### МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕУЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

### Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**Курсовая работа**

По дисциплине: Основы программирования

Тема: Моделирование спортивной игры(баскетбол)

Автор работы \_\_\_\_\_\_ Маркевич Александр Александрович

(подпись)

Руководитель проекта \_\_\_\_\_\_ Харитонов Сергей Дмитриевич

(подпись)

Оценка \_\_\_\_\_\_

Белгород 2024

**Содержание**

1 Введение…………………………………………………1

2 Объектная декомпозиция…………………………….....2

3 Диаграмма классов...........................................................3-4

4 Код программы..................................................................5-31

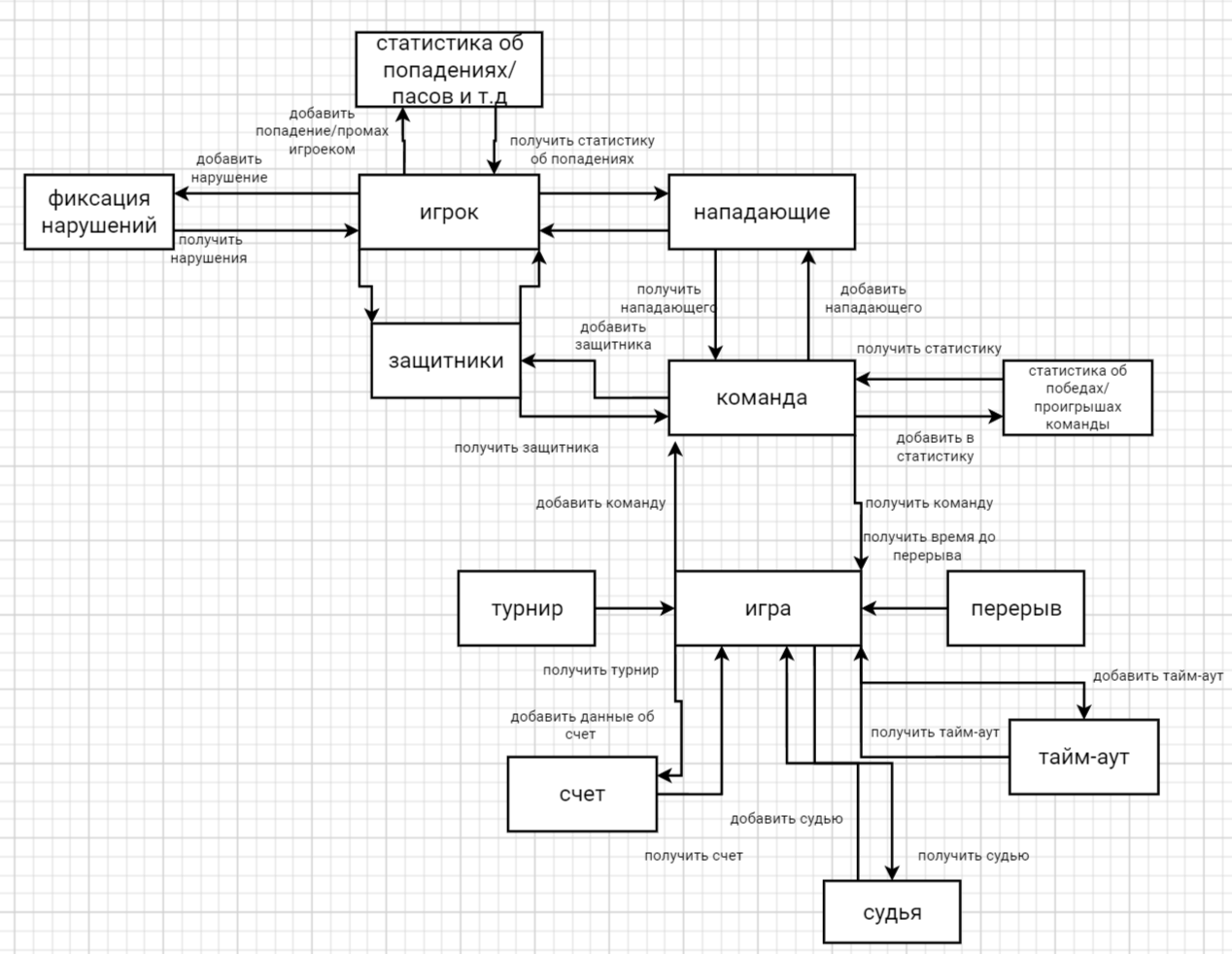
4 Заключение………………………………………………32

5 Список литературы……………………………………...33

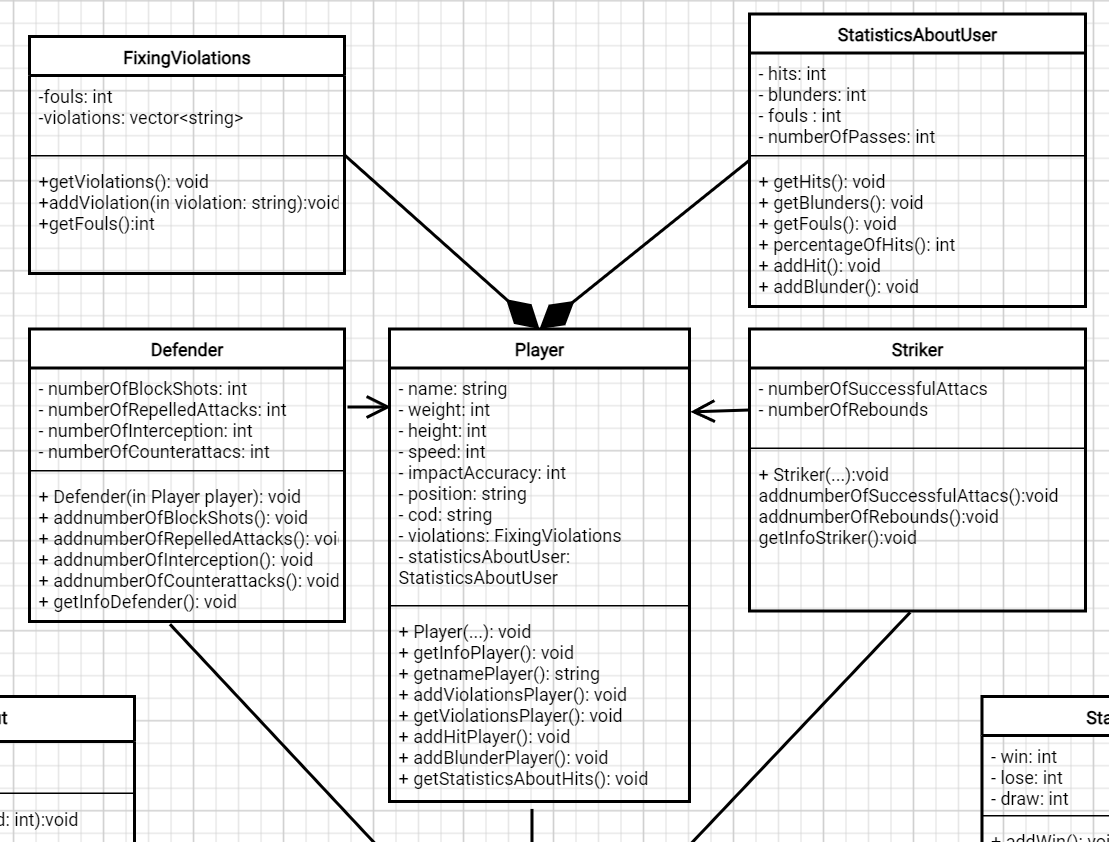
**Глава 1. Введение**

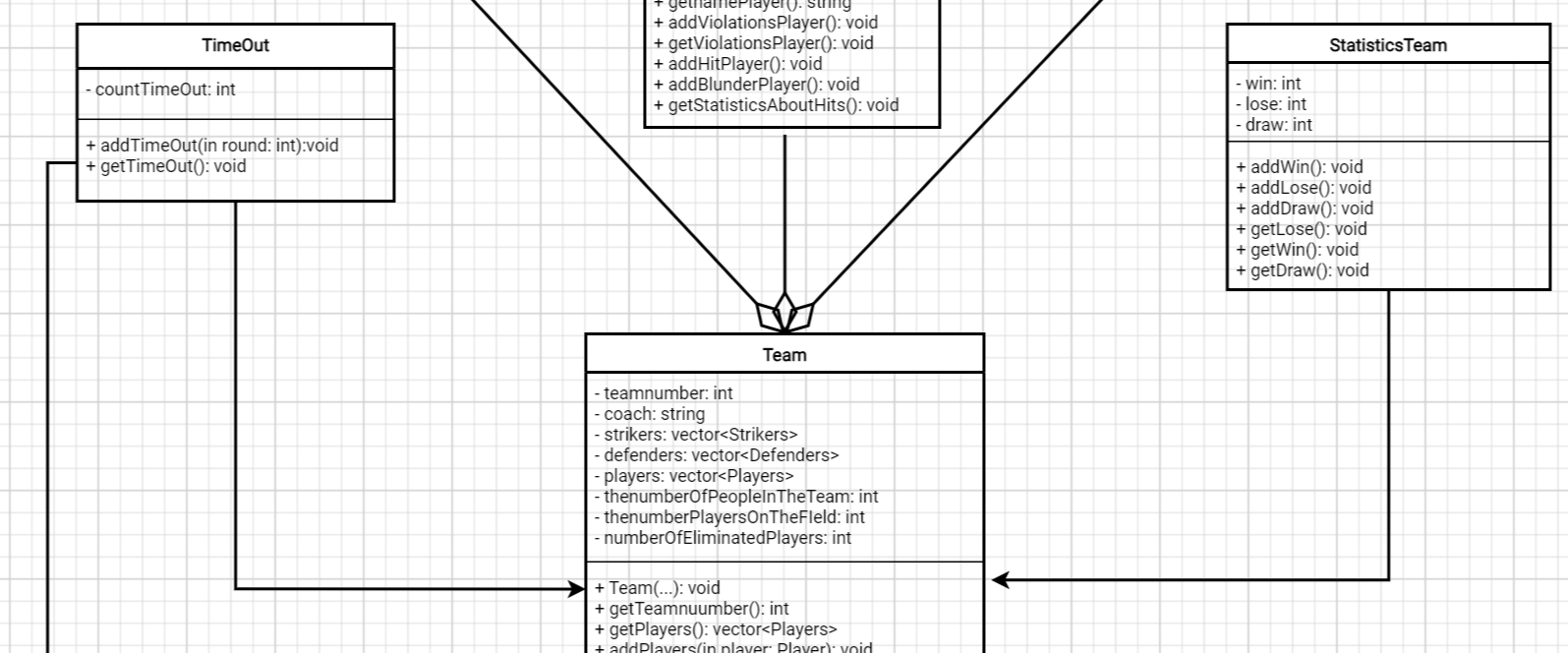
Баскетбол – это захватывающая командная игра, которая требует высокой физической подготовки, стратегического мышления и отличной координации движений. В нашей программе мы предлагаем вам окунуться в мир баскетбола, создавая собственные команды из виртуальных игроков и соревнуясь с ними на игровом поле.

