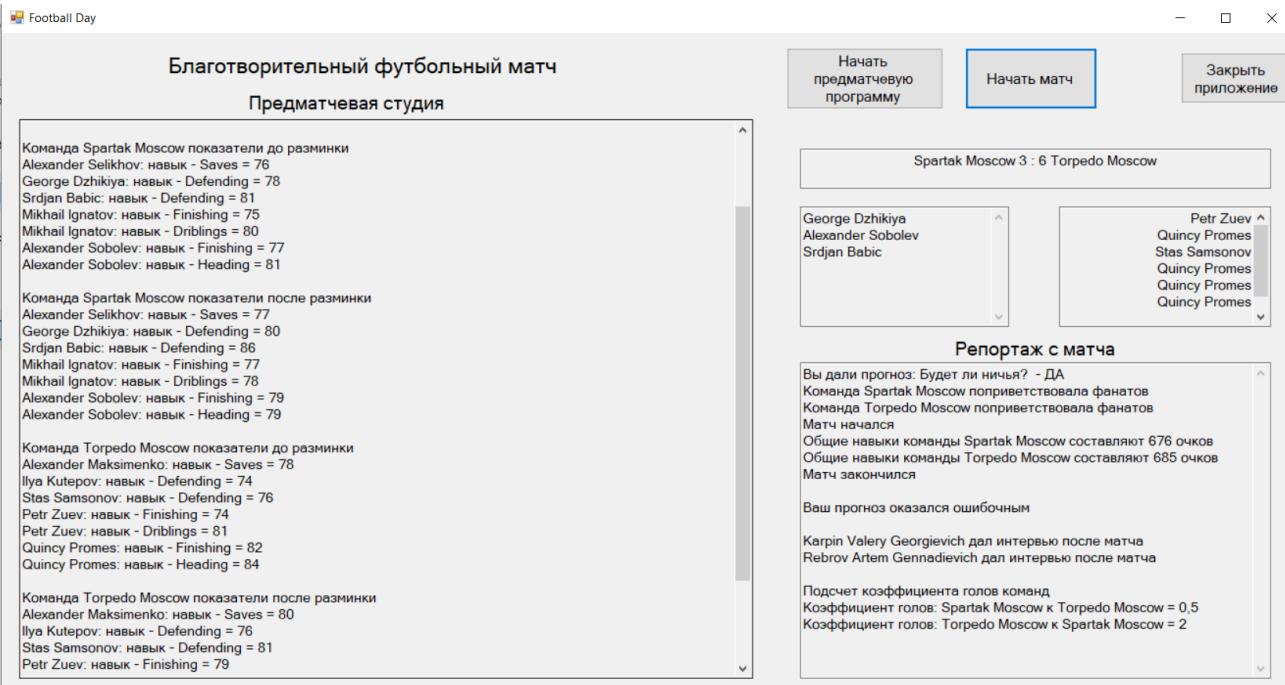


Рис. 10. Тест 4. Победа Торпедо, ошибочный прогноз. Третий тип прогноза

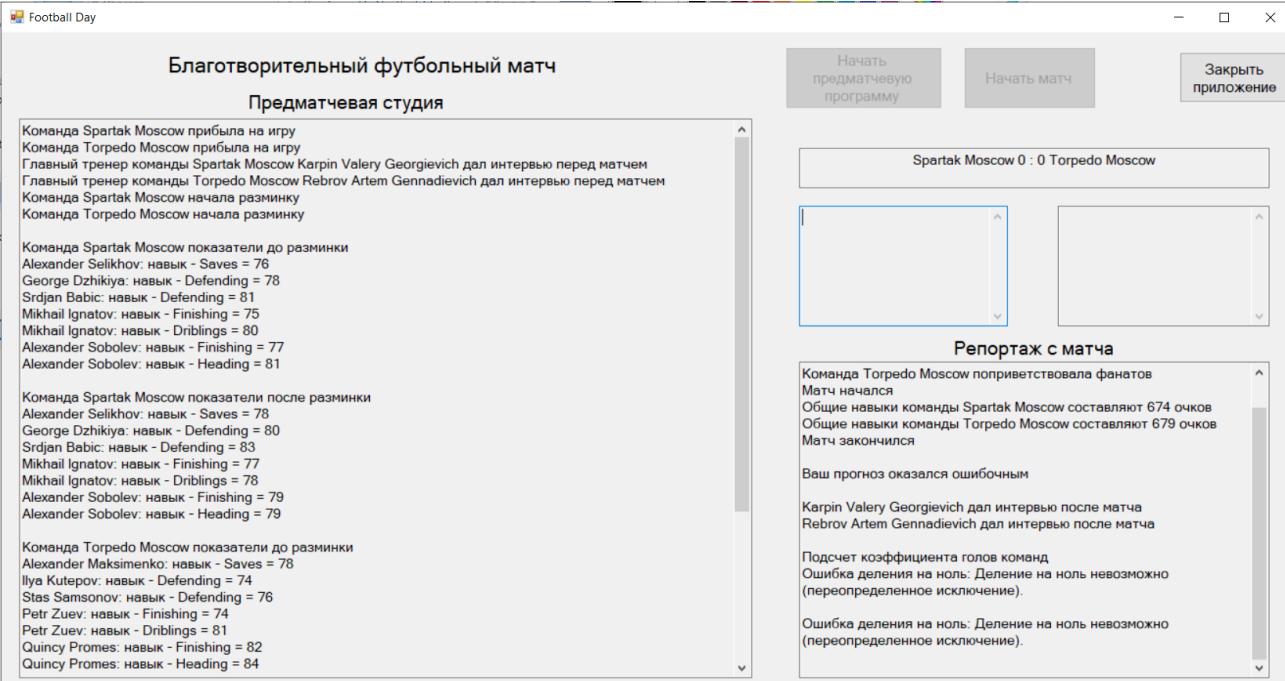


Рис. 11. Тест 5. Нулевая ничья, ошибочный прогноз, два исключения с делением на ноль

