FootballEditor16

Создано системой Doxygen 1.8.11

Оглавление

1 Иерархический список классов 1

1.1 Иерархия классов 1

2 Алфавитный указатель классов 3

2.1 Классы 3

3 Классы 5

3.1 Класс Application 5

3.1.1 Подробное описание 6

3.2 Класс Competition 6

3.2.1 Подробное описание 7

3.2.2 Методы 7

3.2.2.1 addTeam(string name) 7

3.2.2.2 getGroupStage() 7

3.2.2.3 getNumberOfTeams() const 7

3.2.2.4 getPlayoffStage() 7

3.2.2.5 getTeams() 7

3.2.2.6 getTitle() 8

3.2.2.7 setTitle(string title) 8

3.2.2.8 startGroupStage(vector< int > teamIDs) 8

3.2.2.9 startPlayOffStage(vector< int > teamIDs) 8

3.3 Класс CompetitionIsOver 8

3.4 Класс CompetitionWindow 9

3.5 Класс Ui::CreateCompDialog 9

3.6 Класс CreateCompDialog 9
3.7 Класс Ui::Dialog 10
3.8 Класс Ui::EUROWindow 10
3.9 Класс EUROWindow 10
3.10 Класс Ui::ExitDialog 10
3.11 Класс ExitDialog 11
3.12 Класс Group 11
3.12.1 Подробное описание 12
3.12.2 Конструктор(ы) 12
3.12.2.1 Group() 12
3.12.3 Методы 12
3.12.3.1 addTeam(Team &team) 12
3.12.3.2 getId() const 12
3.12.3.3 getMatches() 12
3.12.3.4 getTeams() 13
3.12.3.5 getWinners() 13
3.12.4 Документация по друзьям класса и функциям, относящимся к классу 13
3.12.4.1 operator<< 13
3.13 Класс GroupsAreNotCreated 13
3.14 Класс GroupStage 14
3.14.1 Подробное описание 14
3.14.2 Конструктор(ы) 14
3.14.2.1 GroupStage(vector< Team > &teams) 14
3.14.3 Методы 15
3.14.3.1 getGroup(char id) 15
3.14.3.2 getGroups() 15
3.14.3.3 getNumberOfGroups() const 15
3.14.3.4 getWinners() 15
3.15 Класс GroupStageIsNotOver 16
3.16 Класс IdGenerator 16

3.16.1 Подробное описание 16
3.16.2 Методы 16
3.16.2.1 getGroupId() 16
3.16.2.2 getMatchId() 17
3.16.2.3 getTeamId() 17
3.16.2.4 IdGeneratorInstance() 17
3.17 Класс InputPenaltryScore 17
3.18 Класс Ui::MainWindow 18
3.19 Класс MainWindow 18
3.20 Класс Match 18
3.20.1 Подробное описание 19
3.20.2 Конструктор(ы) 19
3.20.2.1 Match(TeamStat &firstTeam, TeamStat &secondTeam) 19
3.20.3 Методы 20
3.20.3.1 getFirstTeam() 20
3.20.3.2 getId() const 20
3.20.3.3 getResult() 20
3.20.3.4 getSecondTeam() 20
3.20.3.5 isPlayed() const 20
3.20.3.6 setResult(int firstTeamGoals, int secondTeamGoals)=0 20
3.20.4 Документация по друзьям класса и функциям, относящимся к классу 21
3.20.4.1 operator<< 21
3.20.4.2 operator>> 21
3.21 Класс MatchInGroup 21
3.21.1 Подробное описание 22
3.21.2 Методы 22
3.21.2.1 setResult(int firstTeamGoals, int secondTeamGoals) 22
3.22 Класс MatchInPlayoff 22
3.22.1 Подробное описание 23
3.22.2 Методы 23

3.22.2.1 setResult(int firstTeamGoals, int secondTeamGoals) 23
3.23 Структура Penalty 23
3.24 Класс PlayoffStage 23
3.24.1 Подробное описание 24
3.25 Структура qt_meta_stringdata_CreateCompDialog_t 24
3.26 Структура qt_meta_stringdata_EUROWindow_t 24
3.27 Структура qt_meta_stringdata_ExitDialog_t 25
3.28 Структура qt_meta_stringdata_MainWindow_t 25
3.29 Структура qt_meta_stringdata_TeamCheckBoxes_t 25
3.30 Структура qt_meta_stringdata_WarningDialog_t 25
3.31 Класс Randomiser 26
3.31.1 Подробное описание 26
3.31.2 Методы 26
3.31.2.1 random(int begin, int end) 26
3.31.2.2 RandomiserInstance() 26
3.32 Класс Round 27
3.33 Класс Stage 27
3.33.1 Подробное описание 27
3.33.2 Методы 28
3.33.2.1 getTeams() 28
3.34 Класс Team 28
3.34.1 Подробное описание 28
3.34.2 Конструктор(ы) 28
3.34.2.1 Team(string name_=Team, string coach_=Coach) 28
3.34.2.2 Team(const Team &another) 29
3.34.3 Методы 29
3.34.3.1 getCoach() 29
3.34.3.2 getId() const 29
3.34.3.3 getName() 29
3.34.4 Документация по друзьям класса и функциям, относящимся к классу 29

3.34.4.1 operator<< . . . 29
3.35 Класс TeamCheckBoxes . . . 30
3.36 Класс TeamInGroup . . . 30
3.36.1 Подробное описание . . . 31
3.36.2 Конструктор(ы) . . . 31
3.36.2.1 TeamInGroup(Team &team) . . . 31
3.36.3 Методы . . . 32
3.36.3.1 getDraws() const . . . 32
3.36.3.2 getGames() const . . . 32
3.36.3.3 getLoses() const . . . 32
3.36.3.4 getPoints() const . . . 32
3.36.3.5 getWins() const . . . 32
3.36.4 Документация по друзьям класса и функциям, относящимся к классу . . . 32
3.36.4.1 operator< . . . 32
3.37 Класс TeamInPlayoff . . . 33
3.37.1 Подробное описание . . . 33
3.38 Класс TeamStat . . . 33
3.38.1 Подробное описание . . . 34
3.38.2 Конструктор(ы) . . . 34
3.38.2.1 TeamStat(Team &team) . . . 34
3.38.3 Методы . . . 35
3.38.3.1 decreaseGoals(int goalsFor_, int goalsAgainst_) . . . 35
3.38.3.2 getFouls() const . . . 35
3.38.3.3 getGoalsAgainst() const . . . 35
3.38.3.4 getGoalsDiffrence() const . . . 35
3.38.3.5 getGoalsFor() const . . . 36
3.38.3.6 getPossession() const . . . 36
3.38.3.7 getRedCards() const . . . 36
3.38.3.8 getShotOnTarget() const . . . 36
3.38.3.9 getShots() const . . . 36

3.38.3.10 getYellowCards() const 36

3.38.3.11 increaseGoals(int goalsFor_, int goalsAgainst_) 36

3.38.3.12 setFouls(int fouls_) 37

3.38.3.13 setGoals(int goalsFor_, int goalsAgainst_) 37

3.38.3.14 setPossession(int possession_) 37

3.38.3.15 setRedCards(int redCards_) 37

3.38.3.16 setShotOnTarget(int shotsOnTarget_) 38

3.38.3.17 setShots(int shots_) 38

3.38.3.18 setYellowCards(int yellowCards_) 38

3.39 Класс Ui_CreateCompDialog 38

3.40 Класс Ui_Dialog 39

3.41 Класс Ui_EUROWindow 40

3.42 Класс Ui_ExitDialog 44

3.43 Класс Ui_MainWindow 44

3.44 Класс Ui_WarningDialog 45

3.45 Класс WarningDialog 45

3.46 Класс Ui::WarningDialog 46

3.47 Класс WrongID 46

3.48 Класс WrongNumberOfTeams 46

3.49 Класс WrongPenaltyScore 47

Алфавитный указатель 49

Глава 1

Иерархический список классов

1.1 Иерархия классов

Иерархия классов.

Application 5
Competition 6
exception
  CompetitionIsOver 8
  GroupsAreNotCreated 13
  GroupStageIsNotOver 16
  InputPenaltryScore 17
  WrongID 46
  WrongNumberOfTeams 46
  WrongPenaltyScore 47
Group 11
IdGenerator 16
Match 18
  MatchInGroup 21
  MatchInPlayoff 22
Penalty 23
QDialog
  CreateCompDialog 9
  ExitDialog 11
  WarningDialog 45
QGroupBox
  TeamCheckBoxes 30
QMainWindow
  CompetitionWindow 9
  MainWindow 18
qt_meta_stringdata_CreateCompDialog_t 24
qt_meta_stringdata_EUROWindow_t 24
qt_meta_stringdata_ExitDialog_t 25
qt_meta_stringdata_MainWindow_t 25
qt_meta_stringdata_TeamCheckBoxes_t 25
qt_meta_stringdata_WarningDialog_t 25
QWidget
  EUROWindow 10
Randomiser 26
Round 27

Stage 27
GroupStage 14
PlayoffStage 23
Team 28
TeamStat 33
TeamInGroup 30
TeamInPlayoff 33
Ui_CreateCompDialog 38
Ui::CreateCompDialog 9
Ui_Dialog 39
Ui::Dialog 10
Ui_EUROWindow 40
Ui::EUROWindow 10
Ui_ExitDialog 44
Ui::ExitDialog 10
Ui_MainWindow 44
Ui::MainWindow 18
Ui_WarningDialog 45
Ui::WarningDialog 46

Глава 2

Алфавитный указатель классов

2.1 Классы

Классы с их кратким описанием.

Application
The Application class 5
Competition
The Competition class 6
CompetitionIsOver 8
CompetitionWindow 9
Ui::CreateCompDialog 9
CreateCompDialog 9
Ui::Dialog 10
Ui::EUROWindow 10
EUROWindow 10
Ui::ExitDialog 10
ExitDialog 11
Group
The Group class 11
GroupsAreNotCreated 13
GroupStage
The GroupStage class 14
GroupStageIsNotOver 16
IdGenerator
The IdGenerator class 16
InputPenaltryScore 17
Ui::MainWindow 18
MainWindow 18
Match
The Match class 18
MatchInGroup
The MatchInGroup class 21
MatchInPlayoff
The MatchInPlayoff class 22
Penalty 23
PlayoffStage
The PlayoffStage class 23
qt_meta_stringdata_CreateCompDialog_t 24
qt_meta_stringdata_EUROWindow_t 24

qt_meta_stringdata_ExitDialog_t 25
qt_meta_stringdata_MainWindow_t 25
qt_meta_stringdata_TeamCheckBoxes_t 25
qt_meta_stringdata_WarningDialog_t 25
Randomiser
The Randomiser class 26
Round 27
Stage
The Stage class 27
Team
The Team class 28
TeamCheckBoxes 30
TeamInGroup
The TeamInGroup class 30
TeamInPlayoff
The TeamInPlayoff class 33
TeamStat
The TeamStat class 33
Ui_CreateCompDialog 38
Ui_Dialog 39
Ui_EUROWindow 40
Ui_ExitDialog 44
Ui_MainWindow 44
Ui_WarningDialog 45
WarningDialog 45
Ui::WarningDialog 46
WrongID 46
WrongNumberOfTeams 46
WrongPenaltyScore 47

Глава 3

Классы

3.1 Класс Application

The Application class.

#include <application.h>

Открытые члены

- void loadCompetition ()
- void mainMenu ()
- void competitionMenu ()
- void launchCompetitionMenu ()
- void settiingsMenu ()
- void isCompetitionReadyToLaunch ()
- void setTitleOfCompetition ()
- void enterTeamsFromConsole ()
- void enterTeamsFromFile ()
- void showCurrentSettings ()
- void setDefaulSettings ()
- void showGroupsMenu ()
- void generateGroups ()
- void createGroups ()
- void showGroups ()
- void showGroupMatches ()
- void setResultsOfGroupMatches ()
- void simulateResultsOfGroupStage ()
- void showGroupMenu (char groupId)
- void showGroupTable (char groupId)
- void showGroupMatches (char groupId)
- void setResultsOfGroupMatches (char groupId)
- void simulateResultsOfGroupStage (char groupId)
- void showPlayoffMenu ()
- void createPlayoffPairs ()
- void showPlayoffTable ()
- void showPlayoffMatches ()
- void setResultsOfPlayoffMatches ()
- void simulateResultsOfPlayoffMatches ()
- void createNewRound ()

3.1.1 Подробное описание

The Application class.

Объявления и описания членов классов находятся в файлах:

- ConsoleApp/application.h
- ConsoleApp/application.cpp
- ConsoleApp/group.cpp
- ConsoleApp/groups.cpp
- ConsoleApp/playoff.cpp
- ConsoleApp/settings.cpp

3.2 Класс Competition

The Competition class.

#include <competition.h>

Открытые члены

- void setTitle (string title)

  Set the title of the competition.
- void addTeam (string name)

  Add the team to the current competition.
- string & getTitle ()

  Get title of the competition.
- unsigned getNumberOfTeams () const

  Get number of teams participating in the competition.
- vector< Team > & getTeams ()

  Get vector of teams participating in the competition.
- void saveCurrentCompetition ()

  Save the current competition.
- void loadCompetition ()

  Load the competition.
- void startGroupStage ()

  Launch group stage.
- void startGroupStage (vector< int > teamIDs)

  Launch group stage with custom groups.
- void createGroups ()

  Create groups.
- GroupStage & getGroupStage ()

  Get group stage.
- PlayoffStage & getPlayoffStage ()

  Get playoff stage.
- void startPlayOffStage ()

  Launch playoff stage.
- void startPlayOffStage (vector< int > teamIDs)

  Launch playoff stage with custom pairs.

3.2.1 Подробное описание

The Competition class.

3.2.2 Методы

3.2.2.1 void Competition::addTeam (string name)

Add the team to the current competition.

Аргументы

name	Name of the team

3.2.2.2 GroupStage & Competition::getGroupStage ( )

Get group stage.

Возвращает

group stage

3.2.2.3 unsigned Competition::getNumberOfTeams ( ) const

Get number of teams participating in the competition.

Возвращает

number of teams

3.2.2.4 PlayoffStage & Competition::getPlayoffStage ( )

Get playoff stage.

Возвращает

playoff stage

3.2.2.5 vector< Team > & Competition::getTeams ( )

Get vector of teams participating in the competition.

Возвращает

vector of teams

3.2.2.6 string & Competition::getTitle ( )

Get title of the competition.

Возвращает

title of the competition

3.2.2.7 void Competition::setTitle (string title)

Set the title of the competition.

Аргументы

title	Competition name

3.2.2.8 void Competition::startGroupStage (vector< int > teamIDs)

Launch group stage with custom groups.

Аргументы

teamIDs	

3.2.2.9 void Competition::startPlayOffStage (vector< int > teamIDs)

Launch playoff stage with custom pairs.

