Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра Програмної інженерії

КУРСОВА РОБОТА

ПОЯСНЮВАЛЬНА ЗАПИСКА

з дисципліні “ Об’єктно -орієнтоване програмування”

ДОВІДНИК ФАНАТА

Керівник, асистент Ляпота В.М.

Студент гр. ПІ-15-2 Сушинський І.К.

Комісія:

Комісія: проф. \_\_\_\_\_\_\_\_\_\_\_\_\_ Дудар З.В.

асистент \_\_\_\_\_\_\_\_\_\_\_\_\_ Ляпота В.М.

лектор \_\_\_\_\_\_\_\_\_\_\_\_\_ Бондарэв В.М.

Харків 2016

**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**

Кафедра: ***Програмної інженерії***

Дисципліна: ***Об’єктно-орієнтоване програмування***

Спеціальність: ***Програмна інженерія***

Курс 1 .Група *ПІ-15-2* . Семестр *2* .

***ЗАВДАННЯ***

***на курсовий проект студента***

*Сушинського Ігоря Костянтиновича*

(Прізвище, Ім'я, По батькові)

1. Тема проекту: Довідник фаната
2. Термін здачі студентом закінченого проекту: ***“28” - травня - 2016 р***.
3. Вихідні дані до проекту:

*Специфікація програми, методичні вказівки до виконання курсової роботи*

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| *№* | *Назва етапу* | *Термін виконання* |
| 1 | Видача теми, узгодження і затвердження теми | 1-03-2016 р. |
| 2 | Формулювання вимог до програми | 15-03-2016 – 20-03-2016 р. |
| 3 | Розробка постановки задачі. | 20-03-2016 – 31-03-2016 р. |
| 4 | Розробка об’єктної моделі. | 01-04-2016 – 14-04-2016 р. |
| 5 | Колування програмної системи. | 15-04-2016 – 15-05-2016 р. |
| 6 | Тестування і доопрацювання розробленої програмної системи | 16-05-2016 – 20-05-2016 р. |
| 7 | Оформлення пояснювальної записки, додатків, графічного матеріалу | 25-05-2016 – 27-05-2016 р. |
| 8 | Захист | 28-05-2016. |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Прізвище, Ім'я, По батькові)

« 1 »\_березня\_\_\_\_\_\_\_\_\_\_ 2016 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 49 с., 20 рис., 1 додаток, 6 джерел.

Метою роботи є розробка програми “Довідник фаната” на засадах об'єктно-орієнтованого програмування.

Методи розробки базуються на використанні середи розробки Microsoft Visual Studio 2015, Windows Forms, платформи .NET Framework 4.5, мови програмування C#.

В результаті отримана програма під назвою “Довідник фаната”, яка дозволяє переглядати дані професіональних гравців, добавляти, редагувати, порівнювати їх між собою. Для більш зручного вибору конкретного гравця існує пошук та фільтрація. Існує можливість занести свої дані до “мого профілю”, та шукати гравців у списку похожих на вас, та порівнювати себе з другими. Можливо провести синхронізацію так званої бази даних, шляхом вибору файла. Також є прогнозування результату гри при виборі двух команд.

ПРОГРАМА, ОБ’ЕКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, КЛАС, МОВА ПРОГРАМУВАННЯ C#, ПЛАТФОРМА .NET, КІБЕРСПОРТ, DOTA 2, COUNTER-STRIKE GLOBAL OFFENSIVE.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc452320026)

[1 Спецификация программы 9](#_Toc452320027)

[1.1 Состав программы 9](#_Toc452320028)

[1.2 Интерфейс пользователя 9](#_Toc452320029)

[1.2.1 Главная форма и её функции. 10](#_Toc452320030)

[1.2.2 Форма со списком игроков. 14](#_Toc452320031)

[1.2.3 Форма “Мой профиль”. 18](#_Toc452320032)

[2 Проектная спецификация 21](#_Toc452320033)

[2.1 Общая архитектура программы. 21](#_Toc452320034)

[2.2 Пользовательский интерфейс. 22](#_Toc452320035)

[2.3 Формат данных. 23](#_Toc452320036)

[3 Руководство пользователя 25](#_Toc452320037)

[ВЫВОДЫ 26](#_Toc452320038)

[ПЕРЕЧЕНЬ ССЫЛОК 27](#_Toc452320039)

[Приложение А. Частичный код программы. 28](#_Toc452320040)

ВВЕДЕНИЕ

Фанаты — это люди которые увлекаются какой-то определённой вещью. Зачастую понятие фанат, ассоциируется с фанатом официального вида спорта, но также существует и такое понятие как киберспорт. Киберспорт - соревнования по компьютерным видеоиграм. Существует много различных дисциплин, такие как, Dota 2, Counter Strike Global Offensive, Heroes of the Storm, Hearthstone, Starcarft, Warcraft и др. В данной крусовой работе речь пойдёт о двух первых дисциплинах, это Dota 2 и Counter-Strike Global Offensive.

По выбранным дисциплинам проводятся очень много турниров, с достаточной большим денежным вознаграждением, за призовые места. Эти турниры смотрят очень много людей по всему миру, взять в пример последний крупный турнир Epicentr, статистика по просмотрам приведена на картинке ниже:



Из предоставленной информации можно сделать вывод что при таком количестве фанатов, данная тема очень актуальна и неплохо было бы иметь программу для сравнения, например, себя с профессиональным игроком или же со своим другом. А также будет очень полезная функция гипотетическое предсказание исхода матча.

В данной курсовой работе реализуется программа на языке программирование C#. Целью данной курсовой работы является разработка программы-справочника, с помощью которого можно просматривать информацию по игрокам, сравнивать информацию одного игрока с другим. Поиск и фильтрация списка игроков. Посмотреть гипотетический исход матча. А также совершить синхронизацию “базы данных”. За основу взят объектно-ориентированный подход.

Задачи выполнения работы:

1. исследование предметной области с целью выявления основных принципов данной сферы;
2. проектирование иерархии классов, интерфейсов, взаимодействия компонентов на основе выделенных принципов и данных средств;
3. использование встроенные элементы среды для структуризации классов и оптимизации кода;
4. применение принципа инкапсуляции к классам;

Объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. В данном подходе основными концепциями являются понятия объектов и классов.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования. Использование этого подхода предоставляет программисту широкие возможности в функциональности и в сопровождаемости проекта.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2015 Professional. Visual Studio представляет собой интегрированную среду разработки программ, созданную корпорацией Microsoft. Язык программирования C#.

# СПЕЦИФИКАЦИЯ ПРОГРАММЫ

Спецификацией данной программы является просмотр, сравнение, редактирование, удаление игроков в базе данных и синхронизация. Поиск и фильтрация списка игроков, прогнозирование результата матча. Создание своего профиля.

## Состав программы

Программа состоит из набора классов и графического интерфейса пользователя.

Структура программы состоит из 4 частей которые очень тесно взаимосвязаны между собой.

## Интерфейс пользователя

Используются 3 основных формы, которые очень тесно взаимосвязаны между собой, действие на одной форме приводит к изменению результатов отображения на остальных формах:

1. Главная форма – отображает список топ-10 по двум дисциплинам, позволяет переключиться на форму списка игроков и профиля, имеет кнопку синхронизации и форму расчета результатов игры.
2. Форма списка игроков – отображает список игроков выбранной дисциплины, позволяет совершать поиск, фильтрацию, удаление, добавление и редактирование игроков.
3. Форма “Мой профиль” – отображает данные о пользователе которые он вводит в специальную форму.

### Главная форма и её функции.

Работа с программой начинается с главного окна (см. рис.1.1). Здесь пользователь может увидеть список топ-10 по двум дисциплинам Dota 2 и Counter-Strike Global Offensive, две иконки этих дисциплин, при нажатии на которые откроется список игроков по определённой дисциплине, кнопку синхронизации, для выбора файла, кнопки для открытия формы просмотра профиля, а также выбор команд для прогноза результатов матча. Интерфейс программы прост в использование и интуитивно понятен. Это является необходимым условием при разработке пользовательских интерфейсов.