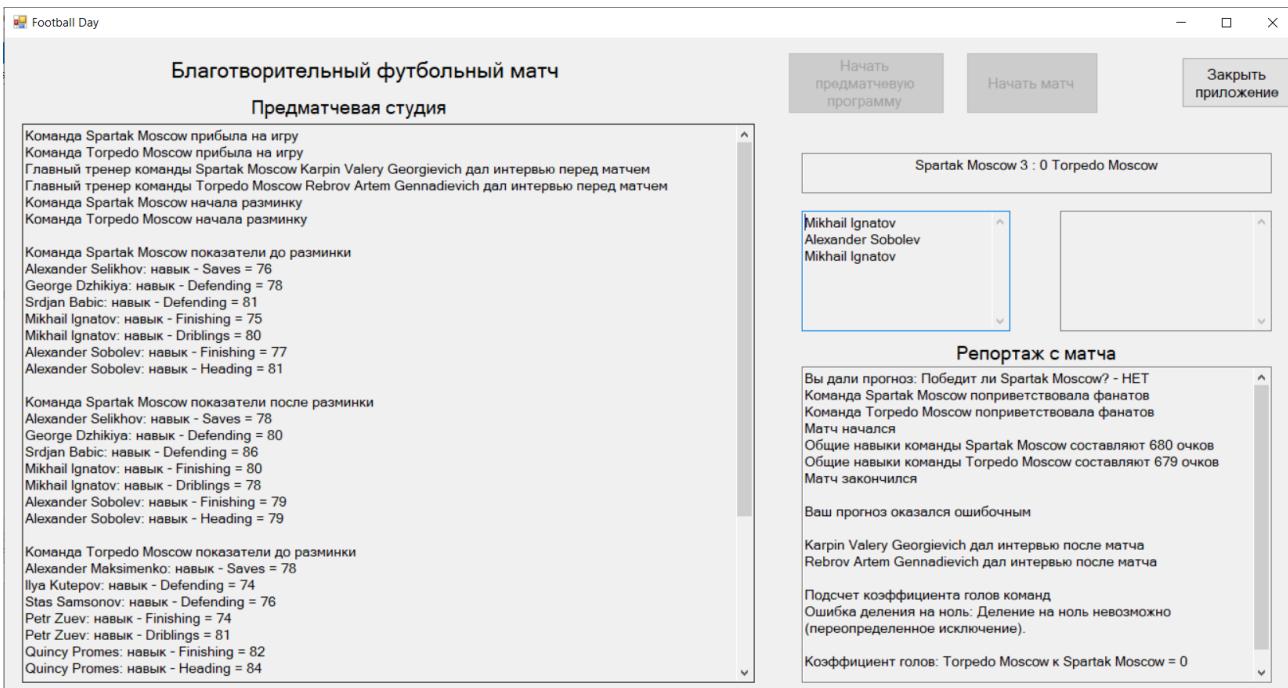


Рис. 12. Тест 6. Победа Спартака, ошибочный прогноз, ошибка деления на ноль

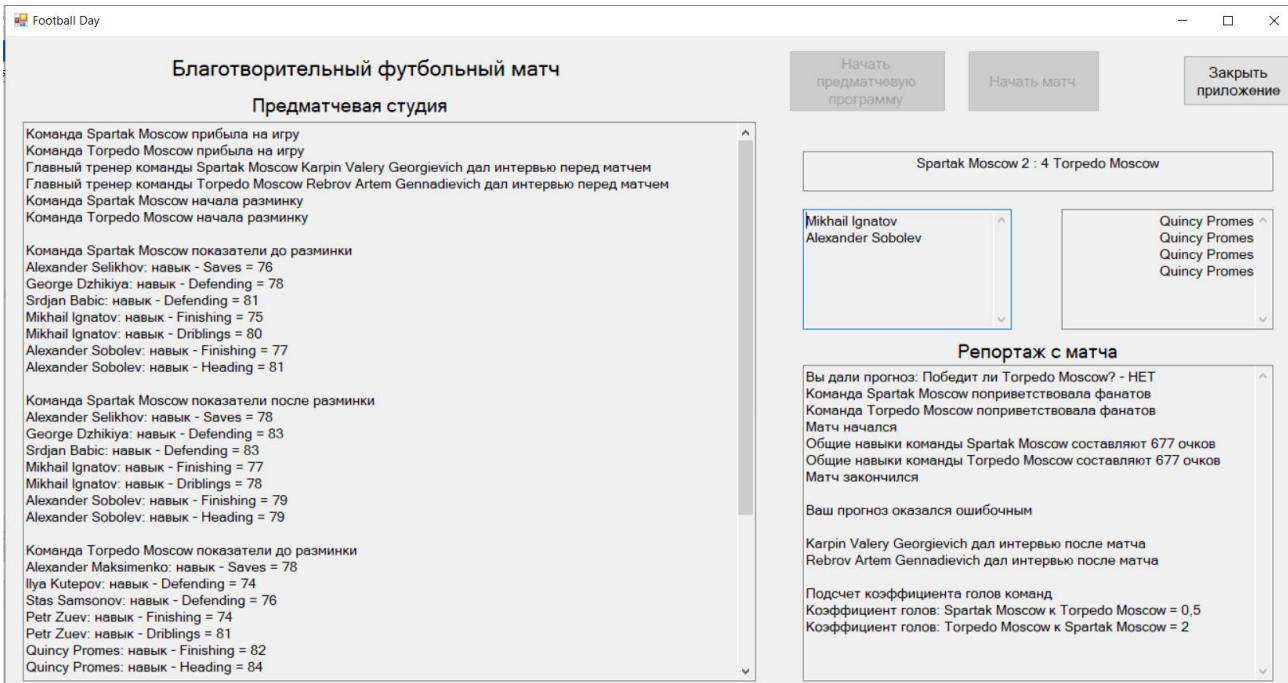


Рис. 13. Тест 7. Равное количество общих навыков, но результат не ничья

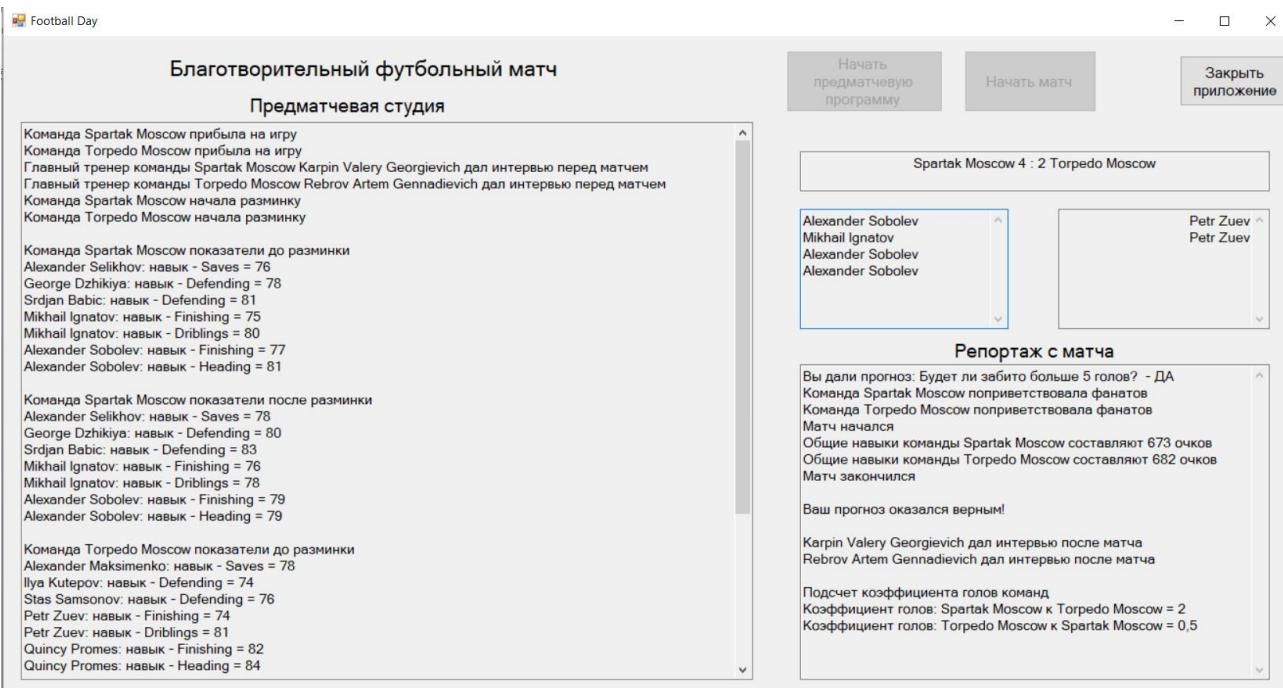


Рис. 14. Тест 8. Четвертый тип прогноза.