Аргументы

teamIDs	

Объявления и описания членов классов находятся в файлах:

- Core/competition.h
- Core/competition.cpp

3.3 Класс CompetitionIsOver

Граф наследования:CompetitionIsOver:

3.4 Класс CompetitionWindow

Граф наследования:CompetitionWindow:

Граф связей класса CompetitionWindow:

Открытые члены

- CompetitionWindow (QWidget *parent=0)
- void setSettings (QString title, QList< QListWidgetItem * > teams)

Объявления и описания членов классов находятся в файлах:

- GUIApp/competitionwindow.h
- GUIApp/competitionwindow.cpp

3.5 Класс Ui::CreateCompDialog

Граф наследования:Ui::CreateCompDialog:

Граф связей класса Ui::CreateCompDialog:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_createcompdialog.h

3.6 Класс CreateCompDialog

Граф наследования:CreateCompDialog:

Граф связей класса CreateCompDialog:

Открытые члены

- CreateCompDialog (QWidget *parent=0)

Объявления и описания членов классов находятся в файлах:

- GUIApp/createcompdialog.h
- GUIApp/createcompdialog.cpp

3.7 Класс Ui::Dialog

Граф наследования:Ui::Dialog:

Граф связей класса Ui::Dialog:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_temporary.h

3.8 Класс Ui::EUROWindow

Граф наследования:Ui::EUROWindow:

Граф связей класса Ui::EUROWindow:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_eurowindow.h

3.9 Класс EUROWindow

Граф наследования:EUROWindow:

Граф связей класса EUROWindow:

Открытые члены

- EUROWindow (QWidget *parent=0)

Объявления и описания членов классов находятся в файлах:

- GUIApp/eurowindow.h
- GUIApp/eurowindow.cpp

3.10 Класс Ui::ExitDialog

Граф наследования:Ui::ExitDialog:

Граф связей класса Ui::ExitDialog:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_exitdialog.h

3.11 Класс ExitDialog

Граф наследования:ExitDialog:

Граф связей класса ExitDialog:

Открытые члены

- ExitDialog (QWidget *parent=0)

Объявления и описания членов классов находятся в файлах:

- GUIApp/exitdialog.h
- GUIApp/exitdialog.cpp

3.12 Класс Group

The Group class.

#include <group.h>

Открытые члены

- Group ()

  Constructor of Group.
- void createMatches ()

  Create matches between members of groups.
- void addTeam (Team &team)

  Add a team to this group.
- char getId () const

  Get group 'name'.
- vector< TeamInGroup > & getTeams ()

  Get vector of teams.
- vector< MatchInGroup > & getMatches ()

  Get vector of created matches.
- vector< Team > getWinners ()

  Get winners of the stage.
- void sort ()

  Sort teams by number of points.
- vector< string > table ()

Друзья

- ostream & operator<< (ostream &os, Group &group)
  Overloading of operator <<.

3.12.1 Подробное описание

The Group class.

3.12.2 Конструктор(ы)

3.12.2.1 Group::Group ( )

Constructor of Group.

Аргументы

character	'Name' of the group (A, B, C,...)

3.12.3 Методы

3.12.3.1 void Group::addTeam (Team & team)

Add a team to this group.

Аргументы

team	Team

3.12.3.2 char Group::getId ( ) const

Get group 'name'.

Возвращает

group 'name' (A, B, C, etc.)

3.12.3.3 vector< MatchInGroup > & Group::getMatches ( )

Get vector of created matches.

Возвращает

vector of matches

3.12.3.4 vector< TeamInGroup > & Group::getTeams ( )

Get vector of teams.

Возвращает

vector of teams

3.12.3.5 vector< Team > Group::getWinners ( )

Get winners of the stage.

Возвращает

vector of winners (1 and 2 place of each group)

3.12.4 Документация по друзьям класса и функциям, относящимся к классу

3.12.4.1 ostream& operator<< (ostream & os, Group & group) [friend]

Overloading of operator <<.

Аргументы

os	Variable of type ostream
group	Variable of type Group

Возвращает

output stream

Объявления и описания членов классов находятся в файлах:

- Core/GroupStage/group.h
- Core/GroupStage/group.cpp

3.13 Класс GroupsAreNotCreated

Граф наследования:GroupsAreNotCreated:

Граф связей класса GroupsAreNotCreated:

Открытые члены

- const char * what ()

Объявления и описания членов класса находятся в файле:

- Core/GroupStage/Exceptions/groupsarenotcreated.h

3.14 Класс GroupStage

The GroupStage class.

#include <groupstage.h>

Граф наследования:GroupStage:

Граф связей класса GroupStage:

Открытые члены

- GroupStage (vector< Team > &teams)
  Constructor of GroupStage.
- void launch ()
  Create groups.
- vector< Group > & getGroups ()
  Get vector of groups.
- unsigned getNumberOfGroups () const
  Get number of groups.
- void createGroups ()
  Generate groups.
- void createGroups (vector< int > teamIDs)
  Create groups.
- vector< Team > & getWinners ()
  Get winners of the group stage.
- Group & getGroup (char id)
  Get group by ID (A, B, C, etc.)
- bool isGroupStageOver ()

Дополнительные унаследованные члены

3.14.1 Подробное описание

The GroupStage class.

3.14.2 Конструктор(ы)

3.14.2.1 GroupStage::GroupStage (vector< Team > & teams)

Constructor of GroupStage.

Аргументы

teams	Vector of the teams

3.14.3 Методы

3.14.3.1 Group & GroupStage::getGroup (char id)

Get group by ID (A, B, C, etc.)

Аргументы

id	

Возвращает

group

3.14.3.2 vector< Group > & GroupStage::getGroups ( )

Get vector of groups.

Возвращает

vector of groups

3.14.3.3 unsigned GroupStage::getNumberOfGroups ( ) const

Get number of groups.

Возвращает

number of groups

3.14.3.4 vector< Team > & GroupStage::getWinners ( )

Get winners of the group stage.

Возвращает

vector of winners

Объявления и описания членов классов находятся в файлах:

- Core/GroupStage/groupstage.h
- Core/GroupStage/groupstage.cpp

3.15 Класс GroupStageIsNotOver

Граф наследования:GroupStageIsNotOver:

Граф связей класса GroupStageIsNotOver:

Открытые члены

- const char * what ()

Объявления и описания членов класса находятся в файле:

- Core/Exceptions/groupstageisnotover.h

3.16 Класс IdGenerator

The IdGenerator class.

#include <idgenerator.h>

Открытые члены

- int getTeamId ()
  Get ID for team.
- int getMatchId ()
  Get ID for match.
- char getGroupId ()
  Get ID for group.

Открытые статические члены

- static IdGenerator * IdGeneratorInstance ()
  Instance ID Generator.

3.16.1 Подробное описание

The IdGenerator class.

3.16.2 Методы

3.16.2.1 char IdGenerator::getGroupId ( )

Get ID for group.

Возвращает

Generated ID

3.16.2.2 int IdGenerator::getMatchId ( )

Get ID for match.

Возвращает

Generated ID

3.16.2.3 int IdGenerator::getTeamId ( )

Get ID for team.

Возвращает

Generated ID

3.16.2.4 IdGenerator ∗ IdGenerator::IdGeneratorInstance ( ) [static]

Instance ID Generator.

Возвращает

pointer to ID Generator

Объявления и описания членов классов находятся в файлах:

- Core/idgenerator.h
- Core/idgenerator.cpp

3.17 Класс InputPenaltryScore

Граф наследования:InputPenaltryScore:

Граф связей класса InputPenaltryScore:

Открытые члены

- const char ∗ what ()

Объявления и описания членов класса находятся в файле:

- Core/PlayoffStage/Exceptions/inputpenaltryscore.h

3.18 Класс Ui::MainWindow

Граф наследования:Ui::MainWindow:

Граф связей класса Ui::MainWindow:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_mainmenu.h

3.19 Класс MainWindow

Граф наследования:MainWindow:

Граф связей класса MainWindow:

Открытые члены

- MainWindow (QWidget *parent=0)

Объявления и описания членов классов находятся в файлах:

- GUIApp/mainmenu.h
- GUIApp/mainmenu.cpp

3.20 Класс Match

The Match class.

#include <match.h>

Граф наследования:Match:

Граф связей класса Match:

Открытые члены

- Match (TeamStat &firstTeam, TeamStat &secondTeam)

  Constructor of Match.
- virtual ~Match ()

  Destructor.
- virtual void setResult (int firstTeamGoals, int secondTeamGoals)=0

  Set result of the match.
- string & getResult ()

  Get result of the match.
- int getId () const

  Get ID of the match.
- bool isPlayed () const

  Is game already played.
- TeamStat & getFirstTeam ()

  Get reference to the first team.
- TeamStat & getSecondTeam ()

  Get reference to the second team.
- virtual void simulate ()

  Simulate result of the match where number of goals of both teams are randomed from 0 to 4.

Защищенные члены

- virtual void updateResult ()

Защищенные данные

- TeamStat firstTeam
- TeamStat secondTeam
- string result

Друзья

- ostream & operator<< (ostream &os, Match &match)

  Overloading of operator <<.
- istream & operator>> (istream &is, Match &match)

  Overloading of operator >>

3.20.1 Подробное описание

The Match class.

3.20.2 Конструктор(ы)

3.20.2.1 Match::Match (TeamStat & firstTeam, TeamStat & secondTeam)

Constructor of Match.

Аргументы

firstTeam	First team
secondTeam	Second team

3.20.3 Методы

3.20.3.1 TeamStat & Match::getFirstTeam ( )

Get reference to the first team.

Возвращает

first team

3.20.3.2 int Match::getId ( ) const

Get ID of the match.

Возвращает

ID of the match

3.20.3.3 string & Match::getResult ( )

Get result of the match.

Возвращает

result of match in format "X:Y", where X - goals of first team, Y - goals of second team

3.20.3.4 TeamStat & Match::getSecondTeam ( )

Get reference to the second team.

Возвращает

second team

3.20.3.5 bool Match::isPlayed ( ) const

Is game already played.

Возвращает

true if game is played, false if game isn't played

3.20.3.6 virtual void Match::setResult (int firstTeamGoals, int secondTeamGoals) [pure virtual]

Set result of the match.

Аргументы

firstTeamGoals	Number of goals of first team
secondTeamGoals	Number of goals of second team

Замещается в MatchInPlayoff и MatchInGroup.

3.20.4 Документация по друзьям класса и функциям, относящимся к классу

3.20.4.1 ostream& operator<< (ostream & os, Match & match) [friend]

Overloading of operator <<.

Аргументы

os	Variable of type ostream
match	Variable of type Match

Возвращает

output stream Output in format: 'Team1 vs Team2' or 'Team1 X:Y Team2'

3.20.4.2 istream& operator>> (istream & is, Match & match) [friend]

Overloading of operator >>

Аргументы

is	Variable of type istream
match	Variable of type Match

Возвращает

input stream Input in format: 'X:Y' or 'X-Y'

Объявления и описания членов классов находятся в файлах:

- Core/match.h
- Core/match.cpp

3.21 Класс MatchInGroup

The MatchInGroup class.

#include <matchingroup.h>

Граф наследования:MatchInGroup:

Граф связей класса MatchInGroup:

Открытые члены

- MatchInGroup (TeamInGroup &firstTeam, TeamInGroup &secondTeam)
- void setResult (int firstTeamGoals, int secondTeamGoals)

  Set result of the match.
- void clear ()

Дополнительные унаследованные члены

3.21.1 Подробное описание

The MatchInGroup class.

3.21.2 Методы

3.21.2.1 void MatchInGroup::setResult (int firstTeamGoals, int secondTeamGoals) [virtual]

Set result of the match.

Аргументы

firstTeamGoals	Number of goals of first team
secondTeamGoals	Number of goals of second team

Замещает Match.

Объявления и описания членов классов находятся в файлах:

- Core/GroupStage/matchingroup.h
- Core/GroupStage/matchingroup.cpp

3.22 Класс MatchInPlayoff

The MatchInPlayoff class.

#include <matchinplayoff.h>

Граф наследования:MatchInPlayoff:

Граф связей класса MatchInPlayoff:

Открытые члены

- MatchInPlayoff (TeamInPlayoff &firstTeam, TeamInPlayoff &secondTeam)
- void setResult (int firstTeamGoals, int secondTeamGoals)

  Set result of the match.
- void setPenaltyScore (int firstTeamPenaltyScore, int secondTeamPenaltyScore)
- void simulate ()

  Simulate result of the match where number of goals of both teams are randomed from 0 to 4.
- TeamInPlayoff & getWinner ()
- string showFirstTeam ()
- string showSecondTeam ()

Дополнительные унаследованные члены

3.22.1 Подробное описание

The MatchInPlayoff class.

3.22.2 Методы

3.22.2.1 void MatchInPlayoff::setResult (int firstTeamGoals, int secondTeamGoals) [virtual]

Set result of the match.

Аргументы

firstTeamGoals	Number of goals of first team
secondTeamGoals	Number of goals of second team

Замещает Match.

Объявления и описания членов классов находятся в файлах:

- Core/PlayoffStage/matchinplayoff.h
- Core/PlayoffStage/matchinplayoff.cpp

3.23 Структура Penalty

Открытые атрибуты

- int firstTeamPenaltyScore = 0
- int secondTeamPenaltyScore = 0

Объявления и описания членов структуры находятся в файле:

- Core/PlayoffStage/matchinplayoff.h

3.24 Класс PlayoffStage

The PlayoffStage class.

#include <playoffstage.h>

Граф наследования:PlayoffStage:

Граф связей класса PlayoffStage:

Открытые члены

- PlayoffStage (vector< Team > &teams)
- void launch ()

  Launch this stage.
- void createMatches (vector< int > teamIDs)
- void createNewMatches ()
- int getNumberOfRounds () const
- vector< MatchInPlayoff > & getMatches ()
- RoundEnum getCurrentRound () const
- vector< Round > & getRounds ()

Друзья

- ostream & operator<< (ostream &os, PlayoffStage &playoff)

Дополнительные унаследованные члены

3.24.1 Подробное описание

The PlayoffStage class.

Объявления и описания членов классов находятся в файлах:

- Core/PlayoffStage/playoffstage.h
- Core/PlayoffStage/playoffstage.cpp

3.25 Структура qt_meta_stringdata_CreateCompDialog_t

Открытые атрибуты

- QByteArrayData data [11]
- char stringdata0 [172]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_createcompdialog.cpp

3.26 Структура qt_meta_stringdata_EUROWindow_t

Открытые атрибуты

- QByteArrayData data [1]
- char stringdata0 [11]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_eurowindow.cpp

3.27 Структура qt_meta_stringdata_ExitDialog_t

Открытые атрибуты

- QByteArrayData data [1]
- char stringdata0 [11]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_exitdialog.cpp

3.28 Структура qt_meta_stringdata_MainWindow_t

Открытые атрибуты

- QByteArrayData data [4]
- char stringdata0 [53]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_mainmenu.cpp

3.29 Структура qt_meta_stringdata_TeamCheckBoxes_t

Открытые атрибуты

- QByteArrayData data [3]
- char stringdata0 [25]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_teamcheckboxes.cpp

3.30 Структура qt_meta_stringdata_WarningDialog_t

Открытые атрибуты

- QByteArrayData data [1]
- char stringdata0 [14]

Объявления и описания членов структуры находятся в файле:

- GUIApp/moc_warningdialog.cpp

3.31 Класс Randomiser

The Randomiser class.

#include <randomiser.h>

Открытые члены

- int random (int begin, int end)
  Get random value from begin to end.

Открытые статические члены

- static Randomiser * RandomiserInstance ()
  Instance randomiser.

3.31.1 Подробное описание

The Randomiser class.

3.31.2 Методы

3.31.2.1 int Randomiser::random (int begin, int end)

Get random value from begin to end.

Аргументы

begin	Bottom bound
end	Upper bound

Возвращает

randomed value

3.31.2.2 Randomiser * Randomiser::RandomiserInstance ( ) [static]

Instance randomiser.

Возвращает

pointer to randomiser

Объявления и описания членов классов находятся в файлах:

- Core/randomiser.h
- Core/randomiser.cpp

3.32 Класс Round

Открытые члены

- Round (RoundEnum round)
- void addPair (TeamInPlayoff &first, TeamInPlayoff &second)
- RoundEnum getRoundEnum () const
- void addTeams (vector< TeamInPlayoff > &teams_)
- vector< TeamInPlayoff > & getTeams ()
- vector< MatchInPlayoff > & getMatches ()
- vector< TeamInPlayoff > & getWinners ()
- void updateWinners ()

Объявления и описания членов классов находятся в файлах:

- Core/PlayoffStage/round.h
- Core/PlayoffStage/round.cpp

3.33 Класс Stage

The Stage class.

#include <stage.h>

Граф наследования:Stage:

Граф связей класса Stage:

Открытые члены

- Stage (vector< Team > &teams)

  Launch this stage.
- vector< Team > & getTeams ()

  Get vector of teams participating in the competition.
- virtual void launch ()=0

  Launch the stage.