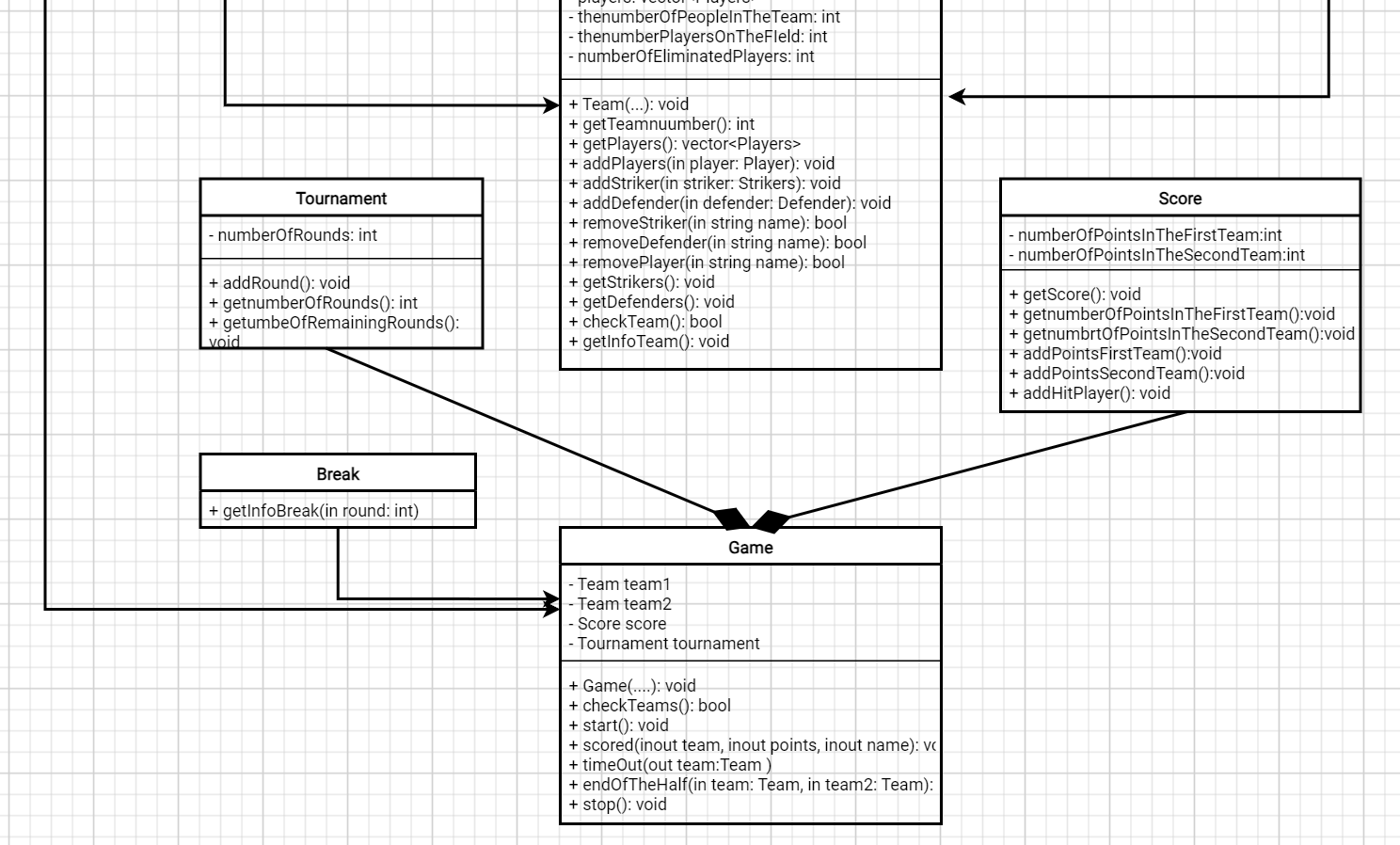
**Объектная декомпозиция**



**Диаграмма классов**







**Код программы**

#include <iostream>

#include <string>

#include <vector>

#include <windows.h>

using namespace std;

#define BROKE\_THE\_RULES\_MAX 6

#define MAXIMUM\_NUMBER\_OF\_PLAYERS\_IN\_A\_TEAM 12

#define THE\_MAXIMUM\_NUMBER\_OF\_PLAYERS\_ON\_THE\_FIELD 5

#define MAXIMUM\_NUMBER\_OF\_ROUNDS 4

#define BREAK\_BETWEEN\_THE\_HALVES\_OF\_THE\_GAME 15

#define BREAK\_BETWEEN\_QUARTERS 2

class FixingViolations

{

private:

int fouls = 0;

vector<string> violations;

public:

void getViolations() const

{

for (int i = 0; i < this->violations.size(); i++)