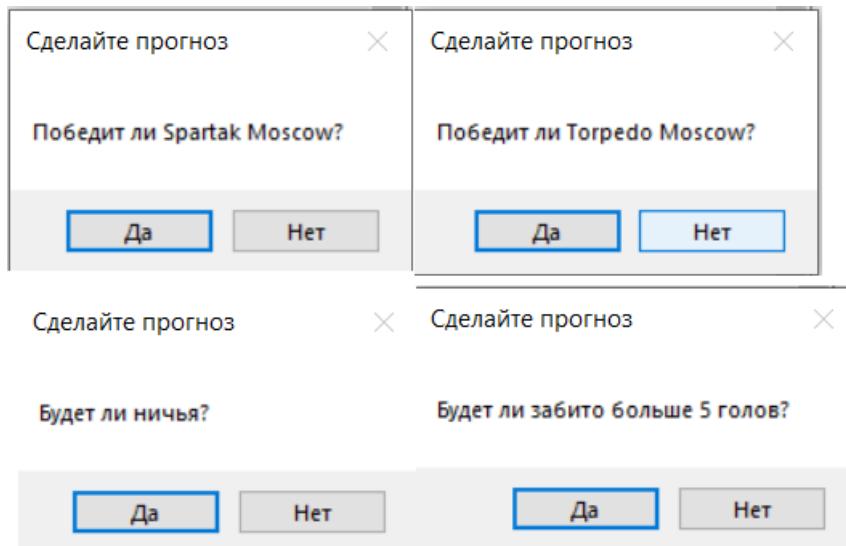


Рис. 15. Все типы прогнозов

```

15.05.2024 11:53:01 program started. log created
15.05.2024 11:53:03 Random number to attack1 = 0,938485843147377
15.05.2024 11:53:03 attack1 = 62
15.05.2024 11:53:03 Random number to attack2 = 0,966475314025989
15.05.2024 11:53:03 attack2 = 64
15.05.2024 11:53:03 6,08980796924333
15.05.2024 11:53:03 7,62500577558447
15.05.2024 11:53:03 9,29293893589528
15.05.2024 11:53:03 Team1. goal no. 1 by CF
15.05.2024 11:53:04 9,36977343145573
15.05.2024 11:53:04 Team1. goal no. 2 by CF
15.05.2024 11:53:04 6,3327187235816
15.05.2024 11:53:04 6,96735026541157
15.05.2024 11:53:04 8,08126074481188
15.05.2024 11:53:04 6,24773922553322
15.05.2024 11:53:04 7,84906860953864
15.05.2024 11:53:04 7,70964572901149
15.05.2024 11:53:04 7,34266054795398
15.05.2024 11:53:04 6,90160939300888
15.05.2024 11:53:04 6,32787238559267
15.05.2024 11:53:04 8,02276434204362
15.05.2024 11:53:04 7,97223394164673
15.05.2024 11:53:04 7,9675519134709
15.05.2024 11:53:04 7,19454497952048
15.05.2024 11:53:04 7,79443151310113
15.05.2024 11:53:04 7,40919967680804
15.05.2024 11:53:04 7,96389790824483
15.05.2024 11:53:09 program finished
15.05.2024 18:10:51 program finished
15.05.2024 18:10:54 program started. log created
15.05.2024 18:10:55 Karpin Valery Georgievich arrived at the match
15.05.2024 18:10:55 Alexander Selikhov arrived at the match
15.05.2024 18:10:55 George Dzhikiya arrived at the match
15.05.2024 18:10:55 Srdjan Babic arrived at the match
15.05.2024 18:10:55 Mikhail Ignatov arrived at the match
15.05.2024 18:10:55 Alexander Sobolev arrived at the match
15.05.2024 18:10:57 Rebrov Artem Gennadievich arrived at the match
15.05.2024 18:10:57 Alexander Maksimenko arrived at the match
15.05.2024 18:10:57 Ilya Kutepor arrived at the match
15.05.2024 18:10:57 Stas Samsonov arrived at the match
15.05.2024 18:10:57 Petr Zuev arrived at the match
15.05.2024 18:10:57 Quincy Promes arrived at the match
15.05.2024 18:10:58 Start Press Conference by Head Coach Team Spartak Moscow
15.05.2024 18:10:58 Correspondent: What do you expect from this game?
15.05.2024 18:10:58 Karpin Valery Georgievich: I'm waiting for a good game. I want to please our fans
15.05.2024 18:10:58 Correspondent: Is your team physically well prepared?
15.05.2024 18:10:58 Karpin Valery Georgievich: My team Spartak Moscow is ready to show spectacular football
15.05.2024 18:10:58 Finish Press Conference by Head Coach Team Spartak Moscow
15.05.2024 18:10:59 Start Press Conference by Head Coach Team Torpedo Moscow
15.05.2024 18:10:59 Correspondent: What do you expect from this game?
15.05.2024 18:10:59 Rebrov Artem Gennadievich: I'm waiting for a good game. I want to please our fans
15.05.2024 18:10:59 Correspondent: Is your team physically well prepared?
15.05.2024 18:10:59 Rebrov Artem Gennadievich: My team Torpedo Moscow is ready to show spectacular football
15.05.2024 18:10:59 Finish Press Conference by Head Coach Team Torpedo Moscow
15.05.2024 18:11:00 Alexander Selikhov dressed up
15.05.2024 18:11:00 George Dzhikiya dressed up

```

Рис. 16. Пример логирования программы

8. Выводы

Эта программа моделирует благотворительный матч по мини-футболу между двумя командами. Она представляет различных участников матча, таких как игроки, тренеры и персонал, а также включает логирование действий каждого участника.

1. Программа демонстрирует использование интерфейсов (IArrival, IWarmUp, IPressConference, IDress) для определения общих методов, которые должны быть реализованы классами, например, методы для прибытия на матч, проведения разминки и участия в пресс-конференции.

2. Классы Person, Player, Coach, Forward, Defender, Goalkeeper, CenterForward, Striker демонстрируют использование наследования для организации иерархии объектов. Например, класс Player наследует от Person и реализует интерфейсы IDress и IWarmUp, а классы Forward, Defender, Goalkeeper наследуют от Player.

3. В классе Coach переопределен метод ApplauseForTheFans, что позволяет каждому типу человека иметь свою уникальную реакцию на события.

4. Программа использует логирование с помощью класса Log, чтобы записывать различные события, такие как прибытие на матч, проведение разминки, участие в пресс-конференции и другие действия участников.

5. Класс Game моделирует игровой процесс, включая атаки команд и забитые голы. Программа использует случайные значения для определения успешности атак и результатов матча.

6. Программа содержит пользовательское исключение MyDivideByZeroException, которое наследует от стандартного исключения DivideByZeroException и используется для обработки ситуаций деления на ноль.

7. Программа генерирует случайные значения для моделирования реалистичных игровых ситуаций, таких как успех атаки, вероятность забить гол и другие параметры игры.

Эта программа демонстрирует принципы ООП, использование интерфейсов, наследование, логирование событий и обработку исключений в контексте моделирования благотворительного матча по мини-футболу.

