НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА АТОМАТИЗОВАНИХ СИСТЕМ ОБРОБКИ ІНФОРМАЦІЇ ТА УПРАВЛІННЯ

**КУРСОВА РОБОТА**

з дисципліни Програмування - 2. Структури даних та алгоритми

на тему: «Клуб футбольних фанатів. Журнал команд та матчів»

Студента 1 курсу ІС-93 групи

напряму підготовки 126

спеціальності Інформаційні системи та технології

Муліка Р. Ю.

Керівник

асистент Лебідь Сергій \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Олександрович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна оцінка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів:\_\_\_\_\_\_\_\_\_\_

Оцінка ECTS:\_\_\_\_\_\_\_\_\_\_\_

Члени комісії \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)

Київ – 2020

**Національний технічний університет України “КПІ ім. Сікорського”**

(назва вищого навчального закладу)

**Кафедра автоматизованих систем обробки інформації і управління**

Дисципліна Структури даних та алгоритми

**Спеціальність 126 «Інформаційні системи та технології»**

Курс 1 Група ІС-93 Семестр 2

**ЗАВДАННЯ**

**на курсову роботу студента** Муліка Рустама Юрійовича

(прізвище, ім’я, по батькові)

1. Тема проекту (роботи)
2. Строк здачі студентом закінченого проекту (роботи)
3. Вихідні дані до проекту (роботи)
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
5. Перелік графічного матеріалу ( з точним зазначенням обов’язкових креслень )
6. Дата видачі завдання

ЗМІСТ

[ВСТУП 3](#_Toc42054255)

[ОСНОВНА ЧАСТИНА 4](#_Toc42054256)

[1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 4](#_Toc42054257)

[1.1 Постановка задачі 4](#_Toc42054258)

[1.2 Структура вхідних та вихідних даних 4](#_Toc42054259)

[1.3 Операції і методи 5](#_Toc42054260)

[1.4 Параметри і функціональні можливості 6](#_Toc42054261)

[2. КОДУВАННЯ 8](#_Toc42054262)

[2.1 Робоче середовище 8](#_Toc42054263)

[2.2 Опис програмного забезпечення 8](#_Toc42054264)

[3. ТЕСТУВАННЯ 11](#_Toc42054265)

[3.1 Методика тестування 11](#_Toc42054266)

[3.2 План тестування 11](#_Toc42054267)

[3.3 Виконання тестування 12](#_Toc42054268)

[ВИСНОВКИ 15](#_Toc42054269)

[ДОДАТКИ 16](#_Toc42054270)

# ВСТУП

Об’єктно-орієнтоване програмування (ООП) – це напрямок програмування, в якій об’єкти програми пов’язані та спроможні взаємодіяти один з одним. Об’єкт є незалежним та здатний отримувати, обробляти та відсилати дані іншим об’єктам. В ООП існує три основних механізми: інкапсуляція, успадкування та поліморфізм.

Інкапсуляція – надання користувачу інтерфейсу для використання об’єкта, при цьому приховуючи від нього його деталі реалізації.

Успадкування – можливість створення об’єкту на базі іншого, надаючи новому об’єкту доступ до функціональності початкового елемента (передача методів та властивостей від базового класу в клас, що успадковує.

Поліморфізм – можливість базового об’єкту використовувати методи похідного класу, який ще не був створений на момент створення базового.

Об’єкт – це структура зі здатністю зберігати свою інформацію та змінювати її.

Клас – це основний елемент , в рамках якого проводиться будування програм. Він може зберігати в собі дані та код який може змінювати ці дані. Частіше всього класс описує реальні об’єкти.

Мови програмування C# та C++, як і багато інших мов програмування, підтримують вищезазначені механізми. Вони обидві є поширеними та підтримують синтаксичну типізацію, поліморфізм, перевантаження операторів, ітератори, властивості, методи, атрибути та виключення.

Головна мета даної роботи – дослідити принципи ООП та закріпити вивчені на курсі знання.

# ОСНОВНА ЧАСТИНА

## 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

### 1.1 Постановка задачі

Завдання: Клуб футбольних фанатів. Журнал команд та матчів

Футбольним фанатам надаються реєстри футболістів та футбольних команд, історія змагань та анонси майбутніх матчів. Фанати можуть слідкувати як за окремими улюбленими футболістами, так і за цілими командами. Також цікавитись окремими сезонами ігор або історією певних видів матчів.