{

cout << violations[i] << endl;

}

}

void addViolation(const string &violation)

{

if (this->fouls == BROKE\_THE\_RULES\_MAX)

{

cout << "Игрок исключен из команды" << endl;

return;

}

this->fouls++;

this->violations.push\_back(violation);

}

int getFouls()

{

return this->fouls;

}

};

class StatisticsAboutUser

{

private:

int hits = 0;

int blunders = 0;

int fouls = 0;

int numberOfPasses = 0;

public:

void getHits() const

{

cout << "Количество попаданий: " << this->hits << endl;

}

void getBlunders() const

{

cout << "Количество промахов: " << this->blunders << endl;

}

void getFouls() const

{

cout << "Количество фолов: " << this->fouls << endl;

}

int percentageOfHits()

{

if (hits + blunders == 0)

{

return 0;

}

return this->hits \* 100 / (this->hits + this->blunders);

}

void addHit()

{

this->hits++;

}

void addBlunder()

{

this->blunders++;

}

};

class Player

{

private:

string name;

int weight;

int height;

int speed;

int impactAccuracy;

string position;

string cod;

FixingViolations violations;

StatisticsAboutUser statisticsAboutUser;

public:

Player(const string &name,

const int &weight,

const int &height,

const int &speed,

const string &position,

const string &cod,

const int &impactAccuracy)

{

this->name = name;

this->weight = weight;

this->height = height;

this->speed = speed;

this->impactAccuracy = impactAccuracy;

}

void getInfoPlayer()

{

cout << "Имя: " << this->name << endl;

cout << "Вес: " << this->weight << endl;

cout << "Рост: " << this->height << endl;

cout << "Скорость: " << this->speed << endl;

cout << "Точность удара: " << this->impactAccuracy << "%" << endl;

this->violations.getViolations();

cout << endl;

}

string getNamePlayer()

{

return this->name;

}

void addViolationPlayer(const string &violation)

{

this->violations.addViolation(violation);

}

void getViolationsPlayer()

{

cout << "Предупреждения: " << endl;

this->violations.getViolations();

}

void addHitPlayer()

{

this->statisticsAboutUser.addHit();

}

void addBlunderPlayer()

{

this->statisticsAboutUser.addBlunder();

}

void getStatisticsAboutHits()

{

this->statisticsAboutUser.getHits();

this->statisticsAboutUser.getBlunders();

this->impactAccuracy = statisticsAboutUser.percentageOfHits();

cout << "Процент попаданий: " << this->statisticsAboutUser.percentageOfHits() << endl;

}

};

class Striker : public Player

{

private:

int numberOfSuccessfulAttacks = 0;

int numberOfRebounds = 0;

public:

Striker(const Player &player) : Player(player) {}

void addNumberOfSuccessfulAttacks()

{

this->numberOfSuccessfulAttacks++;

}

void addNumberOfRebounds()

{

this->numberOfRebounds++;

}

void getInfoStriker()

{

cout << "Количество успешных атак: " << numberOfSuccessfulAttacks << endl;

cout << "Количество подборов: " << numberOfRebounds << endl;

}

};

class Defender : public Player

{

private:

int numberOfBlockShots = 0;

int numberOfRepelledAttacks = 0;

int numberOfInterception = 0;

int numberOfCounterattacks = 0;

public:

Defender(const Player &player) : Player(player) {}

void addNumberOfBlockShots()

{

this->numberOfBlockShots++;

}

void addNumberOfRepelledAttacks()

{

this->numberOfRepelledAttacks++;

}

void addNumberOfInterception()

{

this->numberOfInterception++;

}

void addNumberOfCounterattacks()

{

this->numberOfCounterattacks++;

}

void getInfoDefender()

{

cout << "Количество блоков: " << this->numberOfBlockShots << endl;

cout << "Количество отраженных ударов: " << this->numberOfRepelledAttacks << endl;

cout << "Количество перехватов мяча: " << this->numberOfInterception << endl;

cout << "Количество контударов: " << this->numberOfCounterattacks << endl;

}

};

class StatisticsTeam

{

private:

int win = 0;

int lose = 0;

int draw = 0;

public:

void addWin()

{

this->win++;

}

void addLose()

{

this->lose++;

}

void addDraw()

{

this->draw++;

}

void getWin()

{

cout << "Количество побед: " << this->win << endl;

}

void getLose()

{

cout << "Количество поражений: " << this->lose << endl;

}

void getDraw()

{

cout << "Количество ничей: " << this->draw << endl;

}

};

class TimeOut

{

private:

int countTimeOut = 0;

public:

void addTimeOut(const int &round)

{

if ((round == 1 || round == 2) && this->countTimeOut > 2)

{

cout << "Все тайм-ауты вышли в 1-вой половине игры." << endl;

return;

}

else if ((round == 3 || round == 4) && this->countTimeOut > 3)

{

cout << "Все тайм-ауты вышли в 2-й половине игры." << endl;

return;

}

this->countTimeOut++;

}

void getTimeOut() { cout << "Количество тайм-аутов: " << this->countTimeOut << endl; }