Приложение А. Код иерархии классов.

```
using System;
using System.IO;

namespace WindowsFormsApp1
{
    // =====
    // класс для логирования программы
    public class Log
    {
        private static Log log;
        private Log() { }
        public static Log getLog()
        {
            if (log == null)
                log = new Log();
            return log;
        }
        public async void AddLog(string msg)
        {
            string path = "log.txt";
            using (StreamWriter writer = new StreamWriter(path, true))
            {
                await writer.WriteLineAsync(DateTime.Now.ToString() + " " + msg);
            }
        }
    }
    // =====

    // Определение интерфейсов
    // =====
    interface IArrival
    {
        void ArrivalToMatch();
    }

    interface IWarmUp
    {
        void StartWarmUp();
        void FinishWarmUp();
    }

    interface IPressConference
    {
        void GiveAnAnswer();
    }

    interface IDress
    {
        void DressUp();
    }
    // =====
```

```

public class Person : IArrival, IPressConference
{
    public string PersonName;
    public Person(string name)
    {
        PersonName = name;
    }

    public virtual void ApplauseForTheFans()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " applauded for the fans");
    }

    public void ArrivalToMatch()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " arrived at the match");
    }
    public void GiveAnAnswer()
    {
        var logger = Log.getLog();
        string answer1 = "It was an interesting match.";
        string answer2 = "Thanks to the fans for their support.";

        logger.AddLog("Start Press Conference");
        logger.AddLog("Correspondent: Are you happy with the game?");
        logger.AddLog(PersonName + ": " + answer1);
        logger.AddLog("Correspondent: What can you say about the fans?");
        logger.AddLog(PersonName + ": " + answer2);
        logger.AddLog("Finish Press Conference");
    }
}

public class Player : Person, IDress, IWarmUp
{
    public int PhysicalForm;
    public int Finishing;
    public int Defending;
    public Player(string name, int physicalForm, int finishing, int defending) :
base(name)
    {
        PhysicalForm = physicalForm;
        Finishing = finishing;
        Defending = defending;
    }

    public int TotalBasedSkills()
    {
        return PhysicalForm + Finishing + Defending;
    }
    public void DressUp()
    {
}

```

```

        var logger = Log.getLog();
        logger.AddLog(PersonName + " dressed up");
    }

    public void StartWarmUp()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " started warm up");
    }

    public void FinishWarmUp()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " finished warm up");
    }

}

public class Coach : Person, IWarmUp
{
    public string Team;
    public Coach(string name, string team) : base(name)
    {
        Team = team;
    }
    public override void ApplauseForTheFans()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " greeted the fans");
    }
    public void StartWarmUp()
    {
        var logger = Log.getLog();
        logger.AddLog("Head Coach " + PersonName + " started warm up");
    }

    public void FinishWarmUp()
    {
        var logger = Log.getLog();
        logger.AddLog("Head Coach " + PersonName + " finished warm up");
    }

    public void GiveAnAnswer(string replica)
    {
        var logger = Log.getLog();
        string answer1 = "I'm waiting for a good game. I want to please our fans";
        string answer2 = "My team " + Team + " is ready to show spectacular
football";

        logger.AddLog("Start Press Conference by " + replica);
        logger.AddLog("Correspondent: What do you expect from this game?");
        logger.AddLog(PersonName + ": " + answer1);
        logger.AddLog("Correspondent: Is your team physically well prepared?");
        logger.AddLog(PersonName + ": " + answer2);
    }
}

```

```

        logger.AddLog("Finish Press Conference by" + replica);
    }
}

public class Forward : Player
{
    public string skill = "Finishing";
    public Forward(string name, int physicalForm, int finishing, int defending) :
base(name, physicalForm, finishing, defending)
    {
    }

    public void TrainFinishing()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " finishing before train = " + Finishing);
        Random rand = new Random();
        double randomNumber, result;
        int attempts = 50;
        int goals = 0;

        for (int i = 0; i < attempts; i++)
        {
            randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
            if ((randomNumber + PhysicalForm / 100.0) >= 1.0)
            {
                goals++;
            }
        }

        result = 100.0 * goals / attempts;
        if (result >= 80)
        {
            Finishing = Finishing + 5;
        }
        else
        {
            if (result >= 60)
            {
                Finishing = Finishing + 2;
            }
            else
            {
                if (result >= 50)
                {
                    Finishing = Finishing + 1;
                }
                else
                {
                    Finishing = Finishing - 2;
                }
            }
        }
    }

    logger.AddLog(PersonName + " finishing after train = " + Finishing);
}

```

```

    }

}

public class Defender : Player
{
    public string skill = "Defending";
    public Defender(string name, int physicalForm, int finishing, int defending) : base(name, physicalForm, finishing, defending)
    {
    }

    public void TrainDefending()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " defending before train = " + Defending);
        Random rand = new Random();
        double randomNumber, result;
        int attempts = 50;
        int goals = 0;

        for (int i = 0; i < attempts; i++)
        {
            randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
            if ((randomNumber + PhysicalForm / 100.0) >= 1.0)
            {
                goals++;
            }
        }

        result = 100.0 * goals / attempts;
        if (result >= 75)
        {
            Defending = Defending + 5;
        }
        else
        {
            if (result >= 60)
            {
                Defending = Defending + 2;
            }
            else
            {
                if (result >= 40)
                {
                    Defending = Defending + 1;
                }
                else
                {
                    Defending = Defending - 2;
                }
            }
        }
        logger.AddLog(PersonName + " defending after train = " + Defending);
    }
}

```

```

}

public class Goalkeeper : Player
{
    public int Saves;
    public string skill = "Saves";
    public Goalkeeper(string name, int physicalForm, int finishing, int defending, int saves) : base(name, physicalForm, finishing, defending)
    {
        Saves = saves;
    }

    public void TrainSaves()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " saves before train = " + Saves);
        Random rand = new Random();
        double randomNumber, result;
        int attempts = 50;
        int goals = 0;

        for (int i = 0; i < attempts; i++)
        {
            randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
            if ((randomNumber + PhysicalForm / 100.0) >= 1.0)
            {
                goals++;
            }
        }

        result = 100.0 * goals / attempts;
        if (result >= 90)
        {
            Saves = Saves + 5;
        }
        else
        {
            if (result >= 60)
            {
                Saves = Saves + 2;
            }
            else
            {
                if (result >= 40)
                {
                    Saves = Saves + 1;
                }
                else
                {
                    Saves = Saves - 2;
                }
            }
        }
    }

    logger.AddLog(PersonName + " saves after train = " + Saves);
}

```

```

    }

}

public class CenterForward : Forward
{
    public string skill2 = "Dribblings";
    public int Dribblings;
    public CenterForward(string name, int physicalForm, int finishing, int defending,
int dribblings) : base(name, physicalForm, finishing, defending)
    {
        Dribblings = dribblings;
    }

    public void TrainDribblings()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " Dribblings before train = " + Dribblings);
        Random rand = new Random();
        double randomNumber, result;
        int attempts = 50;
        int goals = 0;

        for (int i = 0; i < attempts; i++)
        {
            randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
            if ((randomNumber / 100.0) >= 1.0)
            {
                goals++;
            }
        }

        result = 100.0 * goals / attempts;
        if (result >= 75)
        {
            Dribblings = Dribblings + 5;
        }
        else
        {
            if (result >= 55)
            {
                Dribblings = Dribblings + 2;
            }
            else
            {
                if (result >= 45)
                {
                    Dribblings = Dribblings + 1;
                }
                else
                {
                    Dribblings = Dribblings - 2;
                }
            }
        }
    }
}