1.2 Структура вхідних та вихідних даних

Вихідні дані являють собою інформацію про дві футбольні команди: «Динамо» та «Шахтер», коротку інформацію про кожного з футболістів цих команд, історію всіх матчів за 2019 – 2020 рр. у яких брали участь ці команди та анонси майбутніх матчів у яких братимуть участь ці команди. Вихідні дані для програми зберігаються у текстових файлах формату .txt. Текстові файли зберігаються у спеціально створеній папці files, що розташована у кореневій папці проекту. Для списку гравців створено по окремій папці для кожної команди, де зберігаються всі файли з інформацією про гравців. Назва кожного файлу відповідає номеру на футболці гравця, що у подальшому полегшить ініціалізацію даних під час реалізації програми та їх використання для взаємодії з користувачем. (див. Додаток 1.2.1)

Вхідні дані являють собою код клавіш, що користувач натискає на клавіатурі для взаємодії з меню та дані, які користувач вводить змінюючи збережені у файлах дані у режимі адміністратора. Текстові файли з інформацією в свою чергу також являють собою вхідні дані, адже під час запуску програми відбувається ініціалізація тексту файлів у відповідні змінні.

Вхідні дані

* Символи клавіш
* Текст, введений з клавіатури
* Текст з текстових файлів для ініціалізації інформації.

Вихідні дані

* Інформація про команди «Динамо» та «Шахтар»
* Список гравців команд
* Інформація про кожного з гравців команд
* Історія матчів, у яких брала участь кожна з команд
* Анонси майбутніх матчів за участі команд

1.3 Операції і методи

У коді програми були реалізовані наступні операції та методи:

* Отримання інформації про команди «Динамо» та «Шахтар»
* Отримання списку гравців команд
* Отримання інформації про кожного з гравців команд
* Отримання історії матчів, у яких брала участь кожна з команд
* Отримання анонсів майбутніх матчів за участі команд
* Ініціалізація даних з зовнішніх текстових файлів
* Взаємодія з користувачем (меню)
* Форматований вивід інформації в консоль

1.4 Параметри і функціональні можливості

Програма надає інформацію про футбольні клуби та їх гравців, а саме містить у собі наступні параметри:

* Назви команд
* Інформація про команди
* Ім’я кожного з гравців команд
* Номер кожного з гравців команд
* Інформація про кожного з гравців команд
* Дати проведення командами кожного матчу
* Назви команд які брали участь у цих матчах
* Рахунок з яким гра була завершена
* Дати запланованих матчів
* Назви команд які братимуть участь у цих матчах

Основні функціональні можливості програми криються в режимі адміністратора. Увійшовши до клубу футбольних фанатів в якості адміністратора користувач має доступ до наступних функціональних можливостей:

* Введення паролю для ідентифікації адміністратора
* Редагування інформації про команди
* Редагування інформації про гравців
* Редагування дати проведення минулих матчів
* Редагування дати проведення запланованих матчів
* Редагування назв команд, які брали участь у минулих матчах
* Редагування назв команд, які братимуть участь у запланованих матчах
* Редагування рахунку, з яким були завершені минулі матчі
* Встановлення рахунку на матчах з дошки анонсів, якщо вони вже відбулись

Основний функціонал незареєстрованого користувача полягає у перегляді представленої інформації.

2. КОДУВАННЯ

2.1 Робоче середовище

Для розробки програми мною було використано робоче середовище Visual Studio 2019. Мова програмування – C#.