};

class Team : public StatisticsTeam, public TimeOut

{

private:

int teamNumber;

string coach;

vector<Striker> strikers;

vector<Defender> defenders;

vector<Player> players;

int theNumberOfPeopleInTheTeam = 0; // количество игроков в команде

int theNumberOfPlayersOnTheField = 0; // количество игроков на поле

int numberOfEliminatedPlayers = 0; // количество вышедших из игры

public:

Team(const string &coach, const int &teamNumber)

{

this->coach = coach;

this->teamNumber = teamNumber;

}

int getTeamNumber()

{

return teamNumber;

}

vector<Player> getPlayers()

{

return players;

}

void addPlayer(const Player &player)

{

this->players.push\_back(player);

this->theNumberOfPeopleInTheTeam++;

}

void addStriker(Striker &striker)

{

for (auto it = this->players.begin(); it != this->players.end(); it++)

{

if (it->getNamePlayer() == striker.getNamePlayer())

{

this->theNumberOfPlayersOnTheField++;

this->strikers.push\_back(striker);

return;

}

}

cout << "Такого нападающего нет в команде" << endl;

}

void addDefender(Defender &defender)

{

for (auto it = this->players.begin(); it != this->players.end(); it++)

{

if (it->getNamePlayer() == defender.getNamePlayer())

{

this->theNumberOfPlayersOnTheField++;

this->defenders.push\_back(defender);

return;

}

}

cout << "Такого защитника нет в команде" << endl;

}

bool removeStriker(const string &name)

{

for (auto it = this->strikers.begin(); it != this->strikers.end(); it++)

{

if (it->getNamePlayer() == name)

{

this->strikers.erase(it);

this->theNumberOfPlayersOnTheField--;

return true;

}

}

return false;

}

bool removeDefender(const string &name)

{

for (auto it = this->defenders.begin(); it != this->defenders.end(); it++)

{

if (it->getNamePlayer() == name)

{

this->defenders.erase(it);

this->theNumberOfPlayersOnTheField--;

return true;

}

}

return false;

}

void removePlayer(const string &name)

{

bool isFlag = false;

for (auto it = this->players.begin(); it != this->players.end(); it++)

{

if (it->getNamePlayer() == name)

{

if (removeDefender(it->getNamePlayer()))

{

removeDefender(it->getNamePlayer());

isFlag = true;

cout << "Был удален защитник: " << it->getNamePlayer() << endl;

}

if (removeStriker(it->getNamePlayer()))

{

isFlag = true;

removeStriker(it->getNamePlayer());

cout << "Был удален нападающий: " << it->getNamePlayer() << endl;

}

this->players.erase(it);

this->theNumberOfPeopleInTheTeam--;

if (isFlag)

{

this->theNumberOfPlayersOnTheField--;

}

else

{

cout << "Был удален игрок: " << it->getNamePlayer() << endl;

}

return;

}

}

cout << "Такого игрока нет в команде" << endl;

}

void getStrikers()

{

cout << "Нападающие: " << endl;

for (int i = 0; i < this->strikers.size(); i++)

{

strikers[i].getInfoPlayer();

}

cout << endl;

}

void getDefenders()

{

cout << "Защитники: " << endl;

for (int i = 0; i < this->defenders.size(); i++)

{

defenders[i].getInfoPlayer();

}

}

bool checkTeam()

{

bool isFlag = false;

if (this->theNumberOfPeopleInTheTeam < MAXIMUM\_NUMBER\_OF\_PLAYERS\_IN\_A\_TEAM)

{

cout << "У вас не хватает игроков: " << MAXIMUM\_NUMBER\_OF\_PLAYERS\_IN\_A\_TEAM - this->theNumberOfPeopleInTheTeam << endl;

isFlag = true;

}

else if (this->theNumberOfPeopleInTheTeam > MAXIMUM\_NUMBER\_OF\_PLAYERS\_IN\_A\_TEAM)

{

cout << "У вас больше игроков чем нужно на: " << this->theNumberOfPeopleInTheTeam - MAXIMUM\_NUMBER\_OF\_PLAYERS\_IN\_A\_TEAM << endl;

isFlag = true;

}

else

{

cout << "Количество игроков приемлемо." << endl;

}

if (this->theNumberOfPlayersOnTheField < THE\_MAXIMUM\_NUMBER\_OF\_PLAYERS\_ON\_THE\_FIELD)

{

cout << "У вас не хватает игроков на поле:" << THE\_MAXIMUM\_NUMBER\_OF\_PLAYERS\_ON\_THE\_FIELD - this->theNumberOfPlayersOnTheField << endl;

isFlag = true;

}

else if (this->theNumberOfPlayersOnTheField > THE\_MAXIMUM\_NUMBER\_OF\_PLAYERS\_ON\_THE\_FIELD)

{

cout << "У вас больше игроков на поле чем нужно на: " << this->theNumberOfPlayersOnTheField - THE\_MAXIMUM\_NUMBER\_OF\_PLAYERS\_ON\_THE\_FIELD << endl;

isFlag = true;

}

else

{

cout << "Количество игроков на поле приемлемо." << endl;

}

cout << endl;

return isFlag;

}

void getInfoTeam()

{

cout << "Тренер: " << this->coach << endl;

cout << "Количество игроков в команде:" << this->theNumberOfPeopleInTheTeam << endl;

cout << "Количество игроков на поле:" << this->theNumberOfPlayersOnTheField << endl;

cout << "Количество вышедших из игры: " << this->numberOfEliminatedPlayers << endl;