```

```

        logger.AddLog(PersonName + " Driblings after train = " + Driblings);
    }
}

public class Striker : Forward
{
    public string skill2 = "Heading";
    public int Heading;
    public Striker(string name, int physicalForm, int finishing, int defending, int heading) : base(name, physicalForm, finishing, defending)
    {
        Heading = heading;
    }

    public void TrainHeading()
    {
        var logger = Log.getLog();
        logger.AddLog(PersonName + " Heading before train = " + Heading);
        Random rand = new Random();
        double randomNumber, result;
        int attempts = 50;
        int goals = 0;

        for (int i = 0; i < attempts; i++)
        {
            randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
            if ((randomNumber / 100.0) >= 1.0)
            {
                goals++;
            }
        }

        result = 100.0 * goals / attempts;
        if (result >= 70)
        {
            Heading = Heading + 5;
        }
        else
        {
            if (result >= 60)
            {
                Heading = Heading + 2;
            }
            else
            {
                if (result >= 50)
                {
                    Heading = Heading + 1;
                }
                else
                {
                    Heading = Heading - 2;
                }
            }
        }
    }
}

```

```

        }
        logger.AddLog(PersonName + " Heading after train = " + Heading);
    }
}

public class Game
{
    public int goals_team1 = 0;
    public int goals_team2 = 0;
    public string GameResult(int skill1, int skill2, int saves1, int saves2, int
finishing11, int finishing12,
                           int finishing21, int finishing22, int def11, int def12, int def21, int def22,
int head1, int head2,
                           int dribbling1, int dribbling2)
    {
        // общий навык команды влияет на то, как много атак она создаст за игру
        // больше навык => больше атак => больше возможностей забить гол
        string res = "";
        int attack1 = 0;
        int attack2 = 0;
        var logger = Log.getLog();

        Random rand = new Random();
        double randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
        attack1 = (int)(randomNumber * skill1 / 10.0);
        logger.AddLog("Random number to attack1 = " + randomNumber);
        logger.AddLog("attack1 = " + attack1);

        randomNumber = rand.NextDouble() * (1.1 - 0.9) + 0.9;
        attack2 = (int)(randomNumber * skill2 / 10.0);
        logger.AddLog("Random number to attack2 = " + randomNumber);
        logger.AddLog("attack2 = " + attack2 + "\n");

        // цикл по всем атакам команды team1
        int goal;

        string name = "";
        double Pgoal;

        for (int i = 1; i <= attack1; i++)
        {
            Pgoal = ((rand.NextDouble() * (1.1 - 0.9) + 0.9) * dribbling1 * (100 -
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * def12)
                     + (rand.NextDouble() * (1.1 - 0.9) + 0.9) * head1 * (100 -
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * def22)
                     + (100 - (rand.NextDouble() * (1.1 - 0.9) + 0.9) * saves2) *
                     ((rand.NextDouble() * (1.1 - 0.9) + 0.9) * finishing11 +
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * finishing21)) / 1000;
            logger.AddLog(Pgoal.ToString());
            if (Pgoal > 8.0)
            {
                goals_team1++;
                // определение автора забитого мяча
            }
        }
    }
}

```

```

        goal = rand.Next(1, 101);
        if ((goal >= 1) && (goal < 2))
        {
            name = "GK";
        }
        if ((goal >= 2) && (goal <= 11))
        {
            name = "DEF1";
        }
        if ((goal >= 12) && (goal <= 21))
        {
            name = "DEF2";
        }
        if ((goal >= 22) && (goal <= 55))
        {
            name = "CF";
        }
        if ((goal >= 56) && (goal <= 100))
        {
            name = "ST";
        }
        res += goals_team1.ToString() + " 1" + name + "\r\n";
        logger.AddLog("Team1. goal no. " + goals_team1.ToString() + " by " +
name);
    }

}

// цикл по всем атакам команды team2
for (int i = 1; i <= attack2; i++)
{
    Pgoal = ((rand.NextDouble() * (1.1 - 0.9) + 0.9) * dribbling2 * (100 -
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * def11)
           + (rand.NextDouble() * (1.1 - 0.9) + 0.9) * head2 * (100 -
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * def21)
           + (100 - (rand.NextDouble() * (1.1 - 0.9) + 0.9) * saves1) *
((rand.NextDouble() * (1.1 - 0.9) + 0.9) * finishing12 +
(rand.NextDouble() * (1.1 - 0.9) + 0.9) * finishing22)) / 1000;
    logger.AddLog(Pgoal.ToString());

    if (Pgoal > 8.0)
    {
        goals_team2++;
        // определение автора забитого мяча
        goal = rand.Next(1, 101);
        if ((goal >= 1) && (goal < 2))
        {
            name = "GK";
        }
        if ((goal >= 2) && (goal <= 11))
        {
            name = "DEF1";
        }
    }
}

```

```

        if ((goal >= 12) && (goal <= 21))
        {
            name = "DEF2";
        }
        if ((goal >= 22) && (goal <= 55))
        {
            name = "CF";
        }
        if ((goal >= 56) && (goal <= 100))
        {
            name = "ST";
        }
        res += goals_team2.ToString() + " 2" + name + "\r\n";
        logger.AddLog("Team2. goal no. " + goals_team2.ToString() + " by " +
name);
    }
}
logger.AddLog("\n" + res);
return res;
}
}

public class MyDivideByZeroException : DivideByZeroException
{
    public MyDivideByZeroException() : base("Деление на ноль невозможно
(переопределенное исключение).")
    {
        // Конструктор без параметров вызывает базовый конструктор с сообщением об
ошибке
    }

    public MyDivideByZeroException(string message) : base(message)
    {
        // Конструктор с параметром для сообщения об ошибке
    }
}
}

```

Приложение Б. Код основной программы

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace WindowsFormsApp1
{

    public partial class Form1 : Form
    {
        public Form1()
        {
            var logger = Log.getLog();
            logger.AddLog("program started. log created");
            InitializeComponent();
            textBox4.TextAlign = HorizontalAlignment.Center;
            textBox4.Text = coach_team1.Team + " " + " : " + " " + coach_team2.Team;
            button3.Enabled = false;
            this.StartPosition = FormStartPosition.CenterScreen;
        }

        // Создание экземпляров классов
        Coach coach_team1 = new Coach("Karpin Valery Georgievich", "Spartak Moscow");
        Defender defender1_team1 = new Defender("George Dzhikiya", 4, 20, 78);
        Defender defender2_team1 = new Defender("Srdjan Babic", 5, 18, 81);
        CenterForward centerForward_team1 = new CenterForward("Mikhail Ignatov", 4, 75,
34, 80);
        Striker striker_team1 = new Striker("Alexander Sobolev", 5, 77, 15, 81);
        Goalkeeper goalkeeper_team1 = new Goalkeeper("Alexander Selikhov", 4, 5, 6, 76);