Защищенные данные

- vector< Team > & teams

3.33.1 Подробное описание

The Stage class.

3.33.2 Методы

3.33.2.1 vector<Team>& Stage::getTeams ( ) [inline]

Get vector of teams participating in the competition.

Возвращает

vector of teams

Объявления и описания членов класса находятся в файле:

- Core/stage.h

3.34 Класс Team

The Team class.

#include <team.h>

Граф наследования:Team:

Открытые члены

- Team (string name_="Team", string coach_="Coach")
  Constructor of Team.
- Team (const Team &another)
  Copy constructor of Team.
- int getId () const
  Get team ID.
- string & getName ()
  Get name of the team.
- string & getCoach ()
  Get name of the coach.

Друзья

- ostream & operator<< (ostream &os, const Team &team)
  Overloading of operator <<.
- bool operator== (const Team &one, const Team &another)

3.34.1 Подробное описание

The Team class.

3.34.2 Конструктор(ы)

3.34.2.1 Team::Team (string name_ = "Team", string coach_ = "Coach")

Constructor of Team.

Аргументы

name	Team name with deafaul value "Team"
coach	Team coach with default value "Coach"

3.34.2.2 Team::Team (const Team & another)

Copy constructor of Team.

Аргументы

other	Variable of type Team

3.34.3 Методы

3.34.3.1 string & Team::getCoach ( )

Get name of the coach.

Возвращает

Name of the coach

3.34.3.2 int Team::getId ( ) const

Get team ID.

Возвращает

team ID

3.34.3.3 string & Team::getName ( )

Get name of the team.

Возвращает

name of the team

3.34.4 Документация по друзьям класса и функциям, относящимся к классу

3.34.4.1 ostream& operator<< (ostream & os, const Team & team) [friend]

Overloading of operator <<.

Аргументы

os	Variable of type ostream
team	Variable of type Team

Возвращает

output stream

Объявления и описания членов классов находятся в файлах:

- Core/team.h
- Core/team.cpp

3.35 Класс TeamCheckBoxes

Граф наследования:TeamCheckBoxes:

Граф связей класса TeamCheckBoxes:

Открытые слоты

- void checkMax ()

Открытые члены

- TeamCheckBoxes (QWidget *parent=0)
- void setMode (bool)
- QList< QCheckBox * > selectedItems () const
- QCheckBox * item (int id) const

Объявления и описания членов классов находятся в файлах:

- GUIApp/teamcheckboxes.h
- GUIApp/teamcheckboxes.cpp

3.36 Класс TeamInGroup

The TeamInGroup class.

#include <teamingroup.h>

Граф наследования:TeamInGroup:

Граф связей класса TeamInGroup:

Открытые члены

- TeamInGroup (Team &team)

  Constructor of TeamInGroup.
- void win ()

  Increase games and wins by 1 & increase points by 3.
- void draw ()

  Increase games and draws by 1 & increase points by 1.
- void lose ()

  Increase games and loses by 1.
- void unWin ()

  Decrease games and wins by 1 & decrease points by 3.
- void unDraw ()

  Decrease games and draws by 1 & decrease points by 1.
- void unLose ()

  Decrease games and loses by 1.
- int getPoints () const

  Get number of points.
- int getGames () const

  Get number of games.
- int getWins () const

  Get number of wins.
- int getDraws () const

  Get number of draws.
- int getLoses () const

  Get number of loses.

Друзья

- bool operator< (const TeamInGroup &one, const TeamInGroup &another)

  Overloading of operator <.

3.36.1 Подробное описание

The TeamInGroup class.

3.36.2 Конструктор(ы)

3.36.2.1 TeamInGroup::TeamInGroup (Team & team)

Constructor of TeamInGroup.

Аргументы

team	Object of type Team

3.36.3 Методы

3.36.3.1 int TeamInGroup::getDraws ( ) const

Get number of draws.

Возвращает

draws

3.36.3.2 int TeamInGroup::getGames ( ) const

Get number of games.

Возвращает

games

3.36.3.3 int TeamInGroup::getLoses ( ) const

Get number of loses.

Возвращает

loses

3.36.3.4 int TeamInGroup::getPoints ( ) const

Get number of points.

Возвращает

points

3.36.3.5 int TeamInGroup::getWins ( ) const

Get number of wins.

Возвращает

wins

3.36.4 Документация по друзьям класса и функциям, относящимся к классу

3.36.4.1 bool operator< (const TeamInGroup & one, const TeamInGroup & another) [friend]

Overloading of operator <.

Аргументы

one	Variable of type TeamInGroup
another	Variable of type TeamInGroup

Возвращает

true, if points of one is less than points of second false, if points of one is greater than or equal of points of second

Объявления и описания членов классов находятся в файлах:

- Core/GroupStage/teamingroup.h
- Core/GroupStage/teamingroup.cpp

3.37 Класс TeamInPlayoff

The TeamInPlayoff class.

#include <teaminplayoff.h>

Граф наследования:TeamInPlayoff:

Граф связей класса TeamInPlayoff:

Открытые члены

- TeamInPlayoff (Team &team)

3.37.1 Подробное описание

The TeamInPlayoff class.

Объявления и описания членов классов находятся в файлах:

- Core/PlayoffStage/teaminplayoff.h
- Core/PlayoffStage/teaminplayoff.cpp

3.38 Класс TeamStat

The TeamStat class.

#include <teamstat.h>

Граф наследования:TeamStat:

Граф связей класса TeamStat:

Открытые члены

- TeamStat (Team &team)

  TeamStat.
- void increaseGoals (int goalsFor_, int goalsAgainst_)

  Increase goals.
- void decreaseGoals (int goalsFor_, int goalsAgainst_)

  Decrease goals.
- void setGoals (int goalsFor_, int goalsAgainst_)

  Set the number of goals.
- void setPossession (int possession_)

  Set possession.
- void setShots (int shots_)

  Set the number of shots.
- void setShotOnTarget (int shotsOnTarget_)

  Set the number of shots on target.
- void setYellowCards (int yellowCards_)

  Set the number of yellow cards.
- void setRedCards (int redCards_)

  Set the number of red cards.
- void setFouls (int fouls_)

  Set the number of fouls.
- int getGoalsFor () const

  Get the number of goals.
- int getGoalsAgainst () const

  Get goals against.
- int getGoalsDiffrence () const

  Get goals diffrence (goals for minus goals against)
- int getPossession () const

  Get possession.
- int getShots () const

  Get number of shots.
- int getShotOnTarget () const

  Get number of shots on target.
- int getYellowCards () const

  Get number of yellow cards.
- int getRedCards () const

  Get number of red cards.
- int getFouls () const

  Get number of fouls.
- void clear ()

3.38.1 Подробное описание

The TeamStat class.

3.38.2 Конструктор(ы)

3.38.2.1 TeamStat::TeamStat (Team & team)

TeamStat.

Аргументы

team	

3.38.3 Методы

3.38.3.1 void TeamStat::decreaseGoals (int goalsFor_, int goalsAgainst_)

Decrease goals.

Аргументы

goalsFor_	Goals scored team
goals↩ Against_	Goals scored rivals

3.38.3.2 int TeamStat::getFouls ( ) const

Get number of fouls.

Возвращает

Number of fouls

3.38.3.3 int TeamStat::getGoalsAgainst ( ) const

Get goals against.

Возвращает

goals aginst

3.38.3.4 int TeamStat::getGoalsDiffrence ( ) const

Get goals diffrence (goals for minus goals against)

Возвращает

goals diffrence

3.38.3.5 int TeamStat::getGoalsFor ( ) const

Get the number of goals.

Возвращает

goals for

3.38.3.6 int TeamStat::getPossession ( ) const

Get possession.

Возвращает

Possession

3.38.3.7 int TeamStat::getRedCards ( ) const

Get number of red cards.

Возвращает

Number of red cards

3.38.3.8 int TeamStat::getShotOnTarget ( ) const

Get number of shots on target.

Возвращает

Number of shots of target

3.38.3.9 int TeamStat::getShots ( ) const

Get number of shots.

Возвращает

Number of shots

3.38.3.10 int TeamStat::getYellowCards ( ) const

Get number of yellow cards.

Возвращает

Number of yellow cards

3.38.3.11 void TeamStat::increaseGoals (int goalsFor_, int goalsAgainst_)

Increase goals.

Аргументы

goalsFor_	Goals scored team
goals↩ Against_	Goals scored rivals

3.38.3.12 void TeamStat::setFouls (int fouls_)

Set the number of fouls.

Аргументы

fouls↩ _	Number of fouls

3.38.3.13 void TeamStat::setGoals (int goalsFor_, int goalsAgainst_)

Set the number of goals.

Аргументы

goals↩ _	Number of goals

3.38.3.14 void TeamStat::setPossession (int possession_)

Set possession.

Аргументы

possession↩ _	Possession

3.38.3.15 void TeamStat::setRedCards (int redCards_)

Set the number of red cards.

Аргументы

red↩ Cards_	Number of red cards

3.38.3.16 void TeamStat::setShotOnTarget (int shotsOnTarget_)

Set the number of shots on target.

Аргументы

shotsOn↩ Target_	Number of shots on target

3.38.3.17 void TeamStat::setShots (int shots_)

Set the number of shots.

Аргументы

shots↩ _	Number of shots

3.38.3.18 void TeamStat::setYellowCards (int yellowCards_)

Set the number of yellow cards.

Аргументы

yellow↩ Cards_	Number of yellow cards

Объявления и описания членов классов находятся в файлах:

- Core/teamstat.h
- Core/teamstat.cpp

3.39 Класс Ui_CreateCompDialog

Граф наследования:Ui_CreateCompDialog:

Открытые члены

- void setupUi (QDialog *CreateCompDialog)
- void retranslateUi (QDialog *CreateCompDialog)

Открытые атрибуты

- QWidget * layoutWidget
- QHBoxLayout * lauoutButtons
- QPushButton * btnMenu
- QSpacerItem * horizontalSpacer
- QPushButton * btnCreate
- QFrame * frame
- QLabel * lblCreate
- QLabel * lblPreferences
- QLabel * lblNumber
- QGroupBox * grbNumberOfTeams
- QRadioButton * rbtn32
- QRadioButton * rbtn16
- QLabel * lblTeams
- QLineEdit * edtTitle
- QLabel * lblTitle
- QComboBox * cmbPreferences
- QListWidget * lstTeams
- QProgressBar * prbTeams

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_createcompdialog.h

3.40 Класс Ui_Dialog

Граф наследования:Ui_Dialog:

Открытые члены

- void setupUi (QDialog *Dialog)
- void retranslateUi (QDialog *Dialog)

Открытые атрибуты

- QTableWidget * tableWidget
- QTabWidget * tabGroups
- QWidget * tabGroupA
- QTabWidget * tabGroups_2
- QWidget * tabGroupA_2
- QLabel * lblTable16_A_1
- QLabel * lblTable16_A_3
- QLabel * lblTable16_A_5
- QLabel * lblTable16_A_7
- QLabel * lblTable16_A_9
- QLabel * lblTable16_A_6
- QLabel * lblTable16_A_10
- QLabel * lblTable16_A_8
- QLabel * lblTable16_A_4

- QLabel * lblTable16_A_2
- QLabel * lblTable16_A_11
- QLabel * lblTable16_A_12
- QLabel * lblTable16_A_13
- QLabel * lblTable16_A_14
- QLabel * lblTable16_A_15
- QWidget * tab_2
- QWidget * tab
- QWidget * tab_3
- QWidget * tab_4

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_temporary.h

3.41 Класс Ui_EUROWindow

Граф наследования:Ui_EUROWindow:

Граф связей класса Ui_EUROWindow:

Открытые члены

- void setupUi (QWidget *EUROWindow)
- void retranslateUi (QWidget *EUROWindow)

Открытые атрибуты

- QStackedWidget * stackedWidget
- QWidget * pgCreateGroups4
- QFrame * frame
- QLabel * lblCreate
- TeamCheckBoxes * grbTeams
- QCheckBox * checkBox_9
- QCheckBox * checkBox_10
- QCheckBox * checkBox_12
- QCheckBox * checkBox_11
- QCheckBox * checkBox_13
- QCheckBox * checkBox_14
- QCheckBox * checkBox_15
- QCheckBox * checkBox_16
- QCheckBox * checkBox_8
- QCheckBox * checkBox_5
- QCheckBox * checkBox_6
- QCheckBox * checkBox_1
- QCheckBox * checkBox_3
- QCheckBox * checkBox_4
- QCheckBox * checkBox_7
- QCheckBox * checkBox_2
- QPushButton * btn_CG4_Start

- QPushButton * btn_CG4_Create
- QPushButton * btn_CG4_Generate
- QWidget * pgGroupStage16
- QFrame * frame_GS16
- QLabel * lbl_GS16_GroupStage
- QTabWidget * tab_GS16_Groups
- QWidget * wid_GS16_GroupA
- QTabWidget * tab_GS16_GroupA
- QWidget * tab_GS16_GroupA_Table
- QTreeWidget * tree_GS16_GroupA
- QLabel * lbl_GS16_GroupA_1st
- QLabel * lbl_GS16_GroupA_2nd
- QLabel * lbl_GS16_GroupA_3rd
- QLabel * lbl_GS16_GroupA_4th
- QLabel * lbl_GS16_GroupA_legend1
- QLabel * lbl_GS16_GroupA_legend2
- QWidget * tab_GS16_GroupA_Matches
- QSpinBox * spb_GS16_GroupA_right_m2
- QLabel * lbl_GS16_GroupA_right_m2
- QLabel * lbl_GS16_GroupA_left_m2
- QSpinBox * spb_GS16_GroupA_left_m2
- QLabel * lbl_GS16_GroupA_sep2
- QSpinBox * spb_GS16_GroupA_left_m1
- QLabel * lbl_GS16_GroupA_right_m1
- QLabel * lbl_GS16_GroupA_sep1
- QLabel * lbl_GS16_GroupA_left_m1
- QSpinBox * spb_GS16_GroupA_right_m1
- QSpinBox * spb_GS16_GroupA_left_m4
- QLabel * lbl_GS16_GroupA_left_m3
- QLabel * lbl_GS16_GroupA_right_m4
- QSpinBox * spb_GS16_GroupA_right_m3
- QLabel * lbl_GS16_GroupA_sep4
- QLabel * lbl_GS16_GroupA_right_m3
- QLabel * lbl_GS16_GroupA_left_m4
- QLabel * lbl_GS16_GroupA_sep3
- QSpinBox * spb_GS16_GroupA_left_m3
- QSpinBox * spb_GS16_GroupA_right_m4
- QSpinBox * spb_GS16_GroupA_left_m6
- QLabel * lbl_GS16_GroupA_left_m5
- QLabel * lbl_GS16_GroupA_right_m6
- QSpinBox * spb_GS16_GroupA_right_m5
- QLabel * lbl_GS16_GroupA_sep6
- QLabel * lbl_GS16_GroupA_right_m5
- QLabel * lbl_GS16_GroupA_left_m6
- QLabel * lbl_GS16_GroupA_sep5
- QSpinBox * spb_GS16_GroupA_left_m5
- QSpinBox * spb_GS16_GroupA_right_m6
- QPushButton * btn_GS16_GroupA_Save
- QPushButton * btn_GS16_GroupA_Simulate
- QWidget * wid_GS16_GroupB
- QTabWidget * tab_GS16_GroupB
- QWidget * tab_GS16_GroupB_Table
- QTreeWidget * tree_GS16_GroupB
- QLabel * lbl_GS16_GroupB_1st
- QLabel * lbl_GS16_GroupB_2nd