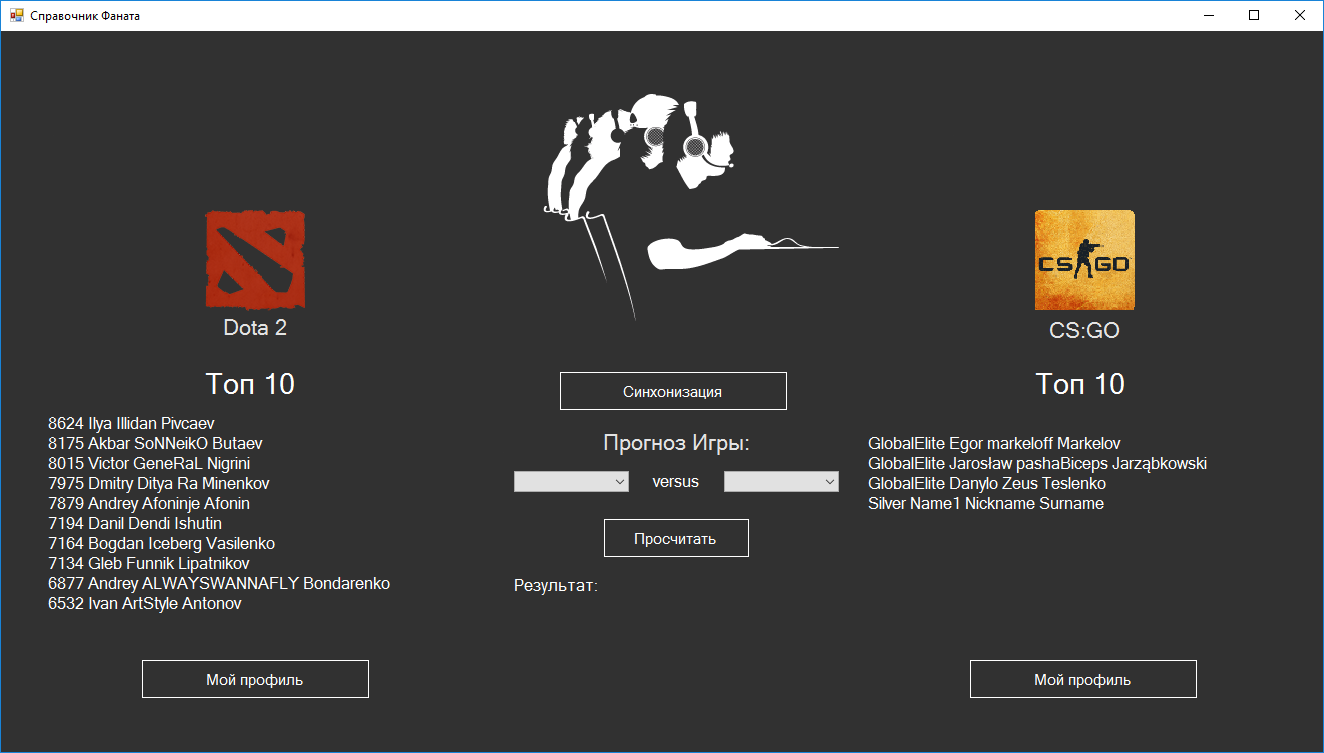


Рисунок 1.1 – Главная форма

Для выбора дисциплины и открытия формы со списком игроков, пользователю требуется нажать на иконку (см. рис.1.2 и рис.1.3) определённой игры.



Рисунок 1.2 – Иконка выбора дисциплины Dota 2

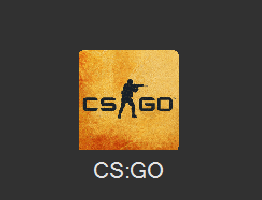


Рисунок 1.2 – Иконка выбора дисциплины Counter Strike Global Offensive

Для синхронизации данных, пользователь должен нажать на кнопку “Синхронизация” (см. рис.1.3), после в окне выбора файла (см. рис.1.4), выбрать текстовый файл со списком игроков.

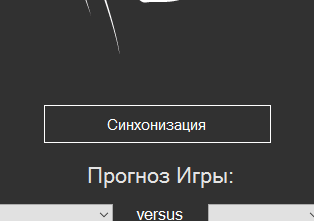


Рисунок 1.3 – Кнопка синхронизации

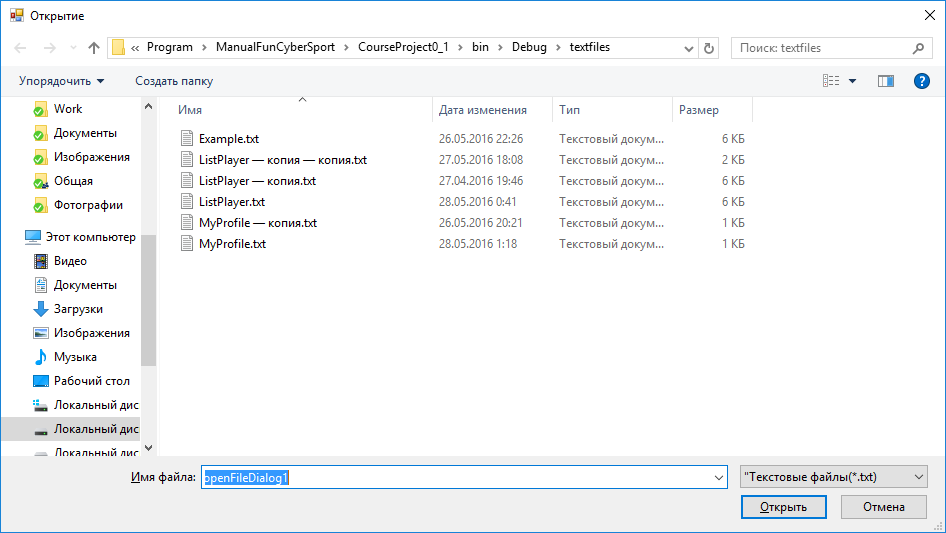


Рисунок 1.4 – Окно выбора текстового файла

Для прогнозирования исхода игры пользователь должен выбрать в списках 2 команды, и нажать на кнопку просчитать (см. рис.1.5), результат отобразится немного ниже.

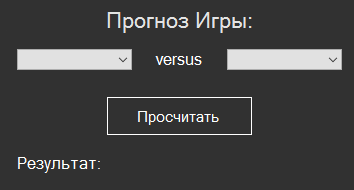


Рисунок 1.5 – Прогноз результата игры

Для отображения своего профиля пользователь должен нажать на соответствующую кнопку (для профиля Dota 2 слева, для Counter Strike Global Offensive справа) (см. рис.1.6 и рис.1.7).

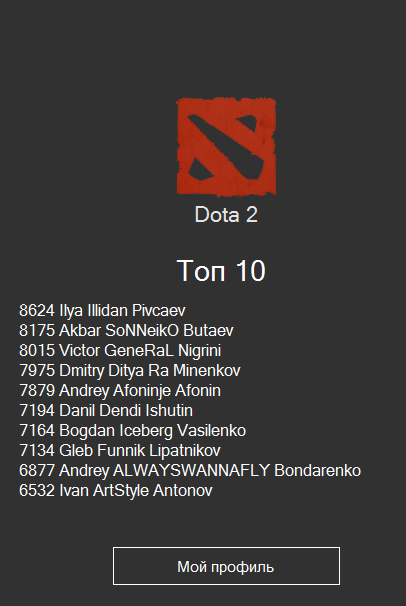


Рисунок 1.6 – Кнопка “Мой профиль” Dota 2



Рисунок 1.7 – кнопка “Мой профиль” Counter-Strike Global Offensive

### Форма со списком игроков.

После выбора дисциплины, открывается новая форма, где отображаются два списка игроков (см. рис.1.8). И их данные.

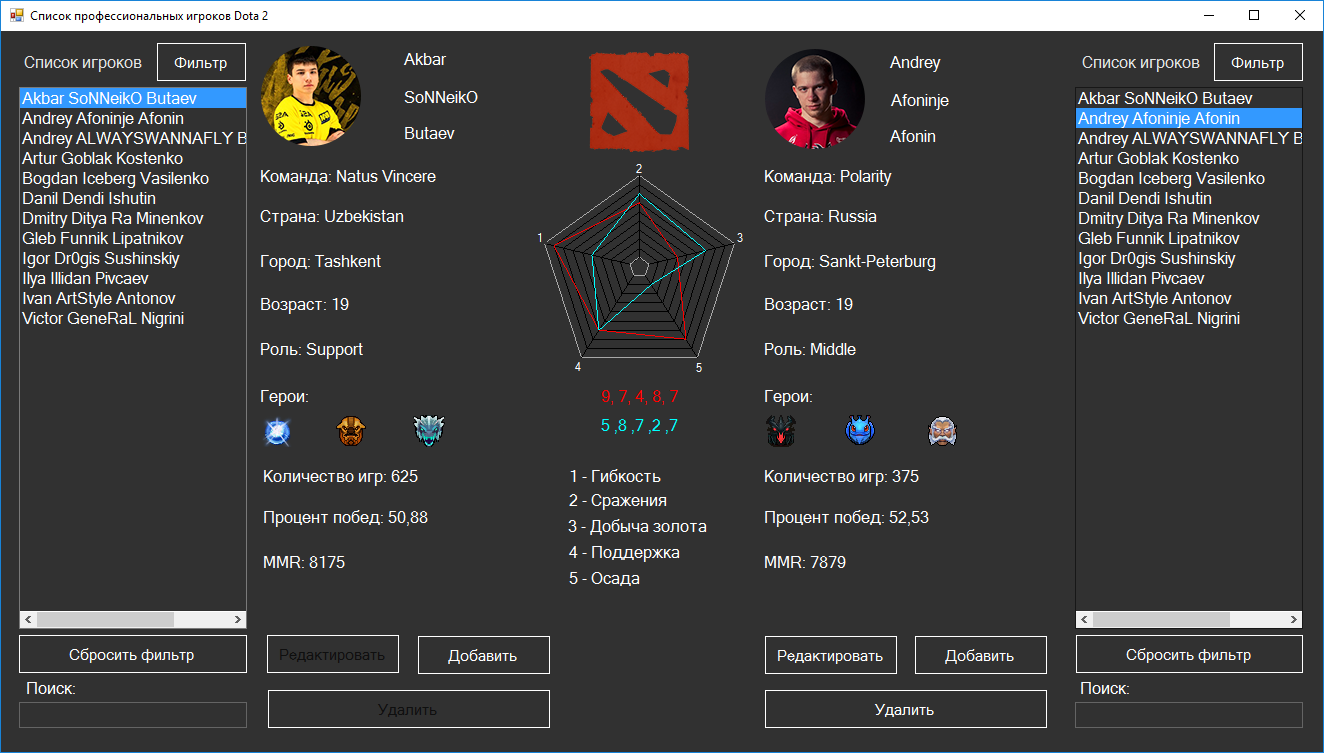


Рисунок 1.8 – Форма отображения списка игроков

Одна из важнейших особеностей это сравнение игроков по так называемому “5-угольнику силы” (см. рис.1.9), который отражает такие важные параметры как, гибкость, сражение, добыча золота, поддержка и осада.