        Coach coach_team2 = new Coach("Rebrov Artem Gennadievich", "Torpedo Moscow");
        Defender defender1_team2 = new Defender("Ilya Kutepov", 3, 31, 74);
        Defender defender2_team2 = new Defender("Stas Samsonov", 5, 18, 76);
        CenterForward centerForward_team2 = new CenterForward("Petr Zuev", 4, 74, 27, 81);
        Striker striker_team2 = new Striker("Quincy Promes", 5, 82, 11, 84);
        Goalkeeper goalkeeper_team2= new Goalkeeper("Alexander Maksimenko", 5, 4, 8, 78);

        Forward forward = new Forward("Ronaldo", 2, 78, 20);
        Player player = new Player("Stas Karpow", 5, 20, 10);

        // кнопка Finish
        private void button1_Click(object sender, EventArgs e)
        {
            var logger = Log.getLog();
```

```

        logger.AddLog("program finished\n");
        Close();
    }

    private async void button2_Click(object sender, EventArgs e)
    {
        button2.Enabled = false;
        // прибытие команд
        //

=====

        coach_team1.ArrivalToMatch();
        goalkeeper_team1.ArrivalToMatch();
        defender1_team1.ArrivalToMatch();
        defender2_team1.ArrivalToMatch();
        centerForward_team1.ArrivalToMatch();
        striker_team1.ArrivalToMatch();
        textBox1.Text += "Команда " + coach_team1.Team + " прибыла на игру\r\n";
        await Task.Delay(1000);

        coach_team2.ArrivalToMatch();
        goalkeeper_team2.ArrivalToMatch();
        defender1_team2.ArrivalToMatch();
        defender2_team2.ArrivalToMatch();
        centerForward_team2.ArrivalToMatch();
        striker_team2.ArrivalToMatch();
        textBox1.Text += "Команда " + coach_team2.Team + " прибыла на игру\r\n";
        await Task.Delay(1000);
        //

=====

        // пресса до матча у тренеров
        //

=====

        string replica = "Head Coach Team " + coach_team1.Team;
        coach_team1.GiveAnAnswer(replica);
        textBox1.Text += "Главный тренер команды " + coach_team1.Team + " " +
coach_team1.PersonName + " дал интервью перед матчем\r\n";
        await Task.Delay(1000);

        replica = "Head Coach Team " + coach_team2.Team;
        coach_team2.GiveAnAnswer(replica);
        textBox1.Text += "Главный тренер команды " + coach_team2.Team + " " +
coach_team2.PersonName + " дал интервью перед матчем\r\n";
        await Task.Delay(1000);
        //

=====

        // разминка
        //

=====

        goalkeeper_team1.DressUp();
        defender1_team1.DressUp();
        defender2_team1.DressUp();

```

```

centerForward_team1.DressUp();
striker_team1.DressUp();

coach_team1.StartWarmUp();

goalkeeper_team1.StartWarmUp();
defender1_team1.StartWarmUp();
defender2_team1.StartWarmUp();
centerForward_team1.StartWarmUp();
striker_team1.StartWarmUp();

textBox1.Text += "Команда " + coach_team1.Team + " начала разминку\r\n";
await Task.Delay(1000);

goalkeeper_team2.DressUp();
defender1_team2.DressUp();
defender2_team2.DressUp();
centerForward_team2.DressUp();
striker_team2.DressUp();

coach_team2.StartWarmUp();

goalkeeper_team2.StartWarmUp();
defender1_team2.StartWarmUp();
defender2_team2.StartWarmUp();
centerForward_team2.StartWarmUp();
striker_team2.StartWarmUp();

textBox1.Text += "Команда " + coach_team2.Team + " начала разминку\r\n";
await Task.Delay(1000);

textBox1.Text += "\r\n";
textBox1.Text += "Команда " + coach_team1.Team + " показатели до
разминки\r\n";
    textBox1.Text += goalkeeper_team1.PersonName + ": навык - " +
goalkeeper_team1.skill + " = " + goalkeeper_team1.Saves.ToString() + "\r\n";
    textBox1.Text += defender1_team1.PersonName + ": навык - " +
defender1_team1.skill + " = " + defender1_team1.Defending.ToString() + "\r\n";
    textBox1.Text += defender2_team1.PersonName + ": навык - " +
defender2_team1.skill + " = " + defender2_team1.Defending.ToString() + "\r\n";
    textBox1.Text += centerForward_team1.PersonName + ": навык - " +
centerForward_team1.skill + " = " + centerForward_team1.Finishing.ToString() + "\r\n";
    textBox1.Text += centerForward_team1.PersonName + ": навык - " +
centerForward_team1.skill2 + " = " + centerForward_team1.Dribblings.ToString() + "\r\n";
    textBox1.Text += striker_team1.PersonName + ": навык - " + striker_team1.skill
+ " = " + striker_team1.Finishing.ToString() + "\r\n";
    textBox1.Text += striker_team1.PersonName + ": навык - " +
striker_team1.skill2 + " = " + striker_team1.Heading.ToString() + "\r\n";
    textBox1.Text += "\r\n";

goalkeeper_team1.TrainSaves();
defender1_team1.TrainDefending();
defender2_team1.TrainDefending();
centerForward_team1.TrainFinishing();

```

```

centerForward_team1.TrainDriblings();
striker_team1.TrainFinishing();
striker_team1.TrainHeading();

await Task.Delay(1000);
textBox1.Text += "Команда " + coach_team1.Team + " показатели после
разминки\r\n";
    textBox1.Text += goalkeeper_team1.PersonName + ": навык - " +
goalkeeper_team1.skill + " = " + goalkeeper_team1.Saves.ToString() + "\r\n";
    textBox1.Text += defender1_team1.PersonName + ": навык - " +
defender1_team1.skill + " = " + defender1_team1.Defending.ToString() + "\r\n";
    textBox1.Text += defender2_team1.PersonName + ": навык - " +
defender2_team1.skill + " = " + defender2_team1.Defending.ToString() + "\r\n";
    textBox1.Text += centerForward_team1.PersonName + ": навык - " +
centerForward_team1.skill + " = " + centerForward_team1.Finishing.ToString() + "\r\n";
    textBox1.Text += centerForward_team1.PersonName + ": навык - " +
centerForward_team1.skill2 + " = " + centerForward_team1.Driblings.ToString() + "\r\n";
    textBox1.Text += striker_team1.PersonName + ": навык - " + striker_team1.skill
+ " = " + striker_team1.Finishing.ToString() + "\r\n";
    textBox1.Text += striker_team1.PersonName + ": навык - " +
striker_team1.skill2 + " = " + striker_team1.Heading.ToString() + "\r\n";
    textBox1.Text += "\r\n";

await Task.Delay(1000);
textBox1.Text += "Команда " + coach_team2.Team + " показатели до
разминки\r\n";
    textBox1.Text += goalkeeper_team2.PersonName + ": навык - " +
goalkeeper_team2.skill + " = " + goalkeeper_team2.Saves.ToString() + "\r\n";
    textBox1.Text += defender1_team2.PersonName + ": навык - " +
defender1_team2.skill + " = " + defender1_team2.Defending.ToString() + "\r\n";
    textBox1.Text += defender2_team2.PersonName + ": навык - " +
defender2_team2.skill + " = " + defender2_team2.Defending.ToString() + "\r\n";
    textBox1.Text += centerForward_team2.PersonName + ": навык - " +
centerForward_team2.skill + " = " + centerForward_team2.Finishing.ToString() + "\r\n";
    textBox1.Text += centerForward_team2.PersonName + ": навык - " +
centerForward_team2.skill2 + " = " + centerForward_team2.Driblings.ToString() + "\r\n";
    textBox1.Text += striker_team2.PersonName + ": навык - " + striker_team2.skill
+ " = " + striker_team2.Finishing.ToString() + "\r\n";
    textBox1.Text += striker_team2.PersonName + ": навык - " +
striker_team2.skill2 + " = " + striker_team2.Heading.ToString() + "\r\n";
    textBox1.Text += "\r\n";

goalkeeper_team2.TrainSaves();
defender1_team2.TrainDefending();
defender2_team2.TrainDefending();
centerForward_team2.TrainFinishing();
centerForward_team2.TrainDriblings();
striker_team2.TrainFinishing();
striker_team2.TrainHeading();

await Task.Delay(1000);
textBox1.Text += "Команда " + coach_team2.Team + " показатели после
разминки\r\n";

```