- QLabel * lbl_GS16_GroupB_3rd
- QLabel * lbl_GS16_GroupB_4th
- QLabel * lbl_GS16_GroupB_legend1
- QLabel * lbl_GS16_GroupB_legend2
- QWidget * tab_GS16_GroupB_Matches
- QSpinBox * spb_GS16_GroupB_right_m6
- QLabel * lbl_GS16_GroupB_left_m1
- QSpinBox * spb_GS16_GroupB_left_m5
- QLabel * lbl_GS16_GroupB_right_m1
- QSpinBox * spb_GS16_GroupB_left_m6
- QSpinBox * spb_GS16_GroupB_right_m4
- QLabel * lbl_GS16_GroupB_left_m3
- QSpinBox * spb_GS16_GroupB_left_m2
- QLabel * lbl_GS16_GroupB_sep6
- QSpinBox * spb_GS16_GroupB_right_m1
- QLabel * lbl_GS16_GroupB_left_m6
- QLabel * lbl_GS16_GroupB_sep3
- QLabel * lbl_GS16_GroupB_sep4
- QLabel * lbl_GS16_GroupB_left_m4
- QSpinBox * spb_GS16_GroupB_right_m5
- QLabel * lbl_GS16_GroupB_right_m4
- QSpinBox * spb_GS16_GroupB_right_m2
- QLabel * lbl_GS16_GroupB_right_m5
- QLabel * lbl_GS16_GroupB_right_m6
- QLabel * lbl_GS16_GroupB_sep2
- QLabel * lbl_GS16_GroupB_left_m2
- QSpinBox * spb_GS16_GroupB_left_m4
- QLabel * lbl_GS16_GroupB_left_m5
- QLabel * lbl_GS16_GroupB_sep5
- QSpinBox * spb_GS16_GroupB_right_m3
- QLabel * lbl_GS16_GroupB_right_m2
- QLabel * lbl_GS16_GroupB_right_m3
- QSpinBox * spb_GS16_GroupB_left_m3
- QSpinBox * spb_GS16_GroupB_left_m1
- QLabel * lbl_GS16_GroupB_sep1
- QPushButton * btn_GS16_GroupB_Simulate
- QPushButton * btn_GS16_GroupB_Save
- QWidget * wid_GS16_GroupC
- QTabWidget * tab_GS16_GroupC
- QWidget * tab_GS16_GroupC_Table
- QTreeWidget * tree_GS16_GroupC
- QLabel * lbl_GS16_GroupC_1st
- QLabel * lbl_GS16_GroupC_2nd
- QLabel * lbl_GS16_GroupC_3rd
- QLabel * lbl_GS16_GroupC_4th
- QLabel * lbl_GS16_GroupC_legend1
- QLabel * lbl_GS16_GroupC_legend2
- QWidget * tab_GS16_GroupC_Matches
- QSpinBox * spb_GS16_GroupC_right_m6
- QLabel * lbl_GS16_GroupC_left_m1
- QSpinBox * spb_GS16_GroupC_left_m5
- QLabel * lbl_GS16_GroupC_right_m1
- QSpinBox * spb_GS16_GroupC_left_m6
- QSpinBox * spb_GS16_GroupC_right_m4
- QLabel * lbl_GS16_GroupC_left_m3

- QSpinBox * spb_GS16_GroupC_left_m2
- QLabel * lbl_GS16_GroupC_sep6
- QSpinBox * spb_GS16_GroupC_right_m1
- QLabel * lbl_GS16_GroupC_left_m6
- QLabel * lbl_GS16_GroupC_sep3
- QLabel * lbl_GS16_GroupC_sep4
- QLabel * lbl_GS16_GroupC_left_m4
- QSpinBox * spb_GS16_GroupC_right_m5
- QLabel * lbl_GS16_GroupC_right_m4
- QSpinBox * spb_GS16_GroupC_right_m2
- QLabel * lbl_GS16_GroupC_right_m5
- QLabel * lbl_GS16_GroupC_right_m6
- QLabel * lbl_GS16_GroupC_sep2
- QLabel * lbl_GS16_GroupC_left_m2
- QSpinBox * spb_GS16_GroupC_left_m4
- QLabel * lbl_GS16_GroupC_left_m5
- QLabel * lbl_GS16_GroupC_sep5
- QSpinBox * spb_GS16_GroupC_right_m3
- QLabel * lbl_GS16_GroupC_right_m2
- QLabel * lbl_GS16_GroupC_right_m3
- QSpinBox * spb_GS16_GroupC_left_m3
- QSpinBox * spb_GS16_GroupC_left_m1
- QLabel * lbl_GS16_GroupC_sep1
- QPushButton * btn_GS16_GroupC_Simulate
- QPushButton * btn_GS16_GroupC_Save
- QWidget * wid_GS16_GroupD
- QTabWidget * tab_GS16_GroupD
- QWidget * tab_GS16_GroupD_Table
- QTreeWidget * tree_GS16_GroupD
- QLabel * lbl_GS16_GroupD_1st
- QLabel * lbl_GS16_GroupD_2nd
- QLabel * lbl_GS16_GroupD_3rd
- QLabel * lbl_GS16_GroupD_4th
- QLabel * lbl_GS16_GroupD_legend1
- QLabel * lbl_GS16_GroupD_legend2
- QWidget * tab_GS16_GroupD_Matches
- QSpinBox * spb_GS16_GroupD_right_m6
- QLabel * lbl_GS16_GroupD_left_m1
- QSpinBox * spb_GS16_GroupD_left_m5
- QLabel * lbl_GS16_GroupD_right_m1
- QSpinBox * spb_GS16_GroupD_left_m6
- QSpinBox * spb_GS16_GroupD_right_m4
- QLabel * lbl_GS16_GroupD_left_m3
- QSpinBox * spb_GS16_GroupD_left_m2
- QLabel * lbl_GS16_GroupD_sep6
- QSpinBox * spb_GS16_GroupD_right_m1
- QLabel * lbl_GS16_GroupD_left_m6
- QLabel * lbl_GS16_GroupD_sep3
- QLabel * lbl_GS16_GroupD_sep4
- QLabel * lbl_GS16_GroupD_left_m4
- QSpinBox * spb_GS16_GroupD_right_m5
- QLabel * lbl_GS16_GroupD_right_m4
- QSpinBox * spb_GS16_GroupD_right_m2
- QLabel * lbl_GS16_GroupD_right_m5
- QLabel * lbl_GS16_GroupD_right_m6

- QLabel * lbl_GS16_GroupD_sep2
- QLabel * lbl_GS16_GroupD_left_m2
- QSpinBox * spb_GS16_GroupD_left_m4
- QLabel * lbl_GS16_GroupD_left_m5
- QLabel * lbl_GS16_GroupD_sep5
- QSpinBox * spb_GS16_GroupD_right_m3
- QLabel * lbl_GS16_GroupD_right_m2
- QLabel * lbl_GS16_GroupD_right_m3
- QSpinBox * spb_GS16_GroupD_left_m3
- QSpinBox * spb_GS16_GroupD_left_m1
- QLabel * lbl_GS16_GroupD_sep1
- QPushButton * btn_GS16_GroupD_Simulate
- QPushButton * btn_GS16_GroupD_Save

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_eurowindow.h

3.42 Класс Ui_ExitDialog

Граф наследования:Ui_ExitDialog:

Открытые члены

- void setupUi (QDialog *ExitDialog)
- void retranslateUi (QDialog *ExitDialog)

Открытые атрибуты

- QLabel * lblSure
- QWidget * layoutWidget
- QHBoxLayout * horizontalLayout
- QPushButton * btnCancel
- QSpacerItem * horizontalSpacer
- QPushButton * btnYes

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_exitdialog.h

3.43 Класс Ui_MainWindow

Граф наследования:Ui_MainWindow:

Открытые члены

- void setupUi (QMainWindow *MainWindow)
- void retranslateUi (QMainWindow *MainWindow)

Открытые атрибуты

- QAction * actionFile
- QWidget * centralWidget
- QWidget * layoutWidget
- QVBoxLayout * verticalLayout_2
- QPushButton * btnNewComp
- QPushButton * btnLoadComp
- QPushButton * btnExit

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_mainmenu.h

3.44 Класс Ui_WarningDialog

Граф наследования:Ui_WarningDialog:

Открытые члены

- void setupUi (QDialog *WarningDialog)
- void retranslateUi (QDialog *WarningDialog)

Открытые атрибуты

- QWidget * layoutWidget
- QHBoxLayout * horizontalLayout
- QPushButton * btnOk
- QLabel * lblText

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_warningdialog.h

3.45 Класс WarningDialog

Граф наследования:WarningDialog:

Граф связей класса WarningDialog:

Открытые члены

- WarningDialog (QWidget *parent=0, const QString text="Error!")

Объявления и описания членов классов находятся в файлах:

- GUIApp/warningdialog.h
- GUIApp/warningdialog.cpp

3.46 Класс Ui::WarningDialog

Граф наследования:Ui::WarningDialog:

Граф связей класса Ui::WarningDialog:

Дополнительные унаследованные члены

Объявления и описания членов класса находятся в файле:

- GUIApp/ui_warningdialog.h

3.47 Класс WrongID

Граф наследования:WrongID:

Граф связей класса WrongID:

Открытые члены

- WrongID (int number)
- int getWrongID () const
- const char * what ()

Объявления и описания членов класса находятся в файле:

- Core/Exceptions/wrongid.h

3.48 Класс WrongNumberOfTeams

Граф наследования:WrongNumberOfTeams:

Граф связей класса WrongNumberOfTeams:

Открытые члены

- WrongNumberOfTeams (int number)
- int getWrongNum () const
- const char ∗ what ()

Объявления и описания членов класса находятся в файле:

- Core/Exceptions/wrongnumberofteams.h

3.49 Класс WrongPenaltyScore

Граф наследования:WrongPenaltyScore:

Граф связей класса WrongPenaltyScore:

Открытые члены

- const char ∗ what ()

Объявления и описания членов класса находятся в файле:

- Core/PlayoffStage/Exceptions/wrongpenaltyscore.h

Предметный указатель

addTeam
    Competition, 7
    Group, 12
Application, 5

Competition, 6
    addTeam, 7
    getGroupStage, 7
    getNumberOfTeams, 7
    getPlayoffStage, 7
    getTeams, 7
    getTitle, 7
    setTitle, 8
    startGroupStage, 8
    startPlayOffStage, 8
CompetitionIsOver, 8
CompetitionWindow, 9
CreateCompDialog, 9

decreaseGoals
    TeamStat, 35

EUROWindow, 10
ExitDialog, 11

getCoach
    Team, 29
getDraws
    TeamInGroup, 32
getFirstTeam
    Match, 20
getFouls
    TeamStat, 35
getGames
    TeamInGroup, 32
getGoalsAgainst
    TeamStat, 35
getGoalsDiffrence
    TeamStat, 35
getGoalsFor
    TeamStat, 35
getGroup
    GroupStage, 15
getGroupId
    IdGenerator, 16
getGroupStage
    Competition, 7
getGroups
    GroupStage, 15
getId
    Group, 12
    Match, 20
    Team, 29
getLoses
    TeamInGroup, 32
getMatchId
    IdGenerator, 16
getMatches
    Group, 12
getName
    Team, 29
getNumberOfGroups
    GroupStage, 15
getNumberOfTeams
    Competition, 7
getPlayoffStage
    Competition, 7
getPoints
    TeamInGroup, 32
getPossession
    TeamStat, 36
getRedCards
    TeamStat, 36
getResult
    Match, 20
getSecondTeam
    Match, 20
getShotOnTarget
    TeamStat, 36
getShots
    TeamStat, 36
getTeamId
    IdGenerator, 17
getTeams
    Competition, 7
    Group, 12
    Stage, 28
getTitle
    Competition, 7
getWinners
    Group, 13
    GroupStage, 15
getWins
    TeamInGroup, 32
getYellowCards
    TeamStat, 36
Group, 11
    addTeam, 12
    getId, 12

getMatches, 12
getTeams, 12
getWinners, 13
Group, 12
operator<<, 13
GroupStage, 14
getGroup, 15
getGroups, 15
getNumberOfGroups, 15
getWinners, 15
GroupStage, 14
GroupStageIsNotOver, 16
GroupsAreNotCreated, 13

IdGenerator, 16
getGroupId, 16
getMatchId, 16
getTeamId, 17
IdGeneratorInstance, 17
IdGeneratorInstance
IdGenerator, 17
increaseGoals
TeamStat, 36
InputPenaltryScore, 17
isPlayed
Match, 20

MainWindow, 18
Match, 18
getFirstTeam, 20
getId, 20
getResult, 20
getSecondTeam, 20
isPlayed, 20
Match, 19
operator<<, 21
operator>>, 21
setResult, 20
MatchInGroup, 21
setResult, 22
MatchInPlayoff, 22
setResult, 23

operator<
TeamInGroup, 32
operator<<
Group, 13
Match, 21
Team, 29
operator>>
Match, 21

Penalty, 23
PlayoffStage, 23

qt_meta_stringdata_CreateCompDialog_t, 24
qt_meta_stringdata_EUROWindow_t, 24
qt_meta_stringdata_ExitDialog_t, 25
qt_meta_stringdata_MainWindow_t, 25
qt_meta_stringdata_TeamCheckBoxes_t, 25
qt_meta_stringdata_WarningDialog_t, 25

random
Randomiser, 26
Randomiser, 26
random, 26
RandomiserInstance, 26
RandomiserInstance
Randomiser, 26
Round, 27

setFouls
TeamStat, 37
setGoals
TeamStat, 37
setPossession
TeamStat, 37
setRedCards
TeamStat, 37
setResult
Match, 20
MatchInGroup, 22
MatchInPlayoff, 23
setShotOnTarget
TeamStat, 37
setShots
TeamStat, 38
setTitle
Competition, 8
setYellowCards
TeamStat, 38
Stage, 27
getTeams, 28
startGroupStage
Competition, 8
startPlayOffStage
Competition, 8

Team, 28
getCoach, 29
getId, 29
getName, 29
operator<<, 29
Team, 28, 29
TeamCheckBoxes, 30
TeamInGroup, 30
getDraws, 32
getGames, 32
getLoses, 32
getPoints, 32
getWins, 32
operator<, 32
TeamInGroup, 31
TeamInPlayoff, 33
TeamStat, 33
decreaseGoals, 35
getFouls, 35
getGoalsAgainst, 35

getGoalsDiffrence, 35
getGoalsFor, 35
getPossession, 36
getRedCards, 36
getShotOnTarget, 36
getShots, 36
getYellowCards, 36
increaseGoals, 36
setFouls, 37
setGoals, 37
setPossession, 37
setRedCards, 37
setShotOnTarget, 37
setShots, 38
setYellowCards, 38
TeamStat, 34

Ui::CreateCompDialog, 9
Ui::Dialog, 10
Ui::EUROWindow, 10
Ui::ExitDialog, 10
Ui::MainWindow, 18
Ui::WarningDialog, 46
Ui_CreateCompDialog, 38
Ui_Dialog, 39
Ui_EUROWindow, 40
Ui_ExitDialog, 44
Ui_MainWindow, 44
Ui_WarningDialog, 45

WarningDialog, 45
WrongID, 46
WrongNumberOfTeams, 46
WrongPenaltyScore, 47

Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

Программирование

Отчет по лабораторной работе
"Редактор фубольного турнира"

Работу выполнил:
Дьячков В.В.
Группа: 13501/4
Преподаватель:
Вылегжанина К.Д.