}

};

// Сингелтон или одиночка (порождающий паттерн)

class Tournament

{

private:

int numberOfRounds = 0;

static Tournament \*instance; // Указатель на единственный экземпляр

// Приватный конструктор, чтобы предотвратить создание объектов вне класса

Tournament() {}

// Приватный деструктор (не обязателен, но рекомендуется)

~Tournament() {}

public:

// Статический метод для получения единственного экземпляра

static Tournament \*getInstance()

{

if (instance == nullptr)

{

instance = new Tournament();

}

return instance;

}

void addRound()

{

if (this->numberOfRounds > MAXIMUM\_NUMBER\_OF\_ROUNDS)

{

cout << "Максимальное количество раундов: " << MAXIMUM\_NUMBER\_OF\_ROUNDS << endl;

return;

}

this->numberOfRounds++;

}

int getNumberOfRounds()

{

cout << "Раунд: " << this->numberOfRounds << endl;

return this->numberOfRounds;

}

void getNumberOfRemainingRounds()

{

cout << "Количество оставшихся раундов: " << MAXIMUM\_NUMBER\_OF\_ROUNDS - this->numberOfRounds << endl;

}

// Удаляем копирующий конструктор и оператор присваивания, чтобы предотвратить копирование

Tournament(const Tournament &) = delete;

Tournament &operator=(const Tournament &) = delete;

};

Tournament \*Tournament::instance = nullptr;

// Итератор (поведенческий паттерн)

class Score

{

private:

int numberOfPointsInTheFirstTeam = 0;

int numberOfPointsInTheSecondTeam = 0;

public:

void getScore() { cout << "Счёт: " << this->numberOfPointsInTheFirstTeam << " - " << this->numberOfPointsInTheSecondTeam << endl; }

int getNumberOfPointsInTheFirstTeam()

{

return numberOfPointsInTheFirstTeam;

}

int getNumberOfPointsInTheSecondTeam()

{

return numberOfPointsInTheSecondTeam;

}

void addPointsFirstTeam(const int &points)

{

this->numberOfPointsInTheFirstTeam += points;

}

void addPointsSecondTeam(const int &points)

{

this->numberOfPointsInTheSecondTeam += points;

}

void addHitPlayer(vector<Player> team, const int &points, const string &name)

{

for (vector<Player>::iterator it = team.begin(); it != team.end(); ++it)

{

if (it->getNamePlayer() == name)

{

it->addHitPlayer(); // увеличиваем количество попаданий в статистике игрока

cout << name << " получает " << points << " очков." << endl;

break;

}

}

}

};

class Break

{

public:

void getInfoBreak(const int &round)

{

if (round % 2 == 0)

{

cout << "Следующий перерыв будет равен: " << BREAK\_BETWEEN\_THE\_HALVES\_OF\_THE\_GAME << " минутам" << endl;

}

else

{

cout << "Следующий перерыв будет равен: " << BREAK\_BETWEEN\_QUARTERS << " минутам" << endl;

}

}

};

// паттерн фасад (структурный шаблон)

class Game

{

private:

Team team1;

Team team2;

Score score;

Tournament \*tournament;

Break breakInfo;

TimeOut timeOutInfo;

public:

Game(const Team &team1, const Team &team2) : team1(team1), team2(team2)

{

this->tournament = Tournament::getInstance();

}

bool checkTeams()

{

bool isFlag = true;

if (!this->team1.checkTeam())

{

cout << "В команде 1 присудствует ошибка при распределении игроков. Повторите попытку" << endl;

isFlag = false;

}

if (!this->team2.checkTeam())

{

cout << "В команде 2 присудствует ошибка при распределении игроков. Повторите попытку" << endl;

isFlag = false;

}

return isFlag;

}

// пример функции для паттерна фасад

void start()

{

if (!checkTeams())

{

return;

}

cout << "Игра началась" << endl;

this->tournament->addRound();

this->score.getScore();

this->breakInfo.getInfoBreak(this->tournament->getNumberOfRounds());

this->timeOutInfo.getTimeOut();

}

void scored(Team team, const int &points, const string &name)

{

this->score.addHitPlayer(team.getPlayers(), points, name); // статистика

if (team.getTeamNumber() == 1)

{

this->score.addPointsFirstTeam(points);

}

else

{

this->score.addPointsSecondTeam(points);

}

this->score.getScore();

}

void timeOut(Team &team)

{

team.addTimeOut(this->tournament->getNumberOfRounds());

}

void endOfTheHalf(Team &team, Team &team2)

{

this->score.getScore();

this->tournament->addRound();

this->breakInfo.getInfoBreak(this->tournament->getNumberOfRounds());

this->tournament->getNumberOfRemainingRounds();

team.getTimeOut();

team2.getTimeOut();

}

void stop()

{

this->score.getScore();

if (this->score.getNumberOfPointsInTheFirstTeam() > this->score.getNumberOfPointsInTheSecondTeam())

{

cout << "Победила команда под номером 1" << endl;

this->team1.addWin();

this->team2.addLose();

}

else if (this->score.getNumberOfPointsInTheFirstTeam() == this->score.getNumberOfPointsInTheSecondTeam())

{

cout << "Ничья" << endl;

this->team1.addDraw();

this->team2.addDraw();

}

else

{

cout << "Победила команда под номером 2" << endl;

this->team1.addLose();

this->team2.addWin();