```

        textBox1.Text += goalkeeper_team2.PersonName + ": навык - " +
goalkeeper_team2.skill + " = " + goalkeeper_team2.Saves.ToString() + "\r\n";
        textBox1.Text += defender1_team2.PersonName + ": навык - " +
defender1_team2.skill + " = " + defender1_team2.Defending.ToString() + "\r\n";
        textBox1.Text += defender2_team2.PersonName + ": навык - " +
defender2_team2.skill + " = " + defender2_team2.Defending.ToString() + "\r\n";
        textBox1.Text += centerForward_team2.PersonName + ": навык - " +
centerForward_team2.skill + " = " + centerForward_team2.Finishing.ToString() + "\r\n";
        textBox1.Text += centerForward_team2.PersonName + ": навык - " +
centerForward_team2.skill2 + " = " + centerForward_team2.Dribblings.ToString() + "\r\n";
        textBox1.Text += striker_team2.PersonName + ": навык - " + striker_team2.skill
+ " = " + striker_team2.Finishing.ToString() + "\r\n";
        textBox1.Text += striker_team2.PersonName + ": навык - " +
striker_team2.skill2 + " = " + striker_team2.Heading.ToString() + "\r\n";
        textBox1.Text += "\r\n";

        await Task.Delay(1000);
coach_team1.FinishWarmUp();
goalkeeper_team1.FinishWarmUp();
defender1_team1.FinishWarmUp();
defender2_team1.FinishWarmUp();
centerForward_team1.FinishWarmUp();
striker_team1.FinishWarmUp();

textBox1.Text += "Команда " + coach_team1.Team + " завершила разминку\r\n";
await Task.Delay(1000);

coach_team2.FinishWarmUp();
goalkeeper_team2.FinishWarmUp();
defender1_team2.FinishWarmUp();
defender2_team2.FinishWarmUp();
centerForward_team2.FinishWarmUp();
striker_team2.FinishWarmUp();

textBox1.Text += "Команда " + coach_team2.Team + " завершила разминку\r\n";
// =====
button3.Enabled = true;
MessageBox.Show("Для начала игры нажмите кнопку Начать матч");
}

Game game = new Game();
public delegate void MyDelegate();
private async void button3_Click(object sender, EventArgs e)
{
    button3.Enabled = false;
    var logger = Log.getLog();
    logger.AddLog("WINForm match started\n");
    int a = 0;
    string question1 = "Победит ли " + coach_team1.Team + "?";
    string question2 = "Победит ли " + coach_team2.Team + "?";
    string question3 = "Будет ли ничья? ";
    string question4 = "Будет ли забито больше 5 голов? ";
}

```

```

Random random = new Random();
int randomNumber = random.Next(1, 5); // Генерируем случайное число от 1 до 4

Console.WriteLine($"Случайное число: {randomNumber}");

switch (randomNumber)
{
    case 1:
        // Показываем MessageBox с двумя кнопками на выбор: "Да" и "Нет"
        DialogResult result = MessageBox.Show(question1, "Сделайте прогноз",
MessageBoxButtons.YesNo);

        // Проверяем результат выбора кнопки
        if (result == DialogResult.Yes)
        {
            a = 1;
            textBox5.Text += "Вы дали прогноз: " + question1 + " - ДА";
        }
        else if (result == DialogResult.No)
        {
            a = 2;
            textBox5.Text += "Вы дали прогноз: " + question1 + " - НЕТ";
        }

        break;
    case 2:
        // Показываем MessageBox с двумя кнопками на выбор: "Да" и "Нет"
        result = MessageBox.Show(question2, "Сделайте прогноз",
MessageBoxButtons.YesNo);

        // Проверяем результат выбора кнопки
        if (result == DialogResult.Yes)
        {
            a = 1;
            textBox5.Text += "Вы дали прогноз: " + question2 + " - ДА";
        }
        else if (result == DialogResult.No)
        {
            a = 2;
            textBox5.Text += "Вы дали прогноз: " + question2 + " - НЕТ";
        }

        break;
    case 3:
        // Показываем MessageBox с двумя кнопками на выбор: "Да" и "Нет"
        result = MessageBox.Show(question3, "Сделайте прогноз",
MessageBoxButtons.YesNo);

        // Проверяем результат выбора кнопки
        if (result == DialogResult.Yes)
        {
            a = 1;
            textBox5.Text += "Вы дали прогноз: " + question3 + " - ДА";
        }
}

```

```

        else if (result == DialogResult.No)
    {
        a = 2;
        textBox5.Text += "Вы дали прогноз: " + question3 + " - НЕТ";
    }
    break;
case 4:
    // Показываем MessageBox с двумя кнопками на выбор: "Да" и "Нет"
    result = MessageBox.Show(question4, "Сделайте прогноз",
    MessageBoxButtons.YesNo);

    // Проверяем результат выбора кнопки
    if (result == DialogResult.Yes)
    {
        a = 1;
        textBox5.Text += "Вы дали прогноз: " + question4 + " - ДА";
    }
    else if (result == DialogResult.No)
    {
        a = 2;
        textBox5.Text += "Вы дали прогноз: " + question4 + " - НЕТ";
    }
    break;
default:
    break;
}

// Создаем словарь для хранения соответствия позиции игрока и его имени
Dictionary<string, string> players = new Dictionary<string, string>
{
    { goalkeeper_team1.PersonName, "1GK" },
    { defender1_team1.PersonName, "1DEF1" },
    { defender2_team1.PersonName, "1DEF2" },
    { centerForward_team1.PersonName, "1CF" },
    { striker_team1.PersonName, "1ST" },
    { goalkeeper_team2.PersonName, "2GK" },
    { defender1_team2.PersonName, "2DEF1" },
    { defender2_team2.PersonName, "2DEF2" },
    { centerForward_team2.PersonName, "2CF" },
    { striker_team2.PersonName, "2ST" },
};

await Task.Delay(1000);
coach_team1.ApplauseForTheFans();
goalkeeper_team1.ApplauseForTheFans();
defender1_team1.ApplauseForTheFans();
defender2_team1.ApplauseForTheFans();
centerForward_team1.ApplauseForTheFans();
striker_team1.ApplauseForTheFans();
textBox5.Text += "\r\nКоманда " + coach_team1.Team + " поприветствовала
фанатов\r\n";
await Task.Delay(1000);

```