Санкт-Петербург
2016

Содержание

1 Редактор футбольного турнира **2**
1.1 Задание . . . 2
1.2 Концепция . . . 2
1.3 Правила игры . . . 2
1.3.1 Основные правила . . . 2
1.3.2 Определение победителя . . . 2
1.3.3 Формат проведения чемпионата мира . . . 2
1.3.4 Формат проведения чемпионата Европы . . . 2
1.4 Минимально работоспособный продукт . . . 3
1.5 Диаграмма прецедентов использования . . . 3
1.6 Выводы . . . 3

2 Проектирование редактора футбольного турнира **4**
2.1 Диаграмма компонентов . . . 4
2.2 Интерфейс библиотеки . . . 4
2.3 Станадрт . . . 5

3 Реализация редактора футбольного турнира **5**
3.1 Среда разработки . . . 5
3.2 Основные классы . . . 5
3.3 Скриншоты экранов пользовательского интерфейса . . . 6
3.3.1 Командная строка . . . 6
3.3.2 Графический интерфейс . . . 8

4 Процесс обеспечения качества и тестирование **11**
4.1 Утилиты . . . 11
4.2 Непрерывная интеграция . . . 12
4.3 Ревью и демо . . . 12
4.4 Тестирование . . . 12

5 Выводы **12**

6 Листинги основных классов **12**

1 Редактор футбольного турнира

1.1 Задание

Футбол — командный вид спорта, в котором целью является забить мяч в ворота соперника ногами или другими частями тела (кроме рук) большее количество раз, чем команда соперника. В настоящее время самый популярный и массовый вид спорта в мире.

Редактор футбольного турнира, предназначенный для учета результатов чемпионата мира или Европы по футболу. Программа должна позволять задать список участвующих команд, редактировать результаты проведённых матчей, автоматически подводить итоги этапов турнира (подсчитывать очки, забитые и пропущенные мячи, определять занятые командами места).

1.2 Концепция

Созданное приложение должно давать пользователю возможность вести учет футбольного соревнования, включая систему сохранения и загрузки турнира. Также, редактор должен обладать графическим интерфейсом, который позволит визуализировать заявленную функциональность и упростить общение с программой.

1.3 Правила игры

1.3.1 Основные правила

Цель игры — забить мяч в ворота противника, сделать это как можно большее количество раз и постараться не допустить гола в свои ворота. Каждая команда состоит максимум из одиннадцати игроков (без учета запасных), один из которых должен быть вратарём. Отдельной футбольной игрой называется матч, который в свою очередь состоит из двух таймов по 45 минут и иногда дополнительного времени.

1.3.2 Определение победителя

Победителем считается команда, забившая в течение матча большее количество голов. Если обе команды забьют одинаковое количество голов или не забьют ни одного гола, матч считается сыгранным вничью.

Дополнительное время в мачтах стадии плейф-офф

Дополнительное время в футбольных матчах стадии плей-офф назначается для разрешения ничьих в основное время. Это два дополнительных тайма по 15 минут и в случае ничьей пробивается серия пенальти.

1.3.3 Формат проведения чемпионата мира

Групповой этап На стадии группового турнира команды делятся на 8 групп по четыре команды. Состав групп определяется жеребьёвкой, во время которой 8 команд («сеяные») получают первые номера в своих группах, а остальные 24 команды разделяются на три корзины в соответствии с рейтингом сборных-участниц. В каждой группе команды проводят однокруговой турнир из трёх туров. За победу в матче команда получает 3 очка, за ничью — 1, за поражение — 0. Из каждой группы во второй раунд (плей-офф) выходят команды, занявшие первые два места, — всего 16 участников чемпионата.

Стадия плей-офф Во втором раунде команды играют матчи на выбывание — проигравший (за исключением участников полуфинальных игр) заканчивает своё участие в турнире. Если счёт после окончания основного времени ничейный, назначается дополнительное время, а если и оно не помогло выявить победителя, проводится серия пенальти. В 1/8 финала победители групп первого раунда играют со вторыми командами из других групп. Победители этих игр выходят в 1/4 финала, победители 1/4 финалов — в полуфинал. Команды, проигравшие в полуфиналах, разыгрывают третье место в матче между собой. Команды, победившие в полуфиналах, играют финальный матч. Проигравший финалист награждается серебряными медалями. Победители финала получают звание чемпионов мира.

1.3.4 Формат проведения чемпионата Европы

Групповой этап На стадии группового турнира команды делятся на 4 групп по четыре команды. Сборные играют по формату лиги, только команды играют со своими соперниками в группе по одному разу. Состав групп определяется жеребьёвкой, во время которой 4 команды («сеяные») получают первые номера в своих группах, а остальные 12 команды разделяются на три корзины в соответствии с рейтингом сборных-участниц. За победу в матче команда получает 3 очка, за ничью — 1, за поражение — 0. Из каждой

группы во второй раунд (плей-офф) выходят команды, занявшие первые два места, — всего 8 участников чемпионата.

Стадия плей-офф Во втором раунде команды играют матчи на выбывание — проигравший заканчивает своё участие в турнире.Если в матчах на вылет после основного времени матча счёт равный, тогда назначается дополнительное время, если оно не выявило победителя, проходит серия пенальти. Победители 1/4 финала проходят в полуфинал. Победители полуфинала играют финальный матч. Проигравшие полуфиналисты награждаются бронзовыми медалями. Проигравшая команда награждается серебряными медалями. Победитель финала становится чемпионом и награждается золотыми медалями.

1.4 Минимально работоспособный продукт

Консольное приложение, позволяющее пользователю вести учет результатов футбольного турнира: задавать список команд, выставлять результаты прошедших матчей и подводить итоги соревнований.

1.5 Диаграмма прецедентов использования

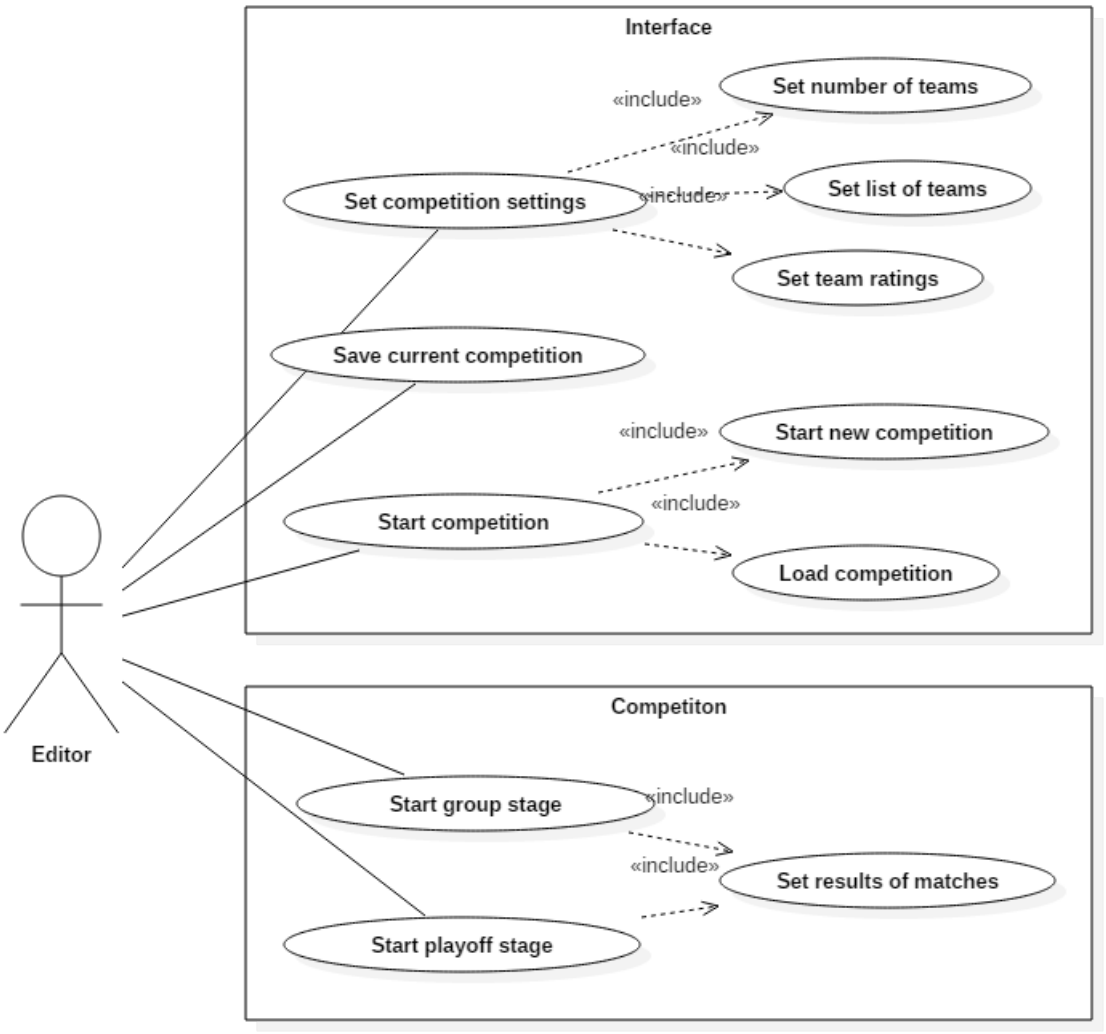


Рис. 1: Диаграмма прецедентов использования

1.6 Выводы

В данном разделе была рассмотрена предметная область данного приложения, приведенеы правила игры в футбол и определены диаграмма прецедентов использования и минимально работоспособный продукт.

2 Проектирование редактора футбольного турнира

В процессе проектирования было решено выделить 4 подпроекта:

1. Core - библиотека, содержащfz всю бизнес-логику проекта;
2. ConsoleApp - консольное приложение, позволяющее взаимодействовать пользователю с библиотекой через командную строку;
3. GUIApp - графическое приложение, позволяющее взаимодействовать пользователю с библиотекой через графический интерфейс. Создано средставами Qt Designer.
4. Tests - тесты, позволяющие тестировать функциональность библиотеки с помощью автоматических тестов.

2.1 Диаграмма компонентов

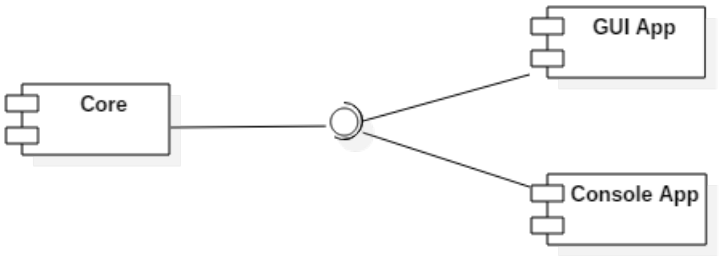


Рис. 2: Диаграмма компонентов

2.2 Интерфейс библиотеки

Ввиду особенностей задачи было решено не выделять интерфейс библиотеки в отдельный класс, а использовать методы класса `Competition` напрямую.

1. Метод, позволяющий задать название турнира

   `void setTitle( string title );`

2. Метод, позволяющий добавить команду в турнир

   `void addTeam( string name );`

3. Метод, позволяющий получить название турнира

   `string &getTitle();`

4. Метод, позволяющий текущее количество команд, участвующих в турнире
5. Метод, позволяющий получить команды, участвующие в турнире

   `vector<Team> &getTeams();`

6. Метод, позволяющий начать групповую стадию турнира

   `void startGroupStage();`

7. Метод, позволяющий получить объект класса `GroupStage`

   `GroupStage &getGroupStage();`

8. Метод, позволяющий начать стадию плей-офф

   `void startPlayOffStage();`

9. Метод, позволяющий получить объект класса `PlayoffStage`

   `PlayoffStage &getPlayoffStage();`

10. Метод, позволяющий начать групповую стадию заново

    `void restartGroupStage();`

11. Метод, позволяющий удалить команды, участвующие в турнире

    `void clearTeams();`

2.3 Станадрт

В качестве стандарта языка был выбран C++11. Разработка велась в IDE Qt Creator. Разработка графического приложения использовалась среда Qt Designer.

3 Реализация редактора футбольного турнира

3.1 Среда разработки

В процессе разработки использовались 2 среды:

1. Linux Ubuntu 14.04
   - IDE: Qt Creator 3.5.1 основанная Qt 5.4.0
   - Компилятор: GCC 4.8.4 (x86 64 bit)
2. Windows 8.1
   - IDE: Qt Creator 3.6.0 основанная Qt 5.5.1 (MSVC 2013, 32 bit)
   - Компилятор: MinGW 4.9.2 32bit

3.2 Основные классы

Были написаны классы для каждой составляющей футбольного соревнования:

- `Competition`, упомянутый выше
- `GroupStage` для представления групповой стадии турнира
- `PlayoffStage` для представления плей-офф стадии турнира
- `Team` для представления команды, участвующей в турнире
- `TeamStat` расширяющий класс `Team` статистикой
- `Group` для представления группы в групповой стадии турнира
- `Round` для представления раунда в плей-офф стадии турнира
- `Match` для представления матча между двумя командами
- `IdGenerator` для создания уникальных ID у команд, групп и матчей
- `Randomiser` для получения случайного значения