2.2 Опис програмного забезпечення

У своїй програмі я реалізував наступні класи:

* Player – клас, що інкапсулює гравця команди. Містить у собі поля з ім’ям гравця, поле з номером гравця та поле з інформацією про гравця. Також містить властивості для введення та отримання цих полів та метод для форматованого консольного виводу інформації про гравця. (див. Додаток 2.2.1)
* Team – клас, що інкапсулює команду. Містить у собі поле з однозв’язним списком всіх гравців команди, поле з назвою команди та поле з інформацією про команду. Також містить метод ініціалізації списку гравців, метод ініціалізації інформації про команду з текстового файлу, метод ініціалізації інформації про гравців з текстових файлів, метод отримання списку гравців, індексатор, що працює за ігровим номером гравця та метод форматованого виводу інформації про команду в консоль. (див. Додаток 2.2.2)
* Match – клас, що інкапсулює матч. Містить у собі поля зі змінними типу Team – команди що беруть участь у матчі та поле типу Date – дата проведення. Також містить властивості для введення та отримання цих полів. (див. Додаток 2.2.3)
  + Was – дочірній від Match клас, що інкапсулює минулий матч. Містить у собі поля з кількістю голів на рахунку кожної команди, метод форматованого виведення данних про матч та властивість для введення та отримання кількості голів. (див. Додаток 2.2.4)
  + Will - дочірній від Match клас, що інкапсулює майбутній матч. Містить у собі метод форматованого виведення данних про матч та конструктор. (див. Додаток 2.2.5)
* Board – клас, що інкапсулює дошку зі списком матчів. Містить у собі поле з однозв’язним списком змінних типу Match – список матчів та поле розміру списку. Також містить метод повернення розміру списку. (див. Додаток 2.2.6)
  + History - дочірній від Board клас, що інкапсулює дошку з історією матчів команди. Містить у собі метод ініціалізації списку минулих матчів. (див. Додаток 2.2.7)
  + Announcement - дочірній від Board клас, що інкапсулює дошку з анонсами матчів команди. Містить у собі метод ініціалізації списку майбутніх матчів. (див. Додаток 2.2.8)
* Text – клас, що інкапсулює текст інформації про команду або гравця. Містить у собі поля зі шляхом до текстових файлів, та статичні методи для редагування тексту, а саме метод введення нової строки там метод видалення строки за її номером. Також містить метод форматованого виводу тексту в консоль. (див. Додаток 2.2.9)
* Date – клас, що інкапсулює дату. Містить у собі поля з числовим представленням дня, місяця та року, метод форматованого виведення, а також конструктор, шо викликає виняток у разі невірного формату дати. (див. Додаток 2.2.10)
* LinkedList – клас, що реалізує однозв'язний список. Містить у собі властивість отримання розміру списку, метод додавання елементу у список та індексатор. (див. Додаток 2.2.11)
* LinkedListNode – клас-вузол однозв’язного списку. (див. Додаток 2.2.12)
* Program – основний клас програми, що містить у собі реалізацію взаємодії з користувачем через консоль. (див. Додаток 2.2.13)

3. ТЕСТУВАННЯ

3.1 Методика тестування

Метод тестування полягає у тому, щоб переглянути всі можливі сценарії та таким чином перевірити програму на працездатність та довести коректність реалізації коду.

3.2 План тестування

* Запуск програми. Перед користувачем повинна відкритись консоль, у якій буде зображено меню.
* Перегляд інструкції. Перед користувачем повинна відобразитись інструкція.
* Вхід в режимі гостя і перегляд інформації. Перед користувачем повинно відкритись меню вибору команди, далі меню вибору інформації для перегляду.
* Вхід в режимі адміністратора, введення паролю і редагування інформації. Перед користувачем повинно відкритись меню вибору команди, далі розширене меню з можливістю вибрати інформацію для перегляду або відредагувати її.
* Редагування інформації про команду. Користувач повинен мати змогу ввести новий рядок або видалити рядок за його номером.
* Редагування інформації про гравця за номером. Користувач повинен мати змогу ввести новий рядок або видалити рядок за його номером. При введенні невірного номеру гравця повинно виникнути повідомлення про помилку та зупинка програми. Наприклад, у жодній з команд немає гравця під номером 2.
* Редагування дошки з історією матчів. Користувач повинен мати змогу змінити дату, рахунок та назви команд що беруть участь у грі.
* Редагування дошки з анонсами матчів. Користувач повинен мати змогу змінити дату, встановити рахунок та змінити назви команд що беруть участь у грі.