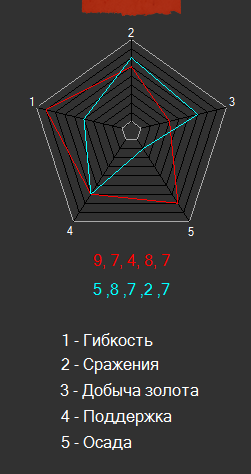


Рисунок 1.9 – “5-угольнику силы” двух выбраных игроков

Так же на окне этой формы расположенное текстовое поле поиска (см. рис.1.10), при введение данных в которое список игроков будет уменьшаться, в соответствии с запросом.

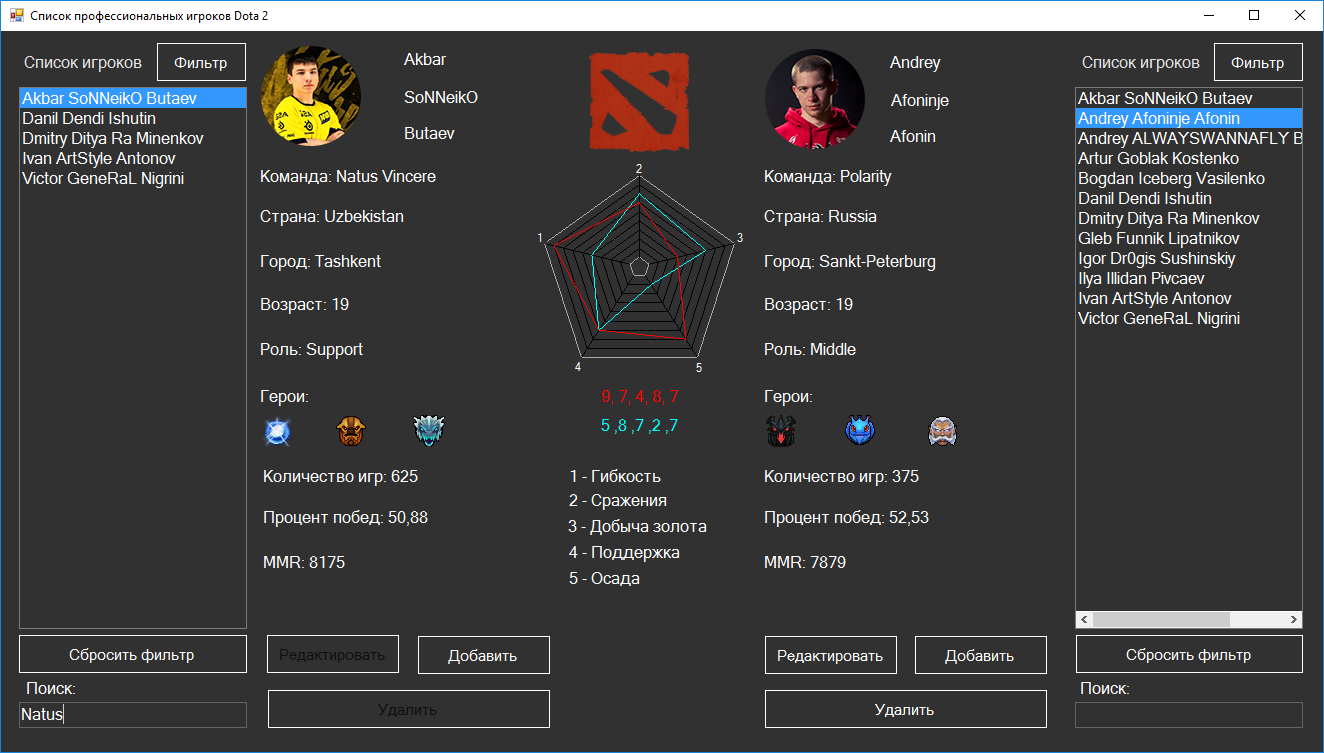


Рисунок 1.10 – Результат поиска.

Нельзя не заметить на рисунке 1.10 две кнопки, относящиеся к фильтрации, это кнопка “Фильтр” и “Очистить фильтр”. При нажатии кнопки “Фильтр”, появяться поля для выбора и 2 дополнительные кнопки, если с кнопкой “Применить”, всё очень интуитивно понятно, то вот кнопка “По профилю” требует разъяснений (кнопка “По профилю” выбирает параметры фильтрации схожие с тем что указано в профиле) (рис. 1.11) (для применения выбранных параметров требуется нажать на кнопку “Применить”) (см. рис.1.11), при нажатии на “Очистить фильтр” все выбранные параметры фильтрации сбросятся. Стоит упомянуть что фильтрация и поиск работают одновременно.

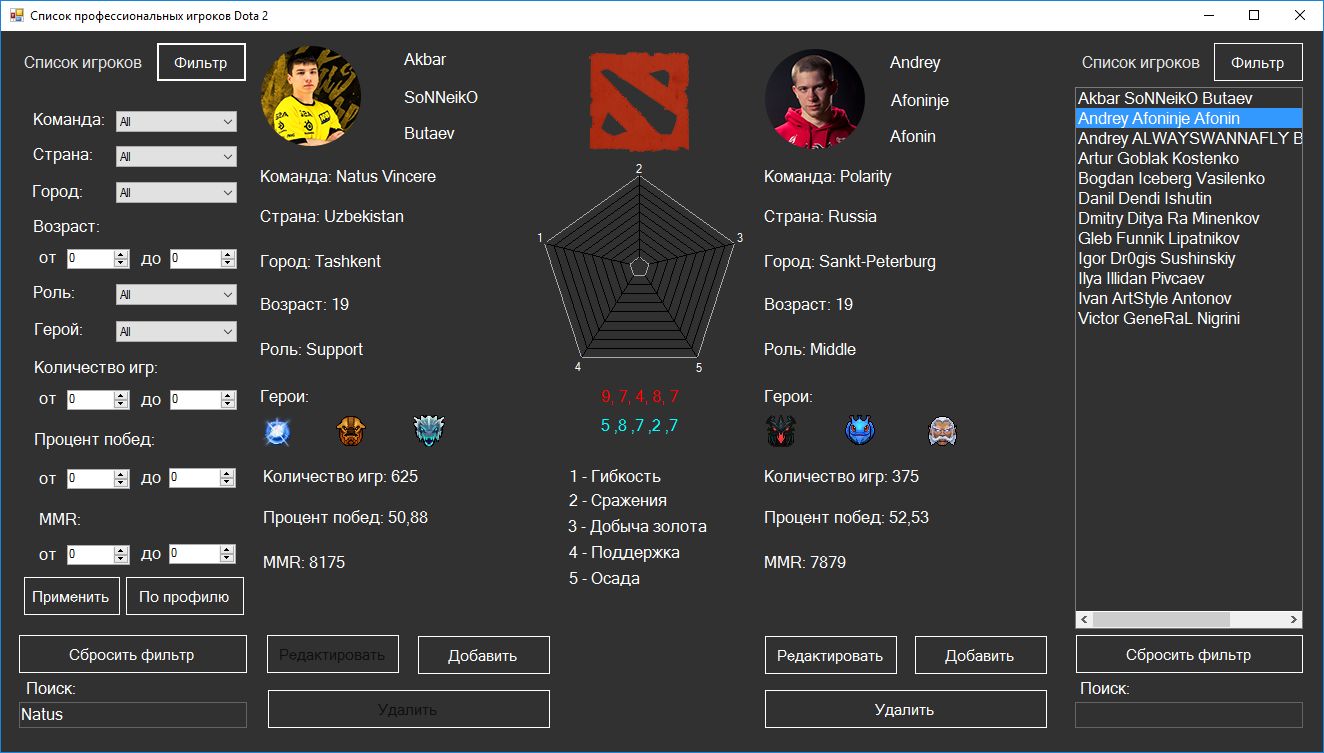


Рисунок 1.11 – Выбор параметров фильтрации.

Вернёмся обратно к рисунку 1.8, видим такие кнопки как “Редактировать”, “Удалить”, “Добавить”. При этом слева кнопки “Редактировать”, “Удалить” неактивны, это значит, что выбранного игрока редактировать запрещено. В отличии от аналогичных кнопок справа. При нажатии кнопки “Редактировать” или “Добавить” появиться поля для ввода и выбора данных. Так же мы видим появление ещё двух кнопок “Отправить” и “Отменить”, название сами за себя говорят (рис. 1.12).