3.3 Скриншоты экранов пользовательского интерфейса

3.3.1 Командная строка

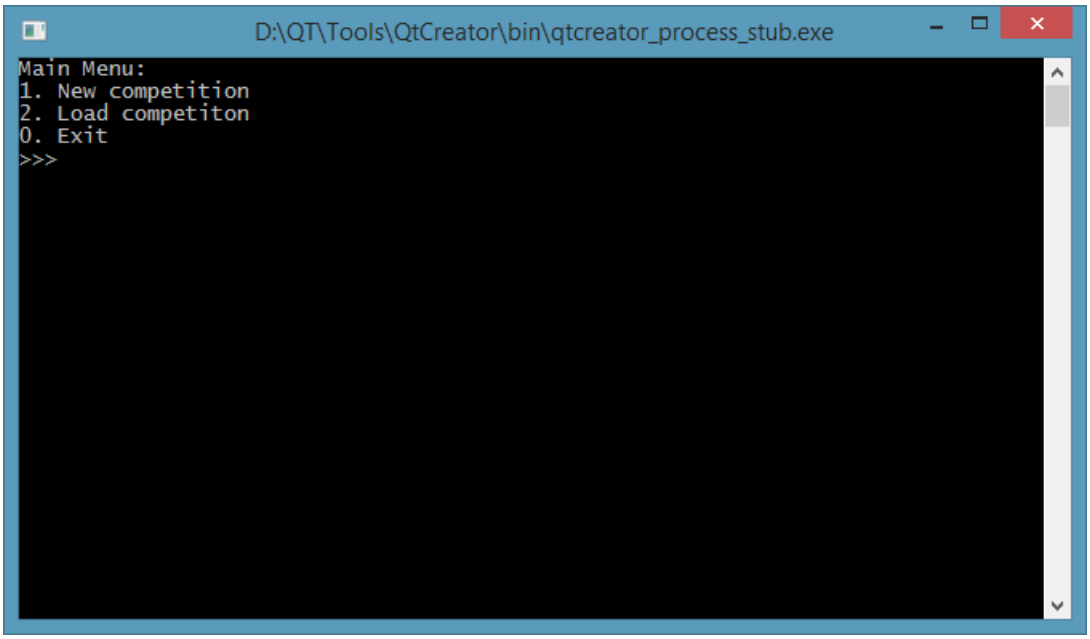


Рис. 3: Главное меню

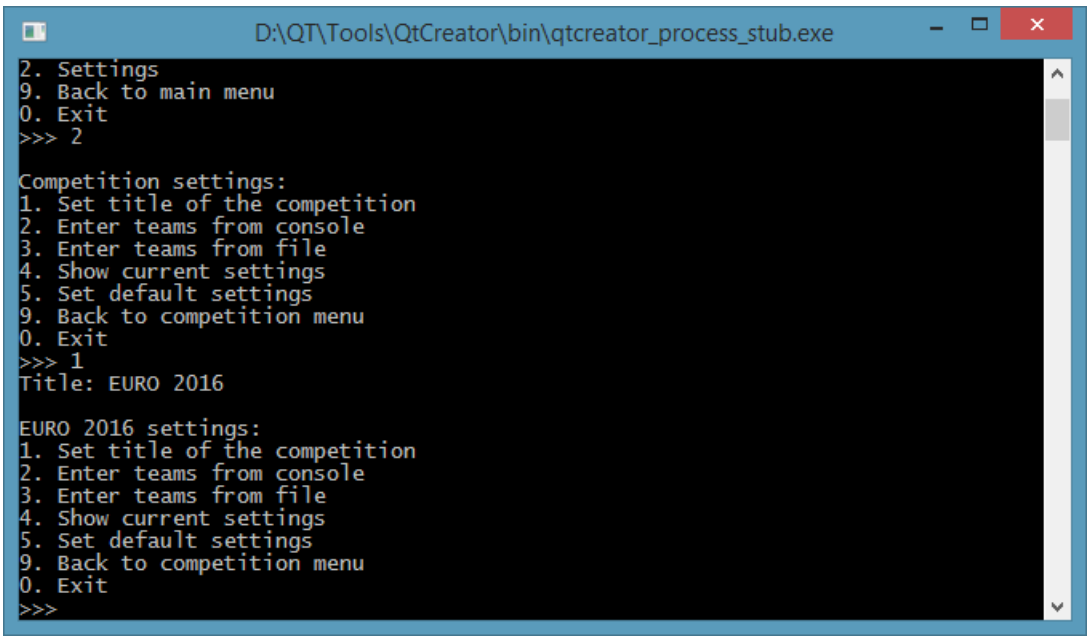
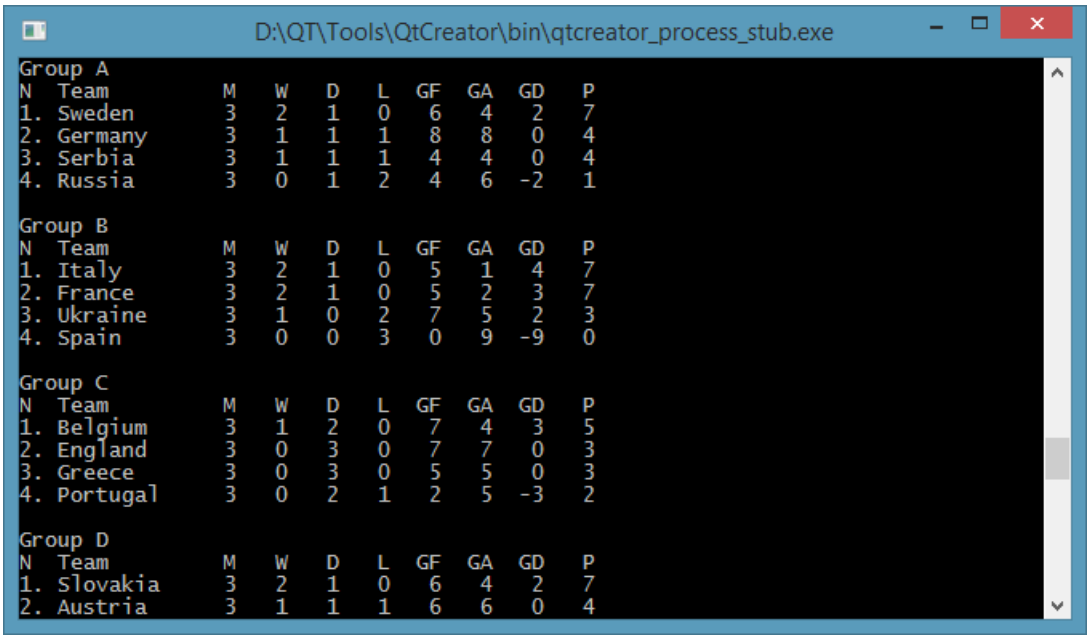


Рис. 4: Настройки



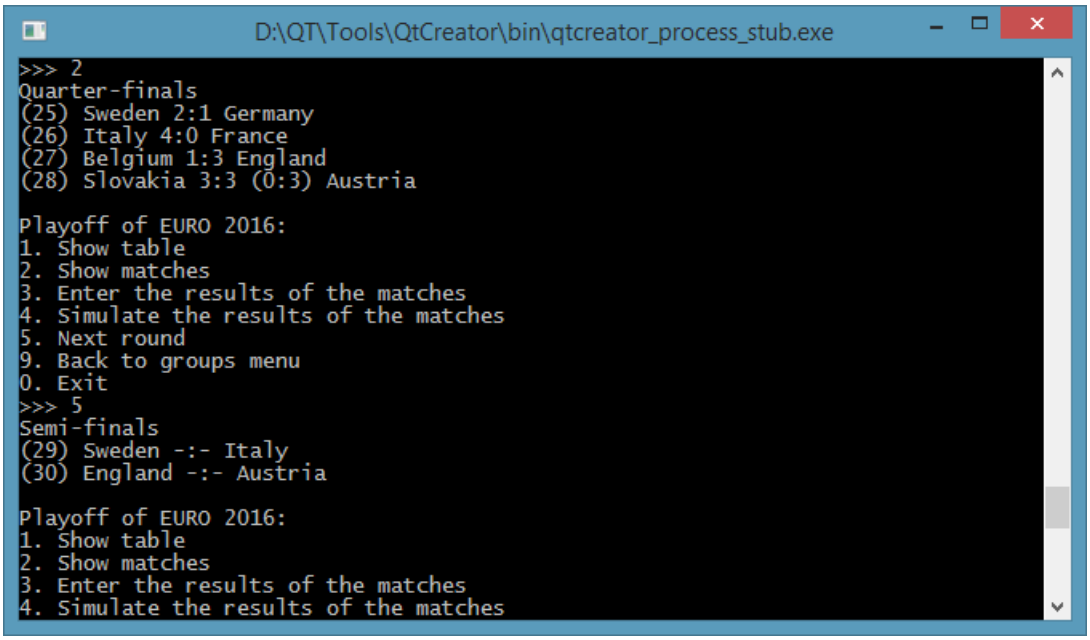
```
D:\QT\Tools\QtCreator\bin\qtcreator_process_stub.exe
Group A
N  Team          M   W   D   L  GF  GA  GD   P
1. Sweden        3   2   1   0   6   4   2   7
2. Germany       3   1   1   1   8   8   0   4
3. Serbia        3   1   1   1   4   4   0   4
4. Russia        3   0   1   2   4   6  -2   1

Group B
N  Team          M   W   D   L  GF  GA  GD   P
1. Italy         3   2   1   0   5   1   4   7
2. France        3   2   1   0   5   2   3   7
3. Ukraine       3   1   0   2   7   5   2   3
4. Spain         3   0   0   3   0   9  -9   0

Group C
N  Team          M   W   D   L  GF  GA  GD   P
1. Belgium       3   1   2   0   7   4   3   5
2. England       3   0   3   0   7   7   0   3
3. Greece        3   0   3   0   5   5   0   3
4. Portugal      3   0   2   1   2   5  -3   2

Group D
N  Team          M   W   D   L  GF  GA  GD   P
1. Slovakia      3   2   1   0   6   4   2   7
2. Austria       3   1   1   1   6   6   0   4
```

Рис. 5: Групповая стадия



```
D:\QT\Tools\QtCreator\bin\qtcreator_process_stub.exe
>>> 2
Quarter-finals
(25) Sweden 2:1 Germany
(26) Italy 4:0 France
(27) Belgium 1:3 England
(28) Slovakia 3:3 (0:3) Austria

Playoff of EURO 2016:
1. Show table
2. Show matches
3. Enter the results of the matches
4. Simulate the results of the matches
5. Next round
9. Back to groups menu
0. Exit
>>> 5
Semi-finals
(29) Sweden -:- Italy
(30) England -:- Austria

Playoff of EURO 2016:
1. Show table
2. Show matches
3. Enter the results of the matches
4. Simulate the results of the matches
```