3.3 Виконання тестування

Після запуску програми перед користувачем відкривається меню (див. Додаток 3.3.1)

Натиснувши 3 користувач має змогу переглянути інструкцію. (див. Додаток 3.3.2)

Повернувшись назад та увійшовши в режимі гостя користувач повинен обрати команду. (див. Додаток 3.3.3) Далі він має змогу:

* Переглянути список гравців. (див. Додаток 3.3.4)
* Переглянути інформацію про гравця за його номером. (див. Додаток 3.3.5)
* Переглянути інформацію про команду. (див. Додаток 3.3.6)
* Переглянути історію матчів команди за 2019 – 2020 роки. (див. Додаток 3.3.7)
* Переглянути новини про анонси матчів команди. (див. Додаток 3.3.8)

Якщо введено невірний номер гравця виникає повідомлення про помилку та програма зупиняє свою роботу. (див. Додаток 3.3.9)

Повернувшись назад та увійшовши в режимі адміністратора користувач повинен ввести пароль. Пароль – mulik. Якщо пароль введено невірно, рівно на 1 секунду виникає повідомлення про помилку та надається наступна спроба. Кількість спроб необмежена. (див. Додаток 3.3.10)

Після вірного введення паролю користувач повинен обрати команду. Далі він має змогу:

* Переглянути список гравців. (див. Додаток 3.3.4)
* Переглянути інформацію про гравця за його номером. (див. Додаток 3.3.5)
* Переглянути інформацію про команду. (див. Додаток 3.3.6)
* Переглянути історію матчів команди за 2019 – 2020 роки. (див. Додаток 3.3.7)
* Переглянути новини про анонси матчів команди. (див. Додаток 3.3.8)
* Редагувати інформацію про гравця за його номером. (див. Додаток 3.3.11)
* Редагувати інформацію про команду. (див. Додаток 3.3.12)
* Редагувати історію матчів команди за 2019 – 2020 роки. (див. Додаток 3.3.13)
* Редагувати новини про анонси матчів команди. (див. Додаток 3.3.14)

Додавання нового рядка до тексту відбувається успішно, так само як і видалення рядка. Після виходу з меню зміни зберігаються

Під час редагування історії матчів користувач повинен вибрати матч для редагування за номером рядку. Далі перед користувачем відкривається меню взаємодії, користуючись яким він має змогу: (див. Додаток 3.3.16)

* Змінити дату проведення гри. (див. Додаток 3.3.17)
* Змінити назву першої команди. (див. Додаток 3.3.18)
* Змінити назву другої команди. (див. Додаток 3.3.19)
* Змінити рахунок гри. (див. Додаток 3.3.20)

При введенні невірної дати (наприклад, 32 день місяця) у консолі виникає повідомлення про помилку та програма припиняє свою роботу. (див. Додаток 3.3.21).

ВИСНОВКИ

Виконуючи курсову роботу, мені вдалося реалізувати програму з використанням компонентів середовища програмування Visual Studio 2019 та мови програмування C#.

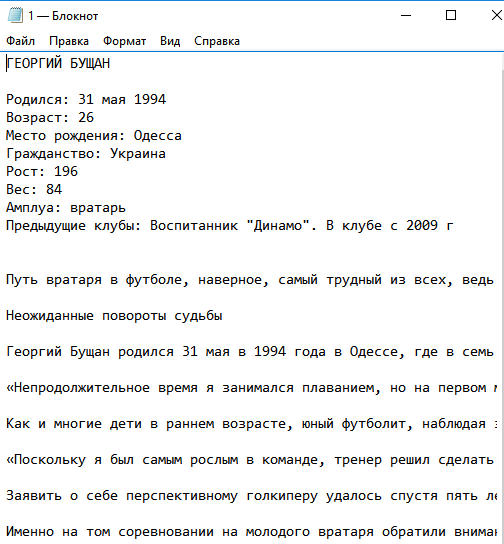
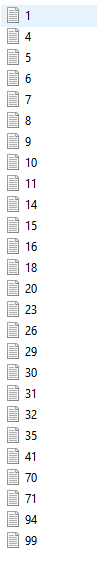
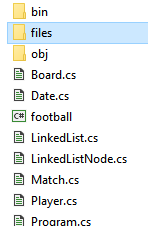
Виконання даної роботи було розділено на два етапи:

Теоретичний аналіз поставленої задачі, що включає в себе постанову задачі, відокремлення основних класів та базових класів, відокремлення основних методів та операцій необхідних для вирішення задачі та створення алгоритму завдання.