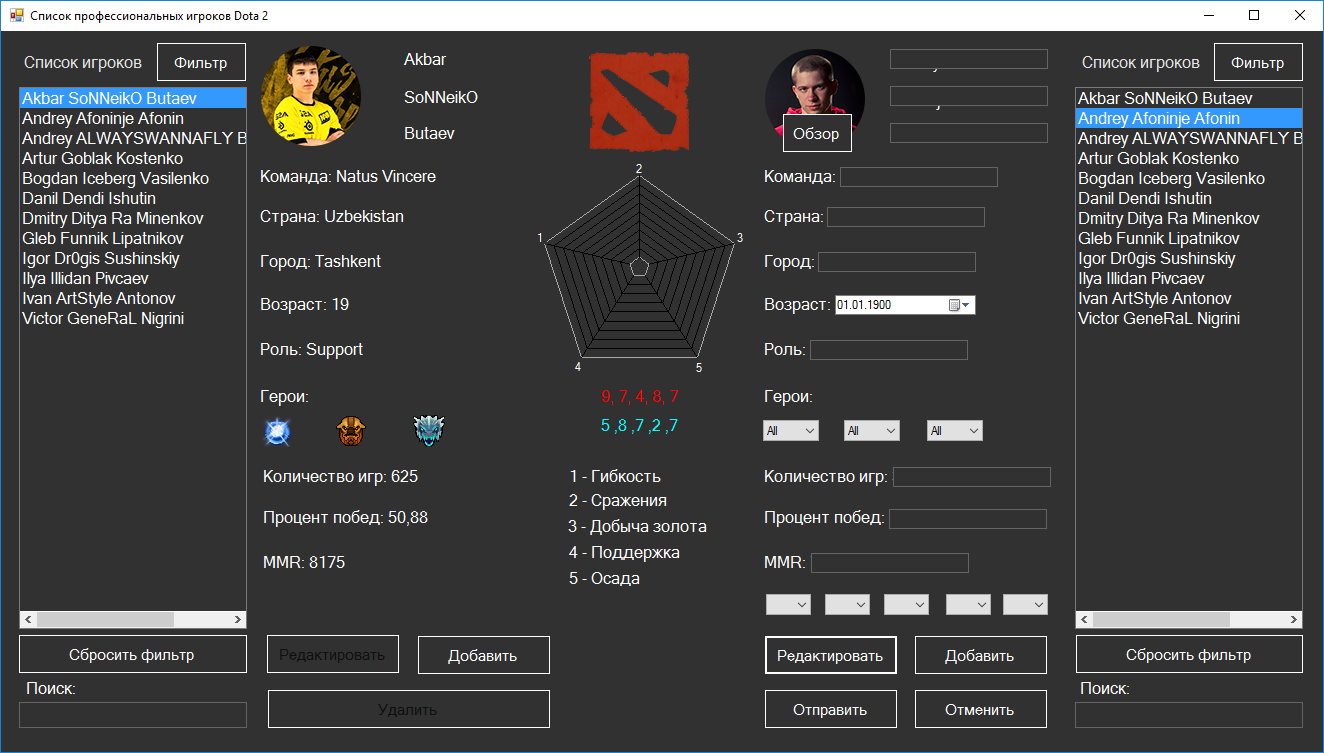


Рисунок 1.12 – Поля для редактирования.

### Форма “Мой профиль”.

Форма “Мой профиль” (см. рис.1.13). На данной форме можно увидеть данные о пользователе, которые он указал, и конечно же может изменять, нажав на кнопку “Редактировать”. Редактирование происходит аналогичным способом, который описывался на форме списка игроков (см. рис.1.14).

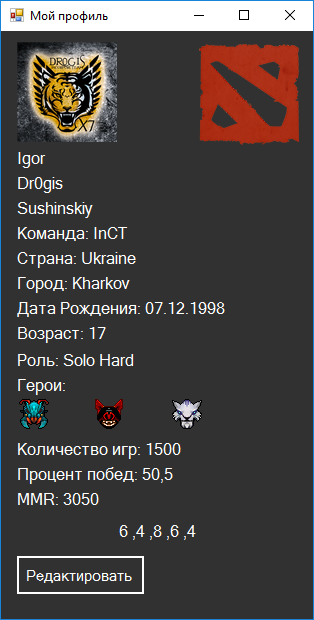


Рисунок 1.13 – Форма “Мой профиль”

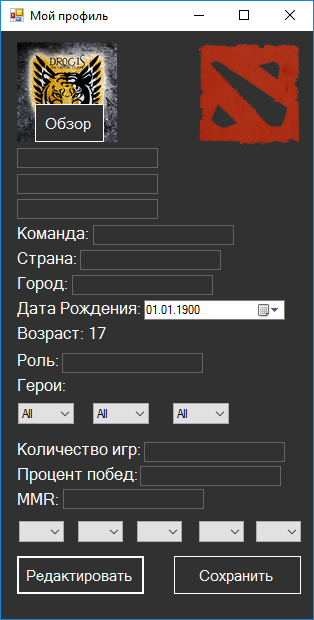


Рисунок 1.14 – Поля для редактирования профиля

Информацию о программе (см. рис.1.15) можно узнать, при помощи нажатия функциональной клавиши F1.

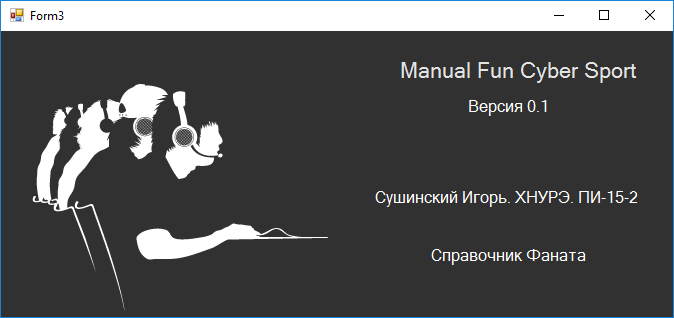


Рисунок 1.15 – Окно «О программе»

# ПРОЕКТНАЯ СПЕЦИФИКАЦИЯ

## Общая архитектура программы.

В ООП главным элементом является класс, в состав которого входит масса объектов с одинаковыми свойствами, операциями и отношениями.

В проекте есть 5 основных классов. В первом, «GlobalVariables» находятся переменные и методы которые требуется вызывать неоднократно и из любой точки программы. В данном классе есть методы, которые открывают сообщение с ошибкой, и т.д. Метод «DrawingPentagonPlayer» (см. рис 2.1) отрисовывает пятиугольник на форме. Код метода:

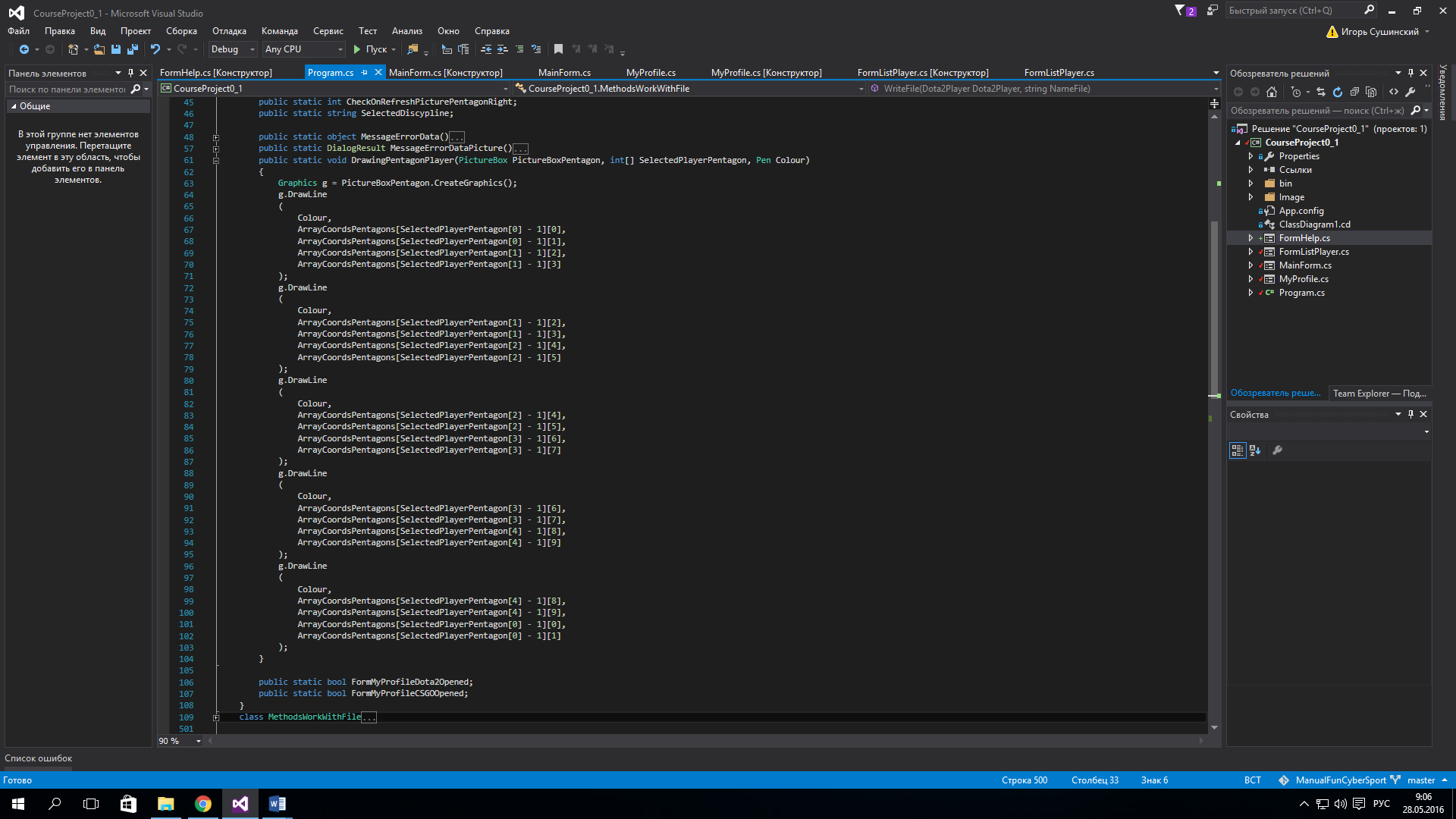


Рисунок 2.1 – Метод «DrawingPentagonPlayer»

Класс «MethodsWorkWithFile» содержит методы для работы с файлами – считывание данных из файла, создание списков на основе считаных данных, запись в файл, удаление из файла, редактирование файла и синхронизация.

«Player» - абстрактный класс, который хранит данные о игроках. От класса «Player» наследуются 2 класса “Dota2Player” и “CSGOPlayer”, которые хранят в себе данные игроков соответствующих дисциплин.