Рис. 6: Стадия плей-офф

3.3.2 Графический интерфейс

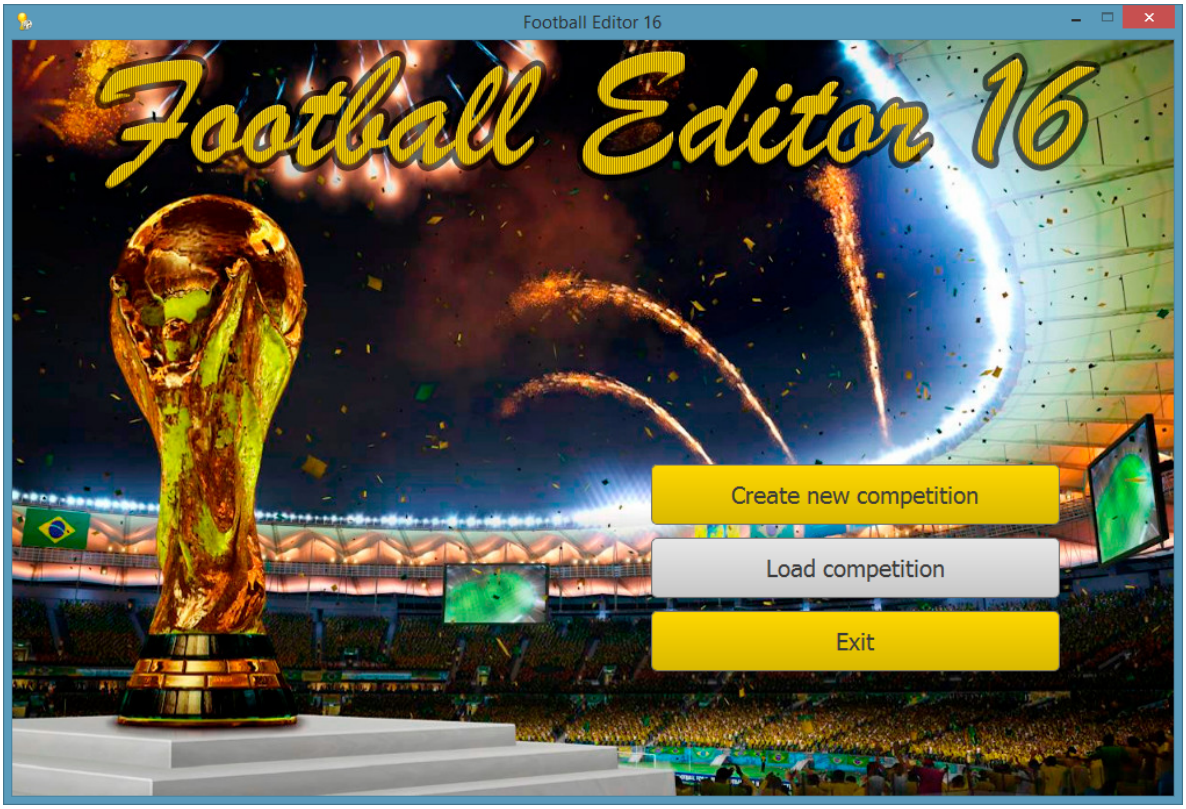


Рис. 7: Главное меню

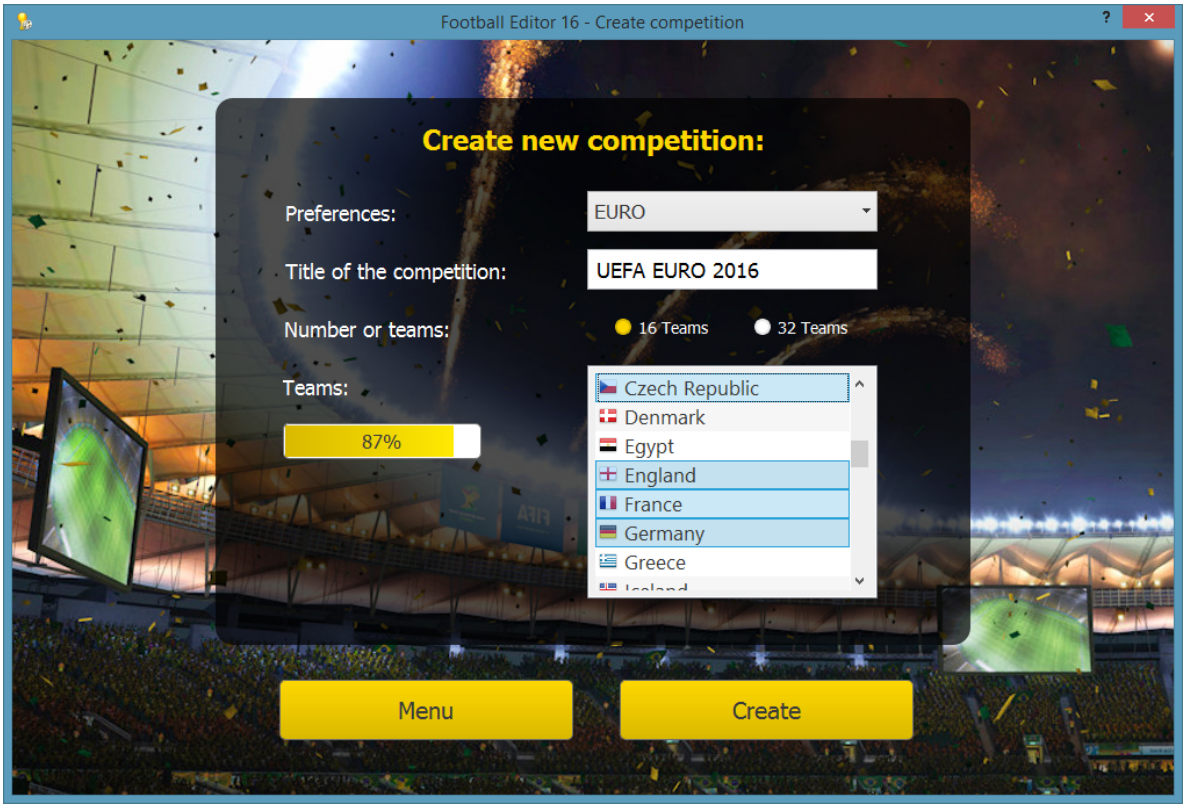


Рис. 8: Настройки

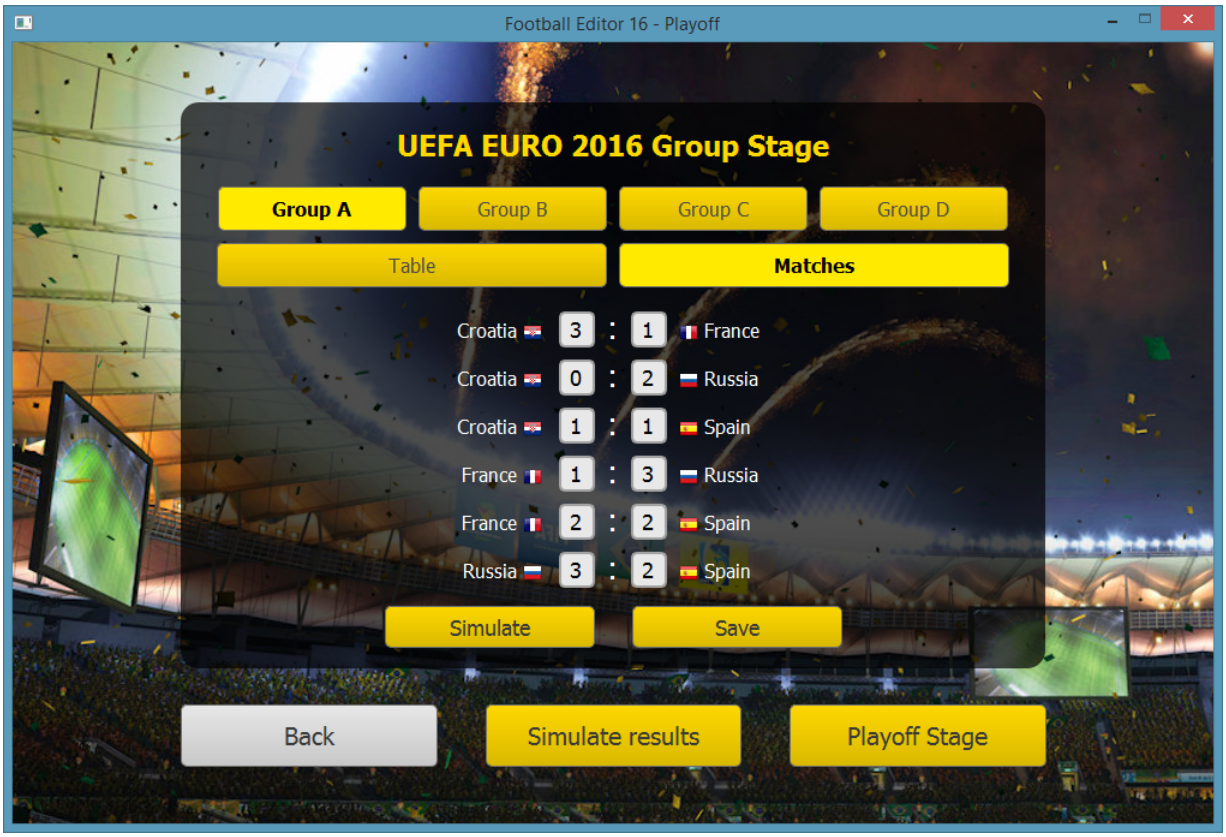


Рис. 9: Матчи в групповой стадии

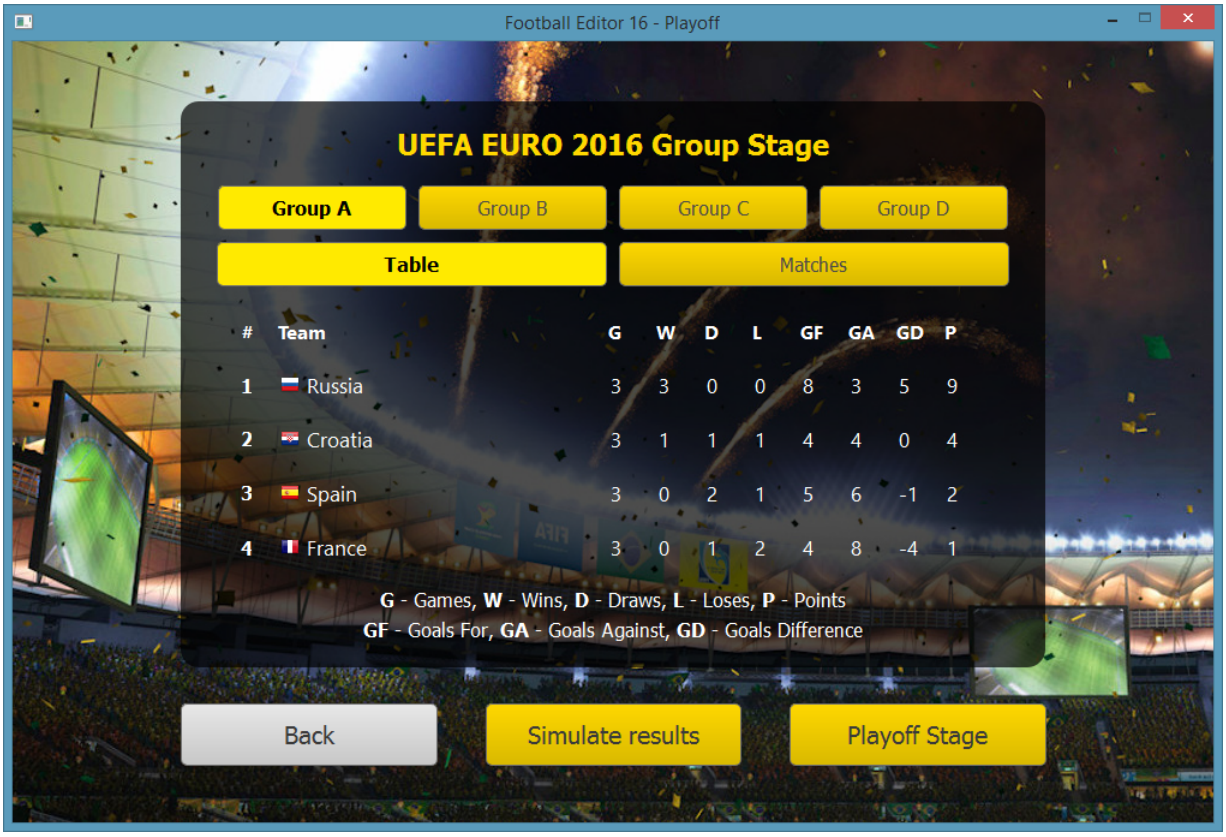


Рис. 10: Таблица в групповой стадии

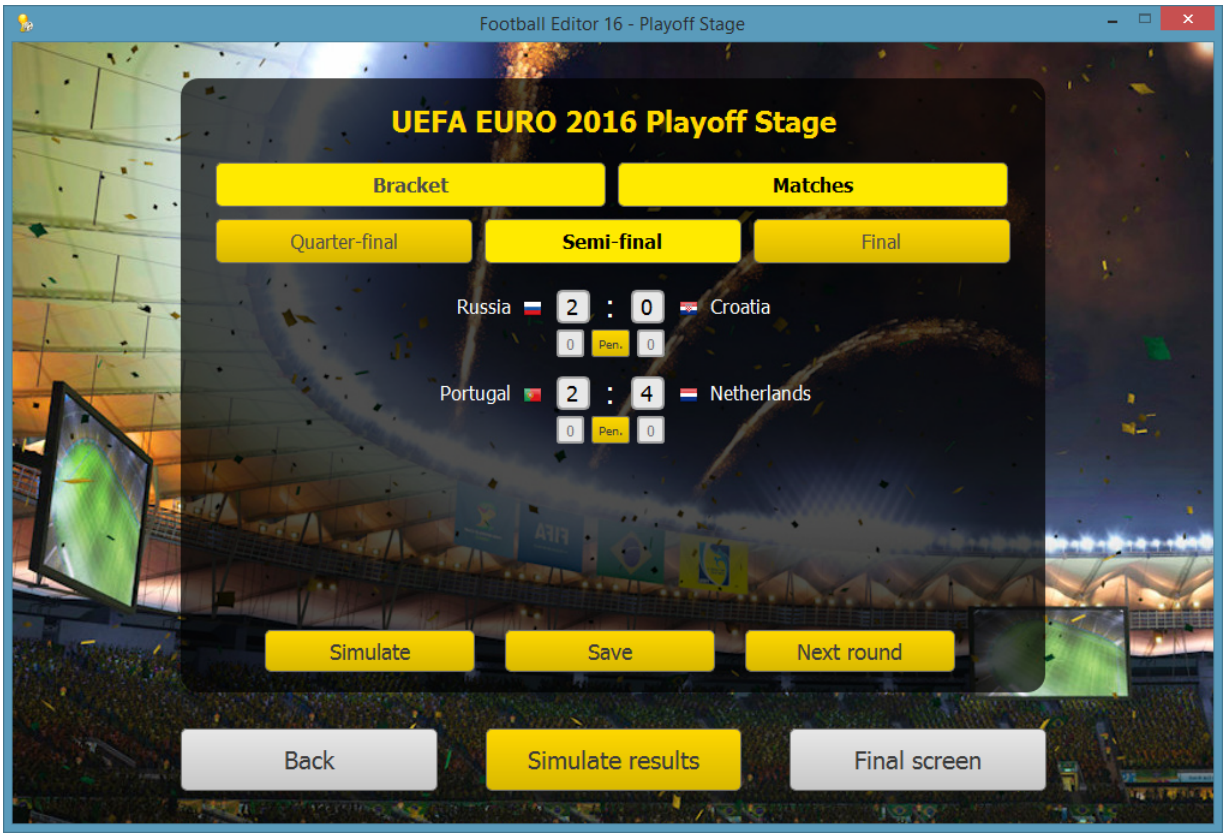


Рис. 11: Матчи в плей-офф

Рис. 12: Таблица в плей-офф

4 Процесс обеспечения качества и тестирование

4.1 Утилиты

В процессе использовались множество утилит, таких как:

- qmake – утилита для генерирования make-файла;
- Doxygen – утилита для документации кода;
- Cppcheck – утилита для анализа C++ кода;
- CLOC – утилита для подсчета количества строк кода;

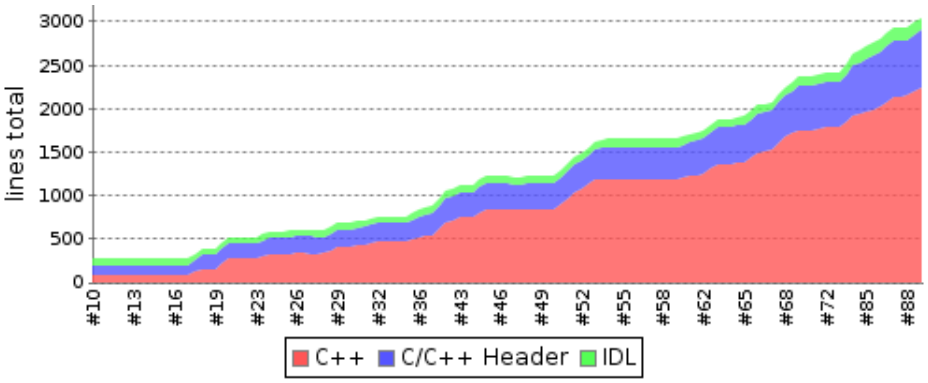


Рис. 13: Утилита CLOC

- Gcovr – утилита для расчета покрытия тестами;

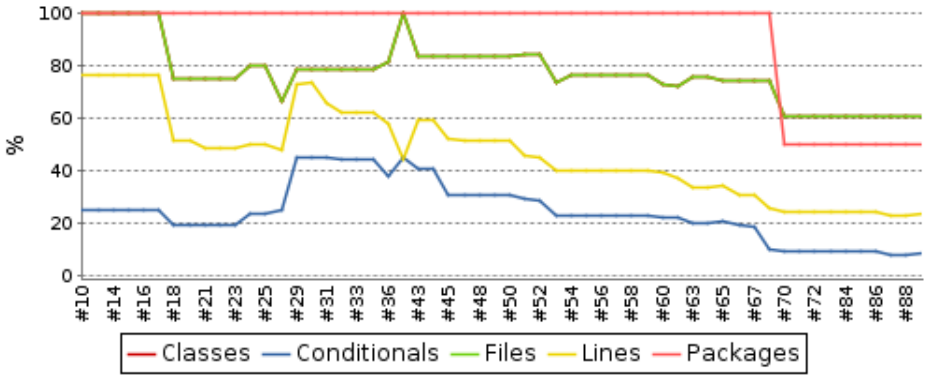


Рис. 14: Утилита Gcovr

- Valgrind – утилита для нахождения утечек памяти.

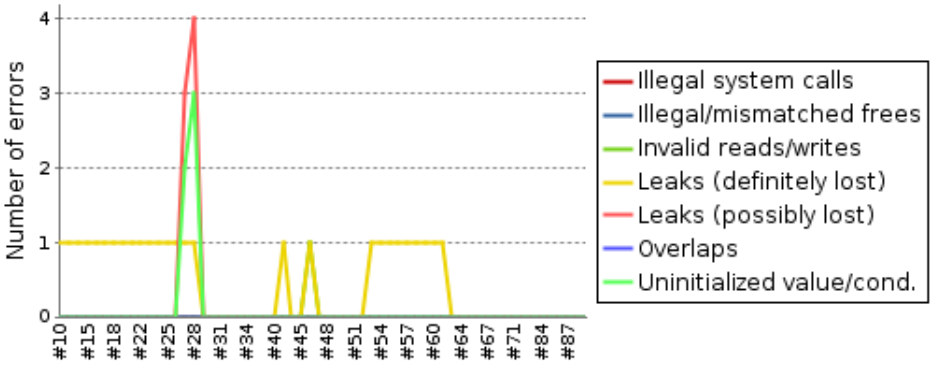


Рис. 15: Утилита Valgrind

4.2 Непрерывная интеграция

В процессе работы использовалась система непрерывной интеграции Jenkins. **Непрерывная интеграция** (англ. Continuous Integration, далее CI) — это практика разработки программного обеспечения, которая заключается в выполнении частых автоматизированных сборок проекта для скорейшего выявления и решения проблем интеграции результатов работы нескольких разработчиков. С помощью интеграционных сборок можно избавиться от синдрома «не знаю, на моей машине всё работает». Также мы защищаемся от «плохого кода», часто повторяющихся багов, «кривых» слияний. CI увеличивает возможности обратной связи потому, что она позволяет следить за состоянием проекта в течение разработки.

4.3 Ревью и демо

В течении семестра были сделаны только одно ревью и только одно демо, что несомненно сказалось в худшую сторону на качестве кода. Тем не менее теперь я понял значимость ревью и осознал его необходимость в процессе работы.

4.4 Тестирование

При разработке приложения были написаны автоматические тесты при помощи фреймворка QtTest, целью которых было выявить неправильное поведение библиотеки. Из-за того, что они покрывали не все классы, как видно на рис. 14, то не все сценарии работы приложения тестировались автоматически. В процессе выполнения задания также производилось ручное тестирование.

5 Выводы

При выполнении задания были закреплены навыки в работе с объектно-ориентированным проектированием и получен опыт в организации многофайлового проекта, создании модульных тестов и использовании утилит, помогающих в разработке приложений.

Огромный шаг вперед сделан в плане практических навыков программирования и проектировании приложений.

Результатом работы стало рабочее приложение, позволяющее с помощью командной строки или графического интерфейса вести учет футбольного турнира, например Чемпионата Европы и Чемпионата Мира.

6 Листинги основных классов

Листинг 1: competition.h

```
#ifndef COMPETITION_H
#define COMPETITION_H

#include "string"
#include "vector"
#include "team.h"
```

```
#include "GroupStage/groupstage.h"
#include "GroupStage/group.h"
#include "PlayoffStage/playoffstage.h"
#include "Exceptions/wrongid.h"
#include "Exceptions/wrongnumberofteams.h"
#include "Exceptions/groupstageisnotover.h"
#include "GroupStage/Exceptions/groupsarenotcreated.h"
using std::string;
using std::vector;

/**
 * @brief The Competition class
 */
class Competition
{
public:
    Competition();
    /**
     * @brief Set the title of the competition
     * @param title Competition name
     */
    void setTitle( string title );
    /**
     * @brief Add the team to the current competition
     * @param name Name of the team
     */
    void addTeam( string name );

    /**
     * @brief Get title of the competition
     * @return title of the competition
     */
    string &getTitle();
    /**
     * @brief Get number of teams participating in the competition
     * @return number of teams
     */
    unsigned getNumberOfTeams() const;
    /**
     * @brief Get vector of teams participating in the competition
     * @return vector of teams
     */
    vector<Team> &getTeams();
    /**
     * @brief Save the current competition
     */
    void saveCurrentCompetition();
    /**
     * @brief Load the competition
     */
    void loadCompetition();
    /**
     * @brief Launch group stage
     */
    void startGroupStage();
    /**
     * @brief Launch group stage with custom groups
     * @param teamIDs
     */
    void startGroupStage( vector<int> teamIDs );
    /**
     * @brief Create groups
     */
    void createGroups();
    /**
     * @brief Get group stage
     * @return group stage
     */
    GroupStage &getGroupStage();
    /**
     * @brief Get playoff stage
     * @return playoff stage
     */
    PlayoffStage &getPlayoffStage();
    /**
     * @brief Launch playoff stage
```

```
     */
    void startPlayOffStage();
    /**
     * @brief Launch playoff stage with custom pairs
     * @param teamIDs
     */
    void startPlayOffStage( vector<int> teamIDs );
    /**
     * @brief Restart group stage
     */
    void restartGroupStage();
    /**
     * @brief Delete all teams
     */
    void clearTeams();

    ~Competition();
private:
    string title;
    unsigned numberOfTeams;
    vector<Team> teams;

    // There are 2 stages in a football competition
    GroupStage* groupStage;
    PlayoffStage* playoffStage;
};

#endif // COMPETITION_H
```