}

cout << "Статистика по 1 команде" << endl;

this->team1.getWin();

this->team1.getLose();

this->team1.getDraw();

cout << "Статистика по 2 команде" << endl;

this->team2.getWin();

this->team2.getLose();

this->team2.getDraw();

cout << "Статистика по 1 команде" << endl;

for (int i = 0; i < this->team1.getPlayers().size(); i++)

{

cout << this->team1.getPlayers()[i].getNamePlayer() << endl;

this->team1.getPlayers()[i].getStatisticsAboutHits();

}

cout << "Статистика по 2 команде" << endl;

for (int i = 0; i < this->team2.getPlayers().size(); i++)

{

cout << this->team2.getPlayers()[i].getNamePlayer() << endl;

this->team2.getPlayers()[i].getStatisticsAboutHits();

}

}

};

int main(int argc, char const \*argv[])

{

SetConsoleOutputCP(CP\_UTF8);

// создаем игроков

Player user1 = Player("Михаил", 200, 195, 30, "Центровой", "C", 90);

Player user2 = Player("Антон", 198, 185, 25, "Форвард", "F", 80);

Player user3 = Player("Владимир", 205, 200, 28, "Центровой", "C", 85);

Player user4 = Player("Денис", 196, 190, 27, "Форвард", "F", 75);

Player user5 = Player("Иван", 202, 192, 29, "Форвард", "F", 78);

Player user6 = Player("Алексей", 198, 185, 25, "Центровой", "C", 88);

Player user7 = Player("Егор", 201, 197, 30, "Форвард", "F", 82);

Player user8 = Player("Максим", 199, 186, 26, "Защитник", "G", 79);

Player user9 = Player("Артем", 205, 198, 31, "Центровой", "C", 87);

Player user10 = Player("Дмитрий", 203, 195, 29, "Форвард", "F", 85);

Player user11 = Player("Николай", 200, 187, 27, "Защитник", "G", 78);

Player user12 = Player("Петр", 204, 196, 28, "Центровой", "C", 86);

Player user13 = Player("Сергей", 197, 182, 24, "Защитник", "G", 77);

Player user14 = Player("Андрей", 201, 194, 30, "Форвард", "F", 83);

Player user15 = Player("Григорий", 199, 186, 26, "Защитник", "G", 80);

Player user16 = Player("Олег", 202, 193, 29, "Форвард", "F", 81);

Player user17 = Player("Игорь", 198, 184, 25, "Защитник", "G", 79);

Player user18 = Player("Александр", 204, 197, 31, "Центровой", "C", 88);

Player user19 = Player("Семен", 196, 181, 24, "Защитник", "G", 76);

Player user20 = Player("Виктор", 200, 190, 28, "Центровой", "C", 84);

Player user21 = Player("Александр", 199, 186, 26, "Защитник", "G", 80);

Player user22 = Player("Александр", 202, 193, 29, "Форвард", "F", 81);

// распределяем по ролям игроков

Striker striker = Striker(user1);

Defender defender = Defender(user2);

Defender defender2 = Defender(user3);

Striker striker2 = Striker(user4);

Defender defender3 = Defender(user5);

Defender defender4 = Defender(user6);

Striker striker3 = Striker(user7);

Defender defender5 = Defender(user8);

Defender defender6 = Defender(user9);

Striker striker4 = Striker(user10);

Defender defender7 = Defender(user11);

Defender defender8 = Defender(user12);

Striker striker5 = Striker(user13);

Defender defender9 = Defender(user14);

Defender defender10 = Defender(user15);

Striker striker6 = Striker(user16);

Defender defender11 = Defender(user17);

Defender defender12 = Defender(user18);

Striker striker7 = Striker(user19);

Defender defender13 = Defender(user20);

Defender defender14 = Defender(user21);

Striker striker8 = Striker(user22);

// создаем команду

Team team = Team("Игорь", 1);

// добавляем игроков в команду

team.addPlayer(user1);

team.addPlayer(user2);

team.addPlayer(user3);

team.addPlayer(user4);

team.addPlayer(user5);

team.addPlayer(user6);

team.addPlayer(user7);

team.addPlayer(user8);

team.addPlayer(user9);

team.addPlayer(user10);

team.addPlayer(user11);

team.addPlayer(user12);

// добавляем защитников в 1 команду

team.addDefender(defender);

team.addDefender(defender2);

team.addDefender(defender3);

team.addDefender(defender4);

// добавляем нападающих в 1 команду

team.addStriker(striker);

team.addStriker(striker2);

// проверяем состояние 1 команды

team.getStrikers();

team.getDefenders();

team.checkTeam();

team.getInfoTeam();

// удаляем игрока из 1 команды, 1 с поля

team.removeDefender("Алексей");

// проверяем состояние 1 команды

team.getStrikers();

team.getDefenders();

team.checkTeam();

team.getInfoTeam();

// создаем 2 команду

Team team2 = Team("Сергей", 2);

// добавляем игроков в 2 команду

team2.addPlayer(user13);

team2.addPlayer(user14);

team2.addPlayer(user15);

team2.addPlayer(user16);

team2.addPlayer(user17);

team2.addPlayer(user18);

team2.addPlayer(user19);

team2.addPlayer(user20);

team2.addPlayer(user21);

team2.addPlayer(user22);