Архитектуру программы можно представить в виде схемы (см. рис.2.3):

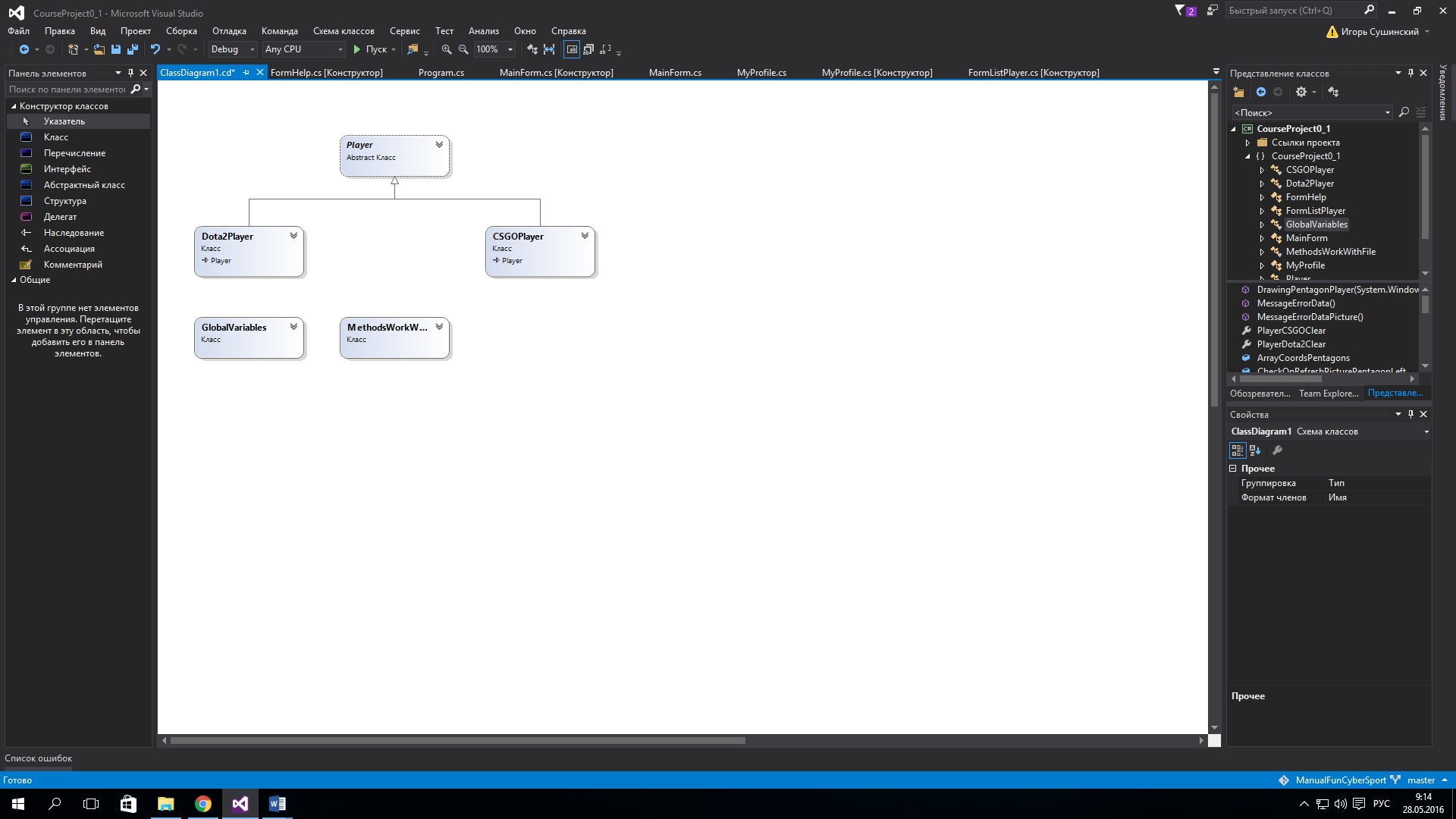


Рис 2.3 – Общая архитектура программы

## Пользовательский интерфейс.

Проект содержит 4 формы (см. рис.2.4) «MainForm.cs» является главной формой проекта, с помощью которой можно перейти к остальным. В этой форме отображается топ-10 и форма для прогнозирования игры, и кнопки для перехода на остальные формы. «FormListPlayer.cs» - форма, которая открывается при выборе дисциплины. В форме есть список игроков, кнопки редактирования, добавления, фильтрации и текстовое поле для поиска. «MyProfile.cs» служит для отображения и редактирования профиля пользователя. А также «FormHelp.cs» - форма вывода информации о программе. Все данные формы являются производными от класса «Form.cs».

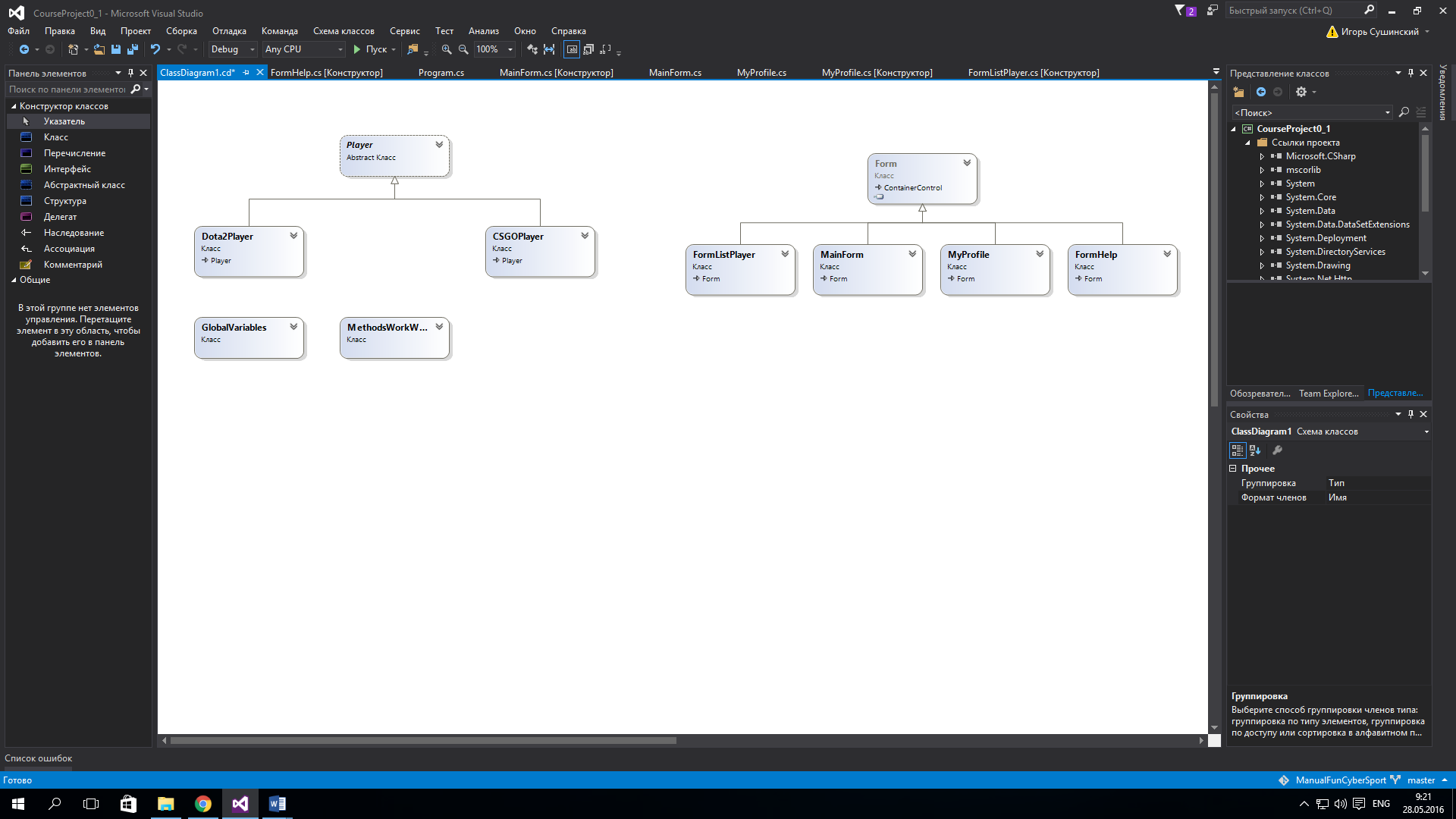


Рисунок 2.4 – Классы форм.

## Формат данных.

Данные загружаются из текстового файла в программу при помощи метода, который считывает файл построчно. В программе они отображаются в списке. Программа работает с данными обращаясь сначала к списку данных, они записаны в объекты (каждый объект имеет свои характеристики), далее к выбранному игроку, позволяя редактировать или удалить его.

При изменении данных игроков, для сохранения данных в файл используется метод, который записывает информацию о деталях в текстовой файл.