Листинг 2: competition.cpp

```
#include "competition.h"

Competition::Competition()
{
    title = "Competition";
    numberOfTeams = 0;
    groupStage = 0;
    playoffStage = 0;
}

void Competition::setTitle(string title)
{
    this->title = title;
}

void Competition::addTeam(string name)
{
    teams.push_back(Team(name));
    numberOfTeams++;
}

string &Competition::getTitle()
{
    return title;
}

unsigned Competition::getNumberOfTeams() const
{
    return numberOfTeams;
}

vector<Team> &Competition::getTeams()
{
    return teams;
}

void Competition::startGroupStage()
{
    if( numberOfTeams % 4 == 0 && numberOfTeams != 0 ) {
        groupStage = new GroupStage(teams);
        groupStage->launch();
        groupStage->createGroups();
    }
    else
        throw WrongNumberOfTeams(numberOfTeams);
```

```
}

void Competition::startGroupStage(vector<int> teamIDs)
{
    if( numberOfTeams % 4 == 0 && numberOfTeams != 0 ) {
        groupStage = new GroupStage(teams);
        groupStage->launch();
        try {
            groupStage->createGroups(teamIDs);
        }
        catch( WrongID& e ) {
            delete groupStage;
            groupStage = 0;
            throw;
        }
    }
    else
        throw WrongNumberOfTeams(numberOfTeams);
}

void Competition::createGroups()
{
    if( numberOfTeams % 16 == 0 && numberOfTeams != 0 ) {
        vector<int> teamIDs;
        if( numberOfTeams == 16 )
            teamIDs = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 };
        else
            teamIDs = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
                        16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31 };
        groupStage = new GroupStage(teams);
        groupStage->launch();
        try {
            groupStage->createGroups(teamIDs);
        }
        catch( WrongID& e ) {
            delete groupStage;
            groupStage = 0;
            throw;
        }
    }
    else
        throw WrongNumberOfTeams(numberOfTeams);
}

GroupStage &Competition::getGroupStage()
{
    if( groupStage != 0 )
        return *groupStage;
    else
        throw GroupsAreNotCreated();
}

PlayoffStage &Competition::getPlayoffStage()
{
    if( playoffStage != 0 )
        return *playoffStage;
    else
        throw GroupStageIsNotOver();
}

void Competition::startPlayOffStage()
{
    if( groupStage != 0 && groupStage->getWinners().size() != 0 ) {
        vector<Team> &winners = groupStage->getWinners();
        vector<int> teamIDs;
        if( numberOfTeams == 16 )
            teamIDs = { winners[0].getId(), winners[3].getId(), winners[2].getId(), winners[1].getId(),
                        winners[4].getId(), winners[6].getId(), winners[7].getId(), winners[5].getId() };
        playoffStage = new PlayoffStage(groupStage->getWinners());
        playoffStage->launch();
        playoffStage->createMatches(teamIDs);
    }
    else
        throw GroupStageIsNotOver();
```

```
}

void Competition::startPlayOffStage(vector<int> teamIDs)
{
    if( groupStage->getWinners().size() != 0 ) {
        playoffStage = new PlayoffStage(groupStage->getWinners());
        playoffStage->launch();
        try {
            playoffStage->createMatches(teamIDs);
        }
        catch( WrongID& e ) {
            delete playoffStage;
            playoffStage = 0;
            throw;
        }
    }
    else
        throw GroupStageIsNotOver();
}

Competition::~Competition()
{
    if( groupStage )
        delete groupStage;
    if( playoffStage )
        delete playoffStage;
}

void Competition::restartGroupStage()
{
    if( groupStage )
        delete groupStage;
}

void Competition::clearTeams()
{
    numberOfTeams = 0;
    teams.clear();
}
```

Листинг 3: team.h

```
#ifndef TEAM_H
#define TEAM_H

#include "idgenerator.h"
#include "iostream"
#include "string"

using std::string;
using std::ostream;

/**
 * @brief The Team class
 */
class Team
{
public:
    /**
     * @brief Constructor of Team
     * @param name Team name with deafaul value "Team"
     * @param coach Team coach with default value "Coach"
     */
    Team( string name_ = "Team", string coach_ = "Coach" );
    /**
     * @brief Copy constructor of Team
     * @param other Variable of type Team
     */
    Team( const Team& another );
    /**
     * @brief Get team ID
     * @return team ID
     */
    int getId() const;
    /**
     * @brief Get name of the team
```

```
     * @return name of the team
     */
    string& getName();
    /**
     * @brief Get name of the coach
     * @return Name of the coach
     */
    string& getCoach();

    /**
     * @brief Overloading of operator <<
     * @param os Variable of type ostream
     * @param team Variable of type Team
     * @return output stream
     */
    friend ostream& operator<<( ostream& os, const Team& team );
    friend bool operator==( const Team& one, const Team& another );
private:
    int id;
    string name;
    string coach;
};

#endif // TEAM_H
```

Листинг 4: team.cpp

```
#include "team.h"

Team::Team( string name_, string coach_ ): name(name_), coach(coach_)
{
    IdGenerator *idGen = IdGenerator::IdGeneratorInstance();
    id = idGen->getTeamId();
}

Team::Team( const Team &another ): id(another.id), name(another.name), coach(another.coach) {}

int Team::getId() const
{
    return id;
}

string& Team::getName()
{
    return name;
}
string& Team::getCoach()
{
    return coach;
}

bool operator==( const Team &one, const Team &another )
{
    return one.getId() == another.getId();
}

ostream& operator<<( ostream& os, const Team& team )
{
    os << team.name;
    return os;
}
```

Листинг 5: match.h

```
#ifndef MATCH_H
#define MATCH_H

#include "team.h"
#include "teamstat.h"
#include "idgenerator.h"
#include "randomiser.h"
#include "string"
#include "sstream"
#include "algorithm"
#include "iostream"
```

```

using std::vector;
using std::istream;
using std::ostringstream;

/**
 * @brief The Match class
 */
class Match
{
public:
    /**
     * @brief Constructor of Match
     * @param firstTeam First team
     * @param secondTeam Second team
     */
    Match( TeamStat& firstTeam, TeamStat& secondTeam );
    /**
     * @brief Destructor
     */
    virtual ~Match() {}
    /**
     * @brief Set result of the match
     * @param firstTeamGoals Number of goals of first team
     * @param secondTeamGoals Number of goals of second team
     */
    virtual void setResult(int firstTeamGoals, int secondTeamGoals) = 0;
    /**
     * @brief Get result of the match
     * @return result of match in format "X:Y", where
     * X - goals of first team,
     * Y - goals of second team
     */
    string& getResult();
    /**
     * @brief Get ID of the match
     * @return ID of the match
     */
    int getId() const;
    /**
     * @brief Is game already played
     * @return true if game is played, false if game isn't played
     */
    bool isPlayed() const;

    /**
     * @brief Get reference to the first team
     * @return first team
     */
    TeamStat& getFirstTeam();
    /**
     * @brief Get reference to the second team
     * @return second team
     */
    TeamStat& getSecondTeam();

    /**
     * @brief Simulate result of the match
     * where number of goals of both teams are randomed from 0 to 4
     */
    virtual void simulate();
    /**
     * @brief Overloading of operator <<
     * @param os Variable of type ostream
     * @param match Variable of type Match
     * @return output stream
     * Output in format: 'Team1 vs Team2' or 'Team1 X:Y Team2'
     */
    friend ostream& operator<<( ostream& os, Match &match );
    /**
     * @brief Overloading of operator >>
     * @param is Variable of type istream
     * @param match Variable of type Match
     * @return input stream
     * Input in format: 'X:Y' or 'X-Y'
     */
```

```
    friend istream& operator>>( istream& is, Match &match );

private:
    int id;

protected:
    TeamStat firstTeam;
    TeamStat secondTeam;
    string result;
    virtual void updateResult();
};

#endif // MATCH_H
```

Листинг 6: match.cpp

```
#include "match.h"

Match::Match( TeamStat &firstTeam, TeamStat &secondTeam ): firstTeam(firstTeam), secondTeam(secondTeam), result("-:-")
{
    IdGenerator *idGen = IdGenerator::IdGeneratorInstance();
    id = idGen->getMatchId();
}

string& Match::getResult()
{
    return result;
}

int Match::getId() const
{
    return id;
}

TeamStat& Match::getFirstTeam()
{
    return firstTeam;
}

TeamStat &Match::getSecondTeam()
{
    return secondTeam;
}

bool Match::isPlayed() const
{
    if( result != "-:-" )
        return true;
    else
        return false;
}

void Match::simulate()
{
    Randomiser *randomiser = Randomiser::RandomiserInstance();
    int firstTeamGoals;
    int secondTeamGoals;
    if( randomiser->random( 0,1,firstTeam.getId()+secondTeam.getId()+firstTeam.getName()[2]+secondTeam.getName()[1]+this->id ) ) { // Chance is 2/3
        firstTeamGoals = randomiser->random(0,3,firstTeam.getId()+secondTeam.getId());
        secondTeamGoals = randomiser->random(0,3,firstTeam.getName()[2]+secondTeam.getName()[1]+this->id);
    }
    else {
        if( randomiser->random( 0,2,firstTeam.getId()+secondTeam.getId()+firstTeam.getName()[2]+secondTeam.getName()[1]+this->id ) ) { // Chance is 2/9
            firstTeamGoals = randomiser->random(0,4,firstTeam.getId()+secondTeam.getId());
            secondTeamGoals = randomiser->random(0,4,firstTeam.getName()[2]+secondTeam.getName()[1]+this->id);
        }
        else { // Chance is 1/9
            firstTeamGoals = randomiser->random(0,5,firstTeam.getId()+secondTeam.getId());
            secondTeamGoals = randomiser->random(0,5,firstTeam.getName()[2]+secondTeam.getName()[1]+this->id);
        }
```

```
    }
    setResult(firstTeamGoals, secondTeamGoals);
}

void Match::updateResult()
{
    ostringstream convert;
    convert << firstTeam.getGoalsFor() << ':' << secondTeam.getGoalsFor();
    result = convert.str();
}

ostream& operator<<( ostream &os, Match &match )
{
    os << "(" << match.getId() << ")␣" << match.getFirstTeam()
       << "␣" << match.getResult() << "␣"
       << match.getSecondTeam();
    return os;
}

istream& operator>>( istream &is, Match &match )
{
    int goalsOfFirstTeam, goalsOfSecondTeam;
    char separator;
    is >> goalsOfFirstTeam >> separator >> goalsOfSecondTeam;
    match.setResult(goalsOfFirstTeam, goalsOfSecondTeam);
    return is;
}
```

Листинг 7: groupstage.h

```
#ifndef GROUPSTAGE_H
#define GROUPSTAGE_H

#include "../stage.h"
#include "group.h"
#include "../Exceptions/wrongid.h"
#include "Exceptions/groupstageisnotover.h"

/**
 * @brief The GroupStage class
 */
class GroupStage : public Stage
{
public:
    /**
     * @brief Constructor of GroupStage
     * @param teams Vector of the teams
     */
    GroupStage( vector<Team> &teams );
    /**
     * @brief Create groups
     */
    void launch();
    /**
     * @brief Get vector of groups
     * @return vector of groups
     */
    vector<Group> &getGroups();
    /**
     * @brief Get number of groups
     * @return number of groups
     */
    unsigned getNumberOfGroups() const;
    /**
     * @brief Generate groups
     */
    void createGroups();
    /**
     * @brief Create groups
     */
    void createGroups( vector<int> teamIDs );
    /**
     * @brief Get winners of the group stage
     * @return vector of winners
     */
    vector<Team>& getWinners();
```

```
    /**
     * @brief Get group by ID (A, B, C, etc.)
     * @param id
     * @return group
     */
    Group& getGroup( char id );
    bool isGroupStageOver();
private:
    void determineWinners();
    vector<Group> groups;
    unsigned numberOfGroups;
    vector<Team> winners;
};

#endif // GROUPSTAGE_H
```

Листинг 8: groupstage.cpp

```
#include "GroupStage/groupstage.h"

GroupStage::GroupStage( vector<Team> &teams ): Stage( teams )
{
    numberOfGroups = teams.size()/4;
    groups.reserve(numberOfGroups);
    winners.reserve(numberOfGroups*2);
}

void GroupStage::launch()
{
    //createGroups();
}

vector<Group> &GroupStage::getGroups()
{
    for( Group &group : groups )
        group.sort();
    return groups;
}

unsigned GroupStage::getNumberOfGroups() const
{
    return numberOfGroups;
}

void GroupStage::determineWinners()
{
    winners.clear();
    vector<Team> winnersOfGroup;
    for( Group &group: groups ) {
        group.sort();
        winnersOfGroup = group.getWinners();
        winners.insert(winners.end(), winnersOfGroup.begin(), winnersOfGroup.end());
    }
}

vector<Team>& GroupStage::getWinners()
{
    if( isGroupStageOver() == 0 )
        throw GroupStageIsNotOver();
    determineWinners();
    return winners;
}

Group &GroupStage::getGroup( char id )
{
    for( Group &group: groups )
        if( id == group.getId() ) {
            group.sort();
            return group;
        }
    throw WrongID( id );
}

bool GroupStage::isGroupStageOver()
{
    for( Group &group: groups )
```

```
        for( MatchInGroup &match: group.getMatches() )
            if( match.getResult() == "-:-" )
                return false;
    return true;
}

void GroupStage::createGroups()
{
    for( unsigned i = 0; i < numberOfGroups; i++ ) {
        groups.push_back(Group());
        for( unsigned j = 0; j < 4; j++ )
            groups.back().addTeam(getTeams()[i+j*numberOfGroups]);
        groups.back().createMatches();
    }
}

void GroupStage::createGroups( vector<int> teamIDs )
{
    int id = 0;
    for( unsigned i = 0; i < numberOfGroups; i++ )
    {
        groups.push_back(Group());
        for( unsigned j = 0; j < 4; j++ ) {
            if( id >=0 || id <= (int)teams.size()-1 )
                groups.back().addTeam(getTeams()[teamIDs[id++]]);
            else
                throw WrongID(id);
        }
        groups.back().createMatches();
    }
}
```

Листинг 9: round.h

```
#ifndef ROUND_H
#define ROUND_H

#include "matchinplayoff.h"

class Round
{
public:
    Round( RoundEnum round );
    void addPair( TeamInPlayoff &first , TeamInPlayoff &second );
    RoundEnum getRoundEnum() const;
    void addTeams(vector<TeamInPlayoff> &teams_ );
    vector<TeamInPlayoff> &getTeams();
    vector<MatchInPlayoff> &getMatches();
    vector<TeamInPlayoff> &getWinners();
    void updateWinners();

private:
    RoundEnum round;
    vector<TeamInPlayoff> teams;
    vector<MatchInPlayoff> matches;
    vector<TeamInPlayoff> winners;
};

#endif // ROUND_H
```

Листинг 10: round.cpp

```
#include "round.h"

Round::Round( RoundEnum round ): round(round)
{
    winners.reserve(round*2);
    matches.reserve(round);
}

void Round::addPair(TeamInPlayoff &first , TeamInPlayoff &second )
{
    teams.push_back( first );
    teams.push_back( second );
    matches.push_back( MatchInPlayoff( first , second ) );
```

```
}

RoundEnum Round::getRoundEnum() const
{
    return round;
}

void Round::addTeams( vector<TeamInPlayoff> &teams_ )
{
    for( std::vector<TeamInPlayoff>::iterator it = teams_.begin(); it != teams_.end(); it+=2 )
        addPair( *it , *(it+1) );
}

vector<MatchInPlayoff> &Round::getMatches()
{
    return matches;
}

vector<TeamInPlayoff> &Round::getTeams()
{
    return teams;
}

vector<TeamInPlayoff> &Round::getWinners()
{
    return winners;
}

void Round::updateWinners()
{
    winners.clear();
    winners.reserve( matches.size() );
    for( MatchInPlayoff &match: matches )
        winners.push_back( match.getWinner() );
}
```