// добавляем защитников в 2 команду

team2.addDefender(defender7);

team2.addDefender(defender8);

team2.addDefender(defender9);

team2.addDefender(defender10);

// добавляем нападающих в 2 команду

team2.addStriker(striker4);

team2.addStriker(striker5);

team2.addStriker(striker6);

// проверяем состояние 2 команды

team2.checkTeam();

// создаем игру

Game game = Game(team, team2);

// пробуем запустить игру

game.start();

// исправляем ошибки

team2.addPlayer(user13);

team2.addPlayer(user14);

team2.addStriker(striker5);

game = Game(team, team2);

// запускаем игру без ошибок

game.start();

// моделируем игру

game.scored(team, 2, "Владимир");

defender2.addNumberOfBlockShots();

defender2.addNumberOfCounterattacks();

game.scored(team, 3, "Владимир");

defender2.addNumberOfRepelledAttacks();

striker5.addNumberOfSuccessfulAttacks();

game.scored(team2, 3, "Сергей");

striker5.addNumberOfRebounds();

game.scored(team2, 2, "Владимир");

game.scored(team, 3, "Владимир");

game.timeOut(team2);

game.scored(team2, 3, "Сергей");

// закончился 1 раунд

game.endOfTheHalf(team, team2);

// конец игры

game.stop();

return 0;

}

**Вывод программы:**

Нападающие:

Имя: Михаил

Вес: 200

Рост: 195

Скорость: 30

Точность удара: 90%

Имя: Денис

Вес: 196

Рост: 190

Скорость: 27

Точность удара: 75%

Защитники:

Имя: Антон

Вес: 198

Рост: 185

Скорость: 25

Точность удара: 80%

Имя: Владимир

Вес: 205

Рост: 200

Скорость: 28

Точность удара: 85%

Имя: Иван

Вес: 202

Рост: 192

Скорость: 29

Точность удара: 78%

Имя: Алексей

Вес: 198

Рост: 185

Скорость: 25

Точность удара: 88%

Количество игроков приемлемо.

У вас больше игроков на поле чем нужно на: 1

Тренер: Игорь

Количество игроков в команде:12

Количество игроков на поле:6

Количество вышедших из игры: 0

Нападающие:

Имя: Михаил

Вес: 200

Рост: 195

Скорость: 30

Точность удара: 90%

Имя: Денис

Вес: 196

Рост: 190

Скорость: 27

Точность удара: 75%

Защитники:

Имя: Антон

Вес: 198

Рост: 185

Скорость: 25

Точность удара: 80%

Имя: Владимир

Вес: 205

Рост: 200

Скорость: 28

Точность удара: 85%

Имя: Иван

Вес: 202

Рост: 192

Скорость: 29

Точность удара: 78%

Количество игроков приемлемо.

Количество игроков на поле приемлемо.

Тренер: Игорь

Количество игроков в команде:12

Количество игроков на поле:5

Количество вышедших из игры: 0

Такого защитника нет в команде

Такого защитника нет в команде

Такого нападающего нет в команде

У вас не хватает игроков: 2

У вас не хватает игроков на поле:1

Количество игроков приемлемо.

Количество игроков на поле приемлемо.

У вас не хватает игроков: 2

У вас не хватает игроков на поле:1

В команде 2 присудствует ошибка при распределении игроков. Повторите попытку

Количество игроков приемлемо.

Количество игроков на поле приемлемо.

Количество игроков приемлемо.

Количество игроков на поле приемлемо.

Игра началась

Счёт: 0 - 0

Раунд: 1

Следующий перерыв будет равен: 2 минутам

Количество тайм-аутов: 0

Владимир получает 2 очков.

Счёт: 2 - 0

Владимир получает 3 очков.

Счёт: 5 - 0

Сергей получает 3 очков.

Счёт: 5 - 3

Владимир получает 2 очков.

Счёт: 5 - 5

Владимир получает 3 очков.

Счёт: 8 - 5

Раунд: 1

Сергей получает 3 очков.

Счёт: 8 - 8

Счёт: 8 - 8

Раунд: 2

Следующий перерыв будет равен: 15 минутам

Количество оставшихся раундов: 2

Количество тайм-аутов: 0

Количество тайм-аутов: 1

Счёт: 8 - 8

Ничья

Статистика по 1 команде

Количество побед: 0

Количество поражений: 0

Количество ничей: 1

Статистика по 2 команде

Количество побед: 0

Количество поражений: 0

Количество ничей: 1

Статистика по 1 команде

Михаил

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Антон

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Владимир

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Денис

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Иван

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Алексей

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Егор

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Максим

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Артем

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Дмитрий

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Николай

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Петр

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Статистика по 2 команде

Михаил

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Антон

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Владимир

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Денис

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Иван

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Алексей

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Егор

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Максим

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Артем

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Дмитрий

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Николай

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

Петр

Количество попаданий: 0

Количество промахов: 0

Процент попаданий: 0

**Заключение**

В результате разработки программы моделирования спортивной игры в баскетбол удалось создать мощный инструмент для анализа и исследования тактических решений, стратегий команд и их влияния на результаты матчей. Программа предоставляет возможность эмулировать игровые ситуации, проводить анализ данных и прогнозировать исходы матчей на основе различных параметров.

**Список литературы**

1. https://learn.microsoft.com/ru-ru/cpp/cpp/?view=msvc-170