```

coach_team2.ApplauseForTheFans();
goalkeeper_team2.ApplauseForTheFans();
defender1_team2.ApplauseForTheFans();
defender2_team2.ApplauseForTheFans();
centerForward_team2.ApplauseForTheFans();
striker_team2.ApplauseForTheFans();
textBox5.Text += "Команда " + coach_team2.Team + " поприветствовала
фанатов\r\n";

```

```

    textBox5.Text += "Матч начался\r\n";
    await Task.Delay(1000);

```

```

        int totalSkills_team1 = goalkeeper_team1.TotalBasedSkills() +
goalkeeper_team1.Saves + defender1_team1.TotalBasedSkills()
+ defender2_team1.TotalBasedSkills() +
centerForward_team1.TotalBasedSkills() + centerForward_team1.Driblings
+ striker_team1.TotalBasedSkills() + striker_team1.Heading;
        textBox5.Text += "Общие навыки команды " + coach_team1.Team + " составляют " +
totalSkills_team1.ToString() + " очков\r\n";

```

```

        int totalSkills_team2 = goalkeeper_team2.TotalBasedSkills() +
goalkeeper_team2.Saves + defender1_team2.TotalBasedSkills()
+ defender2_team2.TotalBasedSkills() +
centerForward_team2.TotalBasedSkills() + centerForward_team2.Driblings
+ striker_team2.TotalBasedSkills() + striker_team2.Heading;
        textBox5.Text += "Общие навыки команды " + coach_team2.Team + " составляют " +
totalSkills_team2.ToString() + " очков\r\n";

```

```

        string resultGame = game.GameResult(totalSkills_team1, totalSkills_team2,
goalkeeper_team1.Saves,
            goalkeeper_team2.Saves, centerForward_team1.Finishing,
            centerForward_team2.Finishing, striker_team1.Finishing,
striker_team2.Finishing, defender1_team1.Defending,
            defender1_team2.Defending, defender2_team1.Defending,
defender2_team2.Defending, striker_team1.Heading,
            striker_team2.Heading, centerForward_team1.Driblings,
centerForward_team2.Driblings);

```

```

    await Task.Delay(1000);
//textBox1.Text += resultGame + "\r\n";

```

```

    // Разделяем исходную строку на строки по символу новой строки
//string[] lines = resultGame.Split('\r\n');
    string[] lines = resultGame.Split(new char[] { '\r', '\n' },
StringSplitOptions.RemoveEmptyEntries);
    // Создаем переменную для хранения преобразованной строки
    string output1 = "";
    string output2 = "";

```

```

    // Проходим по каждой строке в массиве lines
foreach (string line in lines)
{
    string[] parts = line.Split(' ');
    string searchValue = parts[1];

```

```

string key = players.FirstOrDefault(x => x.Value == searchValue).Key;

if (key != null)
{
    char firstChar = searchValue[0];
    if (firstChar == '1')
    {
        output1 += key + "\r\n";
    }
    if (firstChar == '2')
    {
        output2 += key + "\r\n";
    }
}
else
{
    MessageBox.Show("Value not found in the dictionary.");
}

textBox2.Text += output1;
textBox3.TextAlign = HorizontalAlignment.Right;
textBox3.Text += output2;

int lineCount1 = textBox2.Lines.Length;
if (lineCount1 > 0) { lineCount1--; }
int lineCount2 = textBox3.Lines.Length;
if (lineCount2 > 0) { lineCount2--; }

textBox4.TextAlign = HorizontalAlignment.Center;
textBox4.Text = coach_team1.Team + " " + lineCount1.ToString() + " : " +
lineCount2.ToString() + " " + coach_team2.Team;
textBox5.Text += "Матч закончился\r\n";
await Task.Delay(1000);
textBox5.Text += "\r\n";
await Task.Delay(1000);

switch (randomNumber)
{
    case 1:
        if ((lineCount1 > lineCount2 && a == 1) || (lineCount1 <= lineCount2
&& a == 2))
        {
            textBox5.Text += "Ваш прогноз оказался верным!\r\n";
        }
        else
        {
            textBox5.Text += "Ваш прогноз оказался ошибочным\r\n";
        }
        break;
    case 2:
        if ((lineCount1 < lineCount2 && a == 1) || (lineCount1 >= lineCount2
&& a == 2))
        {

```

```

        textBox5.Text += "Ваш прогноз оказался верным! \r\n";
    }
    else
    {
        textBox5.Text += "Ваш прогноз оказался ошибочным\r\n";
    }
    break;
case 3:
    if ((lineCount1 == lineCount2 && a == 1) || (lineCount1 != lineCount2
&& a == 2))
    {
        textBox5.Text += "Ваш прогноз оказался верным! \r\n";
    }
    else
    {
        textBox5.Text += "Ваш прогноз оказался ошибочным\r\n";
    }
    break;
case 4:
    if ((lineCount1 + lineCount2 > 5 && a == 1) || (lineCount1 +
lineCount2 <= 5 && a == 2))
    {
        textBox5.Text += "Ваш прогноз оказался верным! \r\n";
    }
    else
    {
        textBox5.Text += "Ваш прогноз оказался ошибочным\r\n";
    }
    break;
default:

    break;
}

// пресса после игры
// =====

```

```

MyDelegate multiDelegate = null;
// Добавляем методы в делегат
multiDelegate += coach_team1.ApplauseForTheFans;
multiDelegate += coach_team1.GiveAnAnswer;
multiDelegate += coach_team1.ApplauseForTheFans;
textBox5.Text += "\r\n";
textBox5.Text += coach_team1.PersonName + " дал интервью после матча\r\n";
await Task.Delay(1000);

multiDelegate += coach_team2.ApplauseForTheFans;
multiDelegate += coach_team2.GiveAnAnswer;
multiDelegate += coach_team2.ApplauseForTheFans;
textBox5.Text += coach_team2.PersonName + " дал интервью после матча\r\n";
await Task.Delay(1000);
// Вызываем делегат, что вызовет все добавленные методы
multiDelegate?.Invoke();

```

```

// =====

    textBox5.Text += "\r\nПодсчет коэффициента голов команд\r\n";
    try
    {
        // Попытка деления на ноль
        double result = DivideNumbers(lineCount1, lineCount2);
        textBox5.Text += "Коэффициент голов: " + coach_team1.Team + " к " +
coach_team2.Team + " = " + result + "\r\n";
    }
    catch (MyDivideByZeroException ex)
    {
        textBox5.Text += "Ошибка деления на ноль: ";
        textBox5.Text += (ex.Message);
        textBox5.Text += "\r\n\r\n";
        logger.AddLog(ex.Message + "\n");
    }

    try
    {
        // Попытка деления на ноль
        double result = DivideNumbers(lineCount2, lineCount1);
        textBox5.Text += "Коэффициент голов: " + coach_team2.Team + " к " +
coach_team1.Team + " = " + result + "\r\n";
    }
    catch (MyDivideByZeroException ex)
    {
        textBox5.Text += "Ошибка деления на ноль: ";
        textBox5.Text += (ex.Message);
        textBox5.Text += "\r\n";
        logger.AddLog(ex.Message + "\n");
    }
    logger.AddLog("WINForm match finished\n");
}

static double DivideNumbers(int numerator, int denominator)
{
    if (denominator == 0)
    {
        throw new MyDivideByZeroException(); // Генерация переопределенного
исключения
    }
    return (double)numerator / denominator;
}
}

```