В программе данные о деталях находятся в классе «Player.cs», который содержит такие свойства (см. рис.2.5):

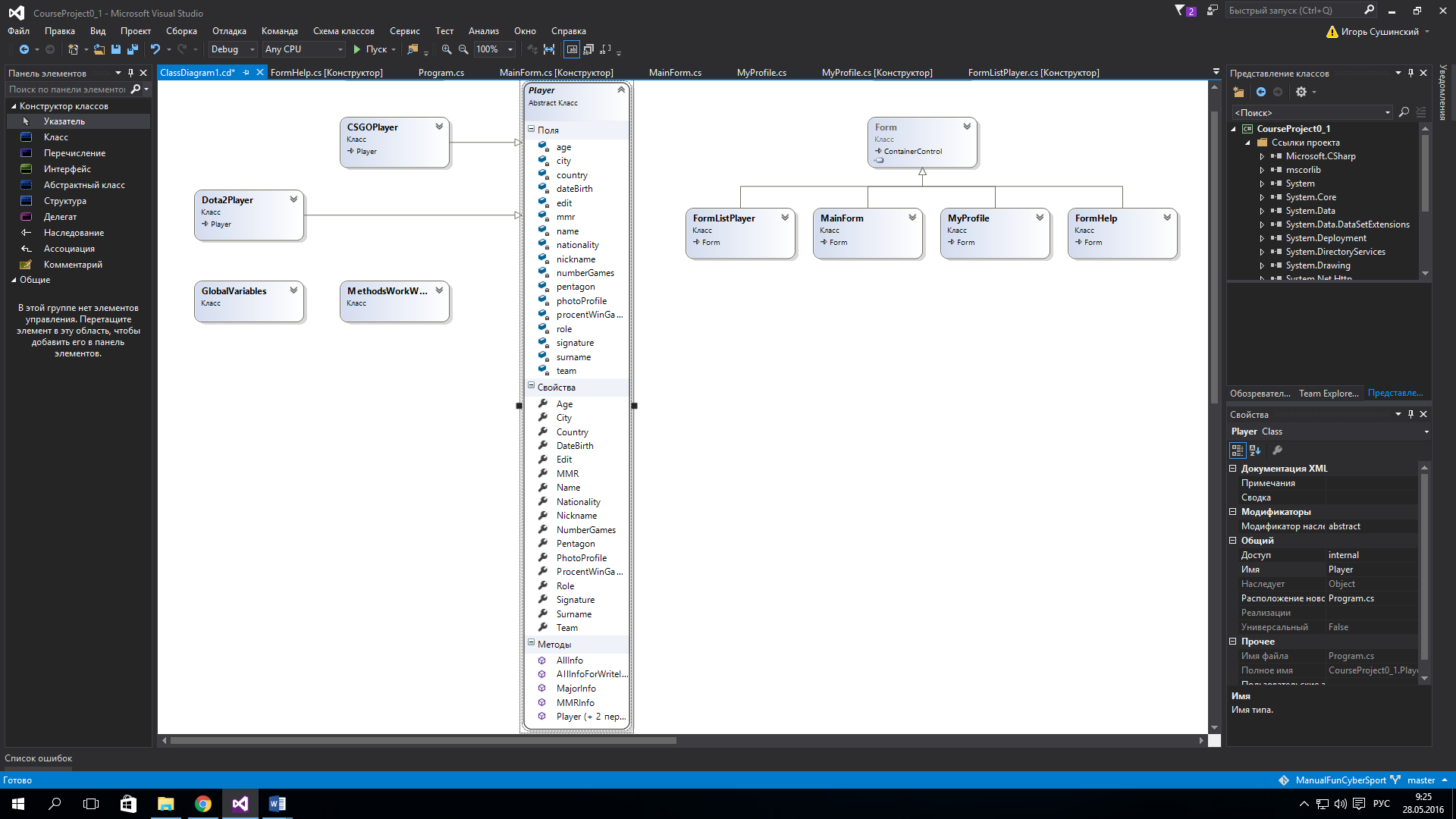


Рисунок 2.5 – Класс “Player”

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

В данном разделе находится информация об особенностях программы, ее установке, применению и использованию. Программа демонстрирует работу с данными, их отображением и работой с ними.

Для установки программы необходимо запустить файл «CourseProject0\_1.sln», и скомпилировать.

Для корректной работы программы необходимо наличие Microsoft .NET Framework версии 4.5 или выше.

На главной странице пользователь выбирает дисциплину, которая ему интересна, и работает с данными по определённой дисциплине.

Для синхронизации данных из файла нужно выбрать «Синхронизация», указать путь файла и имя файла. Произойдёт синхронизация основного файла с выбранным.

Чтобы выйти из программы, нужно закрыть главную форму.

Добавления нового игрока осуществляется кнопкой «Добавить», и заполнением тех полей, которые нужно изменить. Редактирование осуществляется кнопкой «Редактировать».

Удаление осуществляется кнопкой «Удалить».

Поиск по параметрам осуществляется вводом искомых данных в тестовое поле поиска. Также можно осуществить фильтрацию по полю, с помощью кнопки «Фильтр».

Чтобы узнать информацию о программе, нажать на функциональною клавишу F1.

ВЫВОДЫ

Результатом выполнения данной курсовой работы стала программа «Справочник фаната». Данная программа предназначена для удобного поиска данных о игроках, сравнение игроков, прогнозирование результата матча.

В проекте представлен объектно-ориентированный подход в проектировании программного обеспечения информационного плана, дающий возможность на ранних этапах разработки учесть все нюансы будущей программы, необходимый набор функций, состав и структуру баз данных, что в дальнейшем исключает необходимость переработки уже написанных компонентов программы.

В программе присутствует простой и понятный пользовательский интерфейс, так что полностью разобраться во функционале программы не составит труда даже для неопытного пользователя.

Планы на будущее: максимально расширить базу, осуществить синхронизацию с сервером, и добавить новые и интересные возможности.

ПЕРЕЧЕНЬ ССЫЛОК

1. Бондарев, В.М. Объектно-ориентированное программирование на С# [Текст]: учеб. пособ. /В.М. Бондарев. – Х.: Компания СМИТ, 2009 – 224 с.
2. Буч, Г. Объектно-ориентированный анализ и проектирование [Текст]: пер. с англ – М.: 000 "И.Д. Вильяме", 2008–720 с.
3. Мандер, Т. Разработка пользовательского интерфейса [Текст]: пер. с англ. – М.: ДМК Пресс, 2008 – 412 с.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5, 6-е изд. [Текст]: пер. с англ. – М.: Вильямс, 2013 – 1312 с.
5. Шилдт, Г. C# 4.0.: Полное руководство [Текст]: пер. с англ. – М.: Вильямс, 2011 – 1056 с.
6. Microsoft Software Developer Network [Электронный ресурс]; URL: https://msdn.microsoft.com/ru-ru/default.aspx

Приложение А. Частичный код программы.

abstract class Player

{

private string name;

private string nickname;

private string surname;

private string team;

private string country;

private string city;

private string nationality;

private DateTime dateBirth;

private int age;

private string role;

private string[] signature;

private int numberGames;

private double procentWinGames;

private string mmr;

private int[] pentagon;

private string photoProfile;

private bool edit;

public Player(object[] ArrayFilds)

{

Name = (string)ArrayFilds[0];

Nickname = (string)ArrayFilds[1];

Surname = (string)ArrayFilds[2];

Team = (string)ArrayFilds[3];

Country = (string)ArrayFilds[4];

City = (string)ArrayFilds[5];

Nationality = (string)ArrayFilds[6];

string[] tempStringDateBirth = ((string)ArrayFilds[7]).Split('.');

int tempYear = Convert.ToInt32(tempStringDateBirth[2]);

int tempMonth = Convert.ToInt32(tempStringDateBirth[1]);

int tempDay = Convert.ToInt32(tempStringDateBirth[0]);

DateBirth = new DateTime(tempYear, tempMonth, tempDay);

Age = DateTime.Today.Year - DateBirth.Year;

if (DateBirth > DateTime.Today.AddYears(-Age))

{

Age--;

}

Role = (string)ArrayFilds[8];

Signature = (string[])ArrayFilds[9];

NumberGames = Convert.ToInt32((string)ArrayFilds[10]);

ProcentWinGames = Convert.ToDouble((string)ArrayFilds[11]);

MMR = (string)ArrayFilds[12];

Pentagon = (int[])ArrayFilds[13];

PhotoProfile = (string)ArrayFilds[14];

Edit = Convert.ToBoolean(ArrayFilds[15]);

}

public Player(string name, string nickname, string surname, string team)

{

Name = name;

Nickname = nickname;

Surname = surname;

Team = team;

}

public Player(Player player)

{

Name = String.Copy(player.Name);

Nickname = String.Copy(player.Nickname);

Surname = String.Copy(player.Surname);

Team = String.Copy(player.Team);

Country = String.Copy(player.Country);

City = String.Copy(player.City);

Nationality = String.Copy(player.Nationality);

DateBirth = player.DateBirth;

Age = player.Age;

Role = String.Copy(player.Role);

Array.Copy(player.signature, signature = new string[player.signature.Length], player.signature.Length);

NumberGames = player.NumberGames;

ProcentWinGames = player.ProcentWinGames;

MMR = String.Copy(player.MMR);

Array.Copy(player.Pentagon, Pentagon = new int[player.Pentagon.Length], player.Pentagon.Length);

PhotoProfile = String.Copy(player.PhotoProfile);

}

public string Name

{

get

{

return name;

}

set

{

name = value;

}

}

public string Nickname

{

get

{

return nickname;

}

set

{

nickname = value;

}

}

public string Surname

{

get

{

return surname;

}

set

{

surname = value;

}

}

public string Team

{

get

{

return team;

}

set

{

team = value;

}

}

public string Country

{

get

{

return country;

}

set

{

country = value;

}

}

public string City

{

get

{

return city;

}

set

{

city = value;

}

}

public string Nationality

{

get

{

return nationality;

}

set

{

nationality = value;

}

}

public DateTime DateBirth

{

get

{

return dateBirth;

}

set

{

dateBirth = value;

}

}

public int Age

{

get

{

return age;

}

set

{

age = value;

}

}

public string Role

{

get

{

return role;

}

set

{

role = value;

}

}

public string[] Signature

{

get

{

return signature;

}

set

{

signature = value;

}

}

public int NumberGames

{

get

{

return numberGames;

}

set

{

numberGames = value;

}

}

public double ProcentWinGames

{

get

{

return procentWinGames;

}

set

{

procentWinGames = value;

}

}

public string MMR

{

get

{

return mmr;

}

set

{

mmr = value;

}

}

public int[] Pentagon

{

get

{

return pentagon;

}

set

{

pentagon = value;

}

}

public string PhotoProfile

{

get

{

return photoProfile;

}

set

{

photoProfile = value;

}

}

public bool Edit

{

get

{

return edit;

}

set

{

edit = value;

}

}

public string MajorInfo()

{

string rez = "";

rez += Name + ' ';

rez += Nickname + ' ';

rez += Surname;

return rez;

}

public string AllInfo()

{

string rez = "";

rez += Name + ' ';

rez += Nickname + ' ';

rez += Surname + ' ';

rez += Team + ' ';

rez += Country + ' ';

rez += City + ' ';

rez += Nationality + ' ';

rez += DateBirth.ToString() + ' ';

rez += Age;

rez += ' ';

rez += Role + ' ';

foreach (string element in Signature)

{

rez += element + ' ';

}

rez += NumberGames;

rez += ' ';

rez += ProcentWinGames;

rez += ' ';

rez += MMR + ' ';

rez += ' ';

foreach (int element in Pentagon)

{

rez += element;

rez += ' ';

}

rez += PhotoProfile;

return rez;

}

public string MMRInfo()

{

string rez = "";

rez += MMR + ' ';

rez += Name + ' ';

rez += Nickname + ' ';

rez += Surname;

return rez;

}

public string AllInfoForWriteInFile()

{

string rez = "";

rez += "\t";

rez += '\*';

rez += "\r\n";

rez += "\t\t";

rez += "Name = ";

rez += Name;

rez += "\r\n";

rez += "\t\t";

rez += "Nickname = ";

rez += Nickname;

rez += "\r\n";

rez += "\t\t";

rez += "Surname = ";

rez += Surname;

rez += "\r\n";

rez += "\t\t";

rez += "Team = ";

rez += Team;

rez += "\r\n";

rez += "\t\t";

rez += "Country = ";

rez += Country;

rez += "\r\n";

rez += "\t\t";

rez += "City = ";

rez += City;

rez += "\r\n";

rez += "\t\t";

rez += "Nationality = ";

rez += Nationality;

rez += "\r\n";

rez += "\t\t";

rez += "DateBirth = ";

rez += ((DateBirth.Day < 10) ? "0" + DateBirth.Day + "." : DateBirth.Day + ".") + ((DateBirth.Month < 10) ? "0" + DateBirth.Month + "." : DateBirth.Month + ".") + DateBirth.Year;

rez += "\r\n";

rez += "\t\t";

rez += "Role = ";

rez += Role;

rez += "\r\n";

rez += "\t\t";

rez += "Signature = [";

rez += string.Join(", ", Signature);

rez += ']';

rez += "\r\n";

rez += "\t\t";

rez += "NumberGames = ";

rez += NumberGames;

rez += "\r\n";

rez += "\t\t";

rez += "ProcentWinGames = ";

rez += ProcentWinGames;

rez += "\r\n";

rez += "\t\t";

rez += "MMR = ";

rez += MMR;

rez += "\r\n";

rez += "\t\t";

rez += "Pentagon = [";

rez += string.Join(", ", Pentagon);

rez += ']';

rez += "\r\n";

rez += "\t\t";

rez += "PhotoProfile = ";

rez += PhotoProfile;

rez += "\r\n";

rez += "\t\t";

rez += "Edit = ";

rez += Edit;

return rez;

}

}

class MethodsWorkWithFile

{

public static string[] ReadAllLinesFile(string NameFile)

{

return File.ReadAllLines(Environment.CurrentDirectory + @"\textfiles\" + NameFile);

}

public static string ReadAllTextFile(string NameFile)

{

return File.ReadAllText(Environment.CurrentDirectory + @"\textfiles\" + NameFile);

}

public static Player CreateObjectPlayer(string[] LinesFile, int IndexStart, int IndexEnd, string discypline)

{

string[] NameField = new string[]

{

"Name",

"Nickname",

"Surname",

"Team",

"Country",

"City",

"Nationality",

"DateBirth",

"Role",

"Signature",

"NumberGames",

"ProcentWinGames",

"MMR",

"Pentagon",

"PhotoProfile",

"Edit",

};

object[] ValueFild = new object[NameField.Length];

string tempString;

string[] tempStringArray;

int[] tempIntArray;

int templastIndex;

int tempIndexNameField = 0;

for (int i = IndexStart; i < IndexEnd; i++)

{

switch (NameField[tempIndexNameField])

{

case "Signature":

tempString = LinesFile[i].Substring(LinesFile[i].IndexOf(NameField[tempIndexNameField]) + NameField[tempIndexNameField].Length + 3);

if (tempString == "" || tempString[0] != '[' || tempString[tempString.Length - 1] != ']')

{

return (Player)GlobalVariables.MessageErrorData();

}

tempStringArray = tempString.Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

try

{

tempStringArray[0] = tempStringArray[0].Substring(1);

}

catch (IndexOutOfRangeException e)

{

return (Player)GlobalVariables.MessageErrorData();

}

templastIndex = tempStringArray.Length - 1;

tempStringArray[templastIndex] = tempStringArray[templastIndex].Substring(0, tempStringArray[templastIndex].Length - 1);

ValueFild[tempIndexNameField] = tempStringArray;

break;

case "Pentagon":

tempString = LinesFile[i].Substring(LinesFile[i].IndexOf(NameField[tempIndexNameField]) + NameField[tempIndexNameField].Length + 3);

tempStringArray = tempString.Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

tempStringArray[0] = tempStringArray[0].Substring(1);

templastIndex = tempStringArray.Length - 1;

tempStringArray[templastIndex] = tempStringArray[templastIndex].Substring(0, tempStringArray[templastIndex].Length - 1);

tempIntArray = new int[tempStringArray.Length];

int j = 0;

foreach (string element in tempStringArray)

{

tempIntArray[j] = Convert.ToInt32(element);

j++;

}

ValueFild[tempIndexNameField] = tempIntArray;

break;

default:

ValueFild[tempIndexNameField] = LinesFile[i].Substring(LinesFile[i].IndexOf(NameField[tempIndexNameField]) + NameField[tempIndexNameField].Length + 3);

break;

}

tempIndexNameField++;

}

Player tempPlayer;

if (discypline == "Dota2")

{

tempPlayer = new Dota2Player(ValueFild);

}

else

{

tempPlayer = new CSGOPlayer(ValueFild);

}

return tempPlayer;

}

public static List<List<Player>> CreateListPlayer(string NameFile)

{

List<List<Player>> ListDiscyplineListPlayer = new List<List<Player>>();

List<Player> TempListPlayer = new List<Player>();

string[] LinesFile = ReadAllLinesFile(NameFile);

string discypline = "Dota2";

int IndexStart = 1;

int IndexEnd;

int i = 0;

foreach (string Line in LinesFile)

{

if (Line.IndexOf('\*') == 1)

{

IndexStart = i + 1;

IndexEnd = IndexStart + 16;

TempListPlayer.Add(CreateObjectPlayer(LinesFile, IndexStart, IndexEnd, discypline));

}

if (Line.IndexOf('@') == 0)

{

ListDiscyplineListPlayer.Add(TempListPlayer);

TempListPlayer = new List<Player>();

discypline = "CSGO";

}

if (i == LinesFile.Length - 1)

{

ListDiscyplineListPlayer.Add(TempListPlayer);

}

i++;

}

return ListDiscyplineListPlayer;

}

public static void AddLists(string country, List<string> listCountry)

{

if (listCountry.IndexOf(country) == -1)

{

listCountry.Add(country);

}

}

public static void CreateLists(List<List<Player>> ListPlayer)

{

List<string> ListCountry = new List<string>();

ListCountry.Add("All");

List<string> ListTeam = new List<string>();

ListTeam.Add("All");

List<string> ListCity = new List<string>();

ListCity.Add("All");

//List<string> ListAge = new List<string>();

List<string> ListRole = new List<string>();

ListRole.Add("All");

List<string> ListHero = new List<string>();

ListHero.Add("All");

foreach (List<Player> Discypline in ListPlayer)

{

foreach (Player Player in Discypline)

{

AddLists(Player.Country, ListCountry);

AddLists(Player.Team, ListTeam);

AddLists(Player.City, ListCity);

//AddLists(Convert.ToString(Player.Age), ListAge);

AddLists(Player.Role, ListRole);

}

}

string tempAgility = "Anti-Mage;Bloodseeker;Drow Ranger;Shadow Fiend;Juggernaut;Razor;Mirana;Venomancer;Morphling;Faceless Void;Phantom Lancer;Phantom Assassin;Vengeful Spirit;Viper;Riki;Clinkz;Sniper;Broodmother;Templar Assassin;Weaver;Luna;Spectre;Bounty Hunter;Nyx Assassin;Ursa;Meepo;Gyrocopter;Slark;Lone Druid;Medusa;Naga Siren;Terrorblade;Troll Warlord;Arc Warden;Ember Spirit";

string tempStrength = "Earthshaker;Axe;Sven;Pudge;Tiny;Sand King;Kunkka;Slardar;Beastmaster;Tidehunter;Dragon Knight;Wraith King;Clockwerk;Lifestealer;Omniknight;Night Stalker;Huskar;Doom;Alchemist;Spirit Breaker;Brewmaster;Lycan;Treant Protector;Chaos Knight;Io;Undying;Centaur Warrunner;Magnus;Timbersaw;Abaddon;Bristleback;Tusk;Elder Titan;Legion Commander;Earth Spirit;Phoenix";

string tempIntellect = "Crystal Maiden;Bane;Puck;Lich;Storm Spirit;Lion;Windranger;Witch Doctor;Zeus;Enigma;Lina;Necrophos;Shadow Shaman;Warlock;Tinker;Queen of Pain;Nature's Prophet;Death Prophet;Enchantress;Pugna;Jakiro;Dazzle;Chen;Leshrac;Silencer;Dark Seer;Ogre Magi;Batrider;Rubick;Ancient Apparition;Disruptor;Invoker;Keeper of the Light;Outworld Devourer;Skywrath Mage;Shadow Demon;Techies;Visage;Oracle;Winter Wyvern";

string[] teamArrAgility = tempAgility.Split(';');

string[] teamArrStrength = tempStrength.Split(';');

string[] teamArrIntellect = tempIntellect.Split(';');

ListHero.AddRange(teamArrAgility);

ListHero.AddRange(teamArrStrength);

ListHero.AddRange(teamArrIntellect);

GlobalVariables.ListCountry = ListCountry;

GlobalVariables.ListTeam = ListTeam;

GlobalVariables.ListCity = ListCity;

GlobalVariables.ListRole = ListRole;

GlobalVariables.ListHero = ListHero;

}

public static void WriteFile(Dota2Player Dota2Player, string NameFile)

{

string AllFile = ReadAllTextFile(NameFile);

string[] SplitOnDiscypline = AllFile.Split('@');

SplitOnDiscypline[0] += Dota2Player.AllInfoForWriteInFile();

using (StreamWriter sw = new StreamWriter(Environment.CurrentDirectory + @"\textfiles\" + NameFile, false))

{

sw.Write(SplitOnDiscypline[0]);

sw.Write("\r\n@");

sw.Write(SplitOnDiscypline[1]);

}

}

public static void WriteFile(CSGOPlayer CSGOPlayer, string NameFile)

{

string AllFile = ReadAllTextFile(NameFile);

string[] SplitOnDiscypline = AllFile.Split('@');

SplitOnDiscypline[1] += CSGOPlayer.AllInfoForWriteInFile();

using (StreamWriter sw = new StreamWriter(Environment.CurrentDirectory + @"\textfiles\" + NameFile, false))

{

sw.Write(SplitOnDiscypline[0]);

sw.Write("\r\n@");

sw.Write(SplitOnDiscypline[1]);

}

}

public static string[] RemovePlayerInFile(Player Player, string NameFile)

{

string[] AllLineFile = ReadAllLinesFile(NameFile);

string[] NewLineFie = new string[AllLineFile.Length - 15];

//int i = 0;

int indexStart = 0;

int indexEnd = 0;

int index = 0;

foreach (string Line in AllLineFile)

{

index = Line.IndexOf(Player.Nickname);

if (index == 13)

{

break;

}

indexStart++;

}

if (indexStart >= AllLineFile.Length)

{

return null;

}

indexStart -= 1;

indexEnd = indexStart + 16;

int j = 0;

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

{

NewLineFie[i] = AllLineFile[i];

j = i;

}

for (int i = indexEnd; i < AllLineFile.Length; i++)

{

NewLineFie[j] = AllLineFile[i];

j++;

}

using (StreamWriter sw = new StreamWriter(Environment.CurrentDirectory + @"\textfiles\" + NameFile, false))

{

int i = 0;

foreach (string Line in NewLineFie)

{

if (i == NewLineFie.Length - 1)

{

sw.Write(Line);

break;

}

sw.WriteLine(Line);

i++;

}

}

return NewLineFie;

}

public static void EditFile(Dota2Player PlayerOld, Dota2Player PlayerNew, string NameFile)

{

RemovePlayerInFile(PlayerOld, NameFile);

WriteFile(PlayerNew, NameFile);

}

public static void EditFile(CSGOPlayer PlayerOld, CSGOPlayer PlayerNew, string NameFile)

{

RemovePlayerInFile(PlayerOld, NameFile);

WriteFile(PlayerNew, NameFile);

}

public static Player[] ReadMyProfile()

{

Player[] MyProfiles = new Player[2];

string[] AllLinesFile = File.ReadAllLines(Environment.CurrentDirectory + @"\textfiles\MyProfile.txt");

string Dota2Name = AllLinesFile[2].Substring(9);

string Dota2Nickname = AllLinesFile[3].Substring(13);

string Dota2Surname = AllLinesFile[4].Substring(12);

string Dota2Team = AllLinesFile[5].Substring(9);

string Dota2Country = AllLinesFile[6].Substring(12);

string Dota2City = AllLinesFile[7].Substring(9);

string Dota2Nationality = AllLinesFile[8].Substring(16);

string Dota2DateBirth = AllLinesFile[9].Substring(14);

string Dota2Role = AllLinesFile[10].Substring(9);

string[] Dota2SignatureTemp = AllLinesFile[11].Substring(14).Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

Dota2SignatureTemp[0] = Dota2SignatureTemp[0].Substring(1);

int templastIndex = Dota2SignatureTemp.Length - 1;

Dota2SignatureTemp[templastIndex] = Dota2SignatureTemp[templastIndex].Substring(0, Dota2SignatureTemp[templastIndex].Length - 1);

string[] Dota2Signature = Dota2SignatureTemp;

string Dota2NumberGames = AllLinesFile[12].Substring(16);

string Dota2ProcentWinGames = AllLinesFile[13].Substring(20);

string Dota2MMR = AllLinesFile[14].Substring(8);

string[] Dota2PentagonTemp = AllLinesFile[15].Substring(13).Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

Dota2PentagonTemp[0] = Dota2PentagonTemp[0].Substring(1);

templastIndex = Dota2PentagonTemp.Length - 1;

Dota2PentagonTemp[templastIndex] = Dota2PentagonTemp[templastIndex].Substring(0, Dota2PentagonTemp[templastIndex].Length - 1);

int[] tempIntArray = new int[Dota2PentagonTemp.Length];

int j = 0;

foreach (string element in Dota2PentagonTemp)

{

tempIntArray[j] = Convert.ToInt32(element);

j++;

}

int[] Dota2Pentagon = tempIntArray;

string Dota2PhotoProfile = AllLinesFile[16].Substring(17);

string Dota2Edit = AllLinesFile[17].Substring(9);

MyProfiles[0] = new Dota2Player(new object[] { Dota2Name, Dota2Nickname, Dota2Surname, Dota2Team, Dota2Country, Dota2City, Dota2Nationality, Dota2DateBirth, Dota2Role, Dota2Signature, Dota2NumberGames, Dota2ProcentWinGames, Dota2MMR, Dota2Pentagon, Dota2PhotoProfile, Dota2Edit });

string CSGOName = AllLinesFile[21].Substring(9);

string CSGONickname = AllLinesFile[22].Substring(13);

string CSGOSurname = AllLinesFile[23].Substring(12);

string CSGOTeam = AllLinesFile[24].Substring(9);

string CSGOCountry = AllLinesFile[25].Substring(12);

string CSGOCity = AllLinesFile[26].Substring(9);

string CSGONationality = AllLinesFile[27].Substring(16);

string CSGODateBirth = AllLinesFile[28].Substring(14);

string CSGORole = AllLinesFile[29].Substring(9);

string[] CSGOSignatureTemp = AllLinesFile[30].Substring(14).Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

CSGOSignatureTemp[0] = CSGOSignatureTemp[0].Substring(1);

templastIndex = CSGOSignatureTemp.Length - 1;

CSGOSignatureTemp[templastIndex] = CSGOSignatureTemp[templastIndex].Substring(0, CSGOSignatureTemp[templastIndex].Length - 1);

string[] CSGOSignature = CSGOSignatureTemp;

string CSGONumberGames = AllLinesFile[31].Substring(16);

string CSGOProcentWinGames = AllLinesFile[32].Substring(20);

string CSGOMMR = AllLinesFile[33].Substring(8);

string[] CSGOPentagonTemp = AllLinesFile[34].Substring(13).Split(new string[] { ", " }, StringSplitOptions.RemoveEmptyEntries);

CSGOPentagonTemp[0] = CSGOPentagonTemp[0].Substring(1);

templastIndex = CSGOPentagonTemp.Length - 1;

CSGOPentagonTemp[templastIndex] = CSGOPentagonTemp[templastIndex].Substring(0, CSGOPentagonTemp[templastIndex].Length - 1);

tempIntArray = new int[CSGOPentagonTemp.Length];

j = 0;

foreach (string element in CSGOPentagonTemp)

{

tempIntArray[j] = Convert.ToInt32(element);

j++;

}

int[] CSGOPentagon = tempIntArray;

string CSGOPhotoProfile = AllLinesFile[35].Substring(17);

string CSGOEdit = AllLinesFile[36].Substring(9);

MyProfiles[1] = new CSGOPlayer(new object[] { CSGOName, CSGONickname, CSGOSurname, CSGOTeam, CSGOCountry, CSGOCity, CSGONationality, CSGODateBirth, CSGORole, CSGOSignature, CSGONumberGames, CSGOProcentWinGames, CSGOMMR, CSGOPentagon, CSGOPhotoProfile, CSGOEdit });

return MyProfiles;

}

public static void SearchPlayer(List<Player> ListPlayer, Player Player, string Discypline)

{

foreach (Player player in ListPlayer)

{

if (player.Nickname == Player.Nickname)

{

if (Discypline == "Dota2")

{

EditFile((Dota2Player)player, (Dota2Player)Player, "ListPlayer.txt");

return;

}

else

{

EditFile((CSGOPlayer)(Player)player, (CSGOPlayer)(Player)Player, "ListPlayer.txt");

return;

}

}

}

if (Discypline == "Dota2")

{

WriteFile((Dota2Player)Player, "ListPlayer.txt");

return;

}

else

{

WriteFile((CSGOPlayer)(Player)Player, "ListPlayer.txt");

return;

}

}

public static void Sinhronizacia(List<List<Player>> ListPlayerMain, List<List<Player>> ListPlayerSelected)

{

foreach (Dota2Player player in ListPlayerSelected[0])

{

SearchPlayer(ListPlayerMain[0], player, "Dota2");

}

foreach (CSGOPlayer player in ListPlayerSelected[1])

{

SearchPlayer(ListPlayerMain[1], player, "CSGO");

}

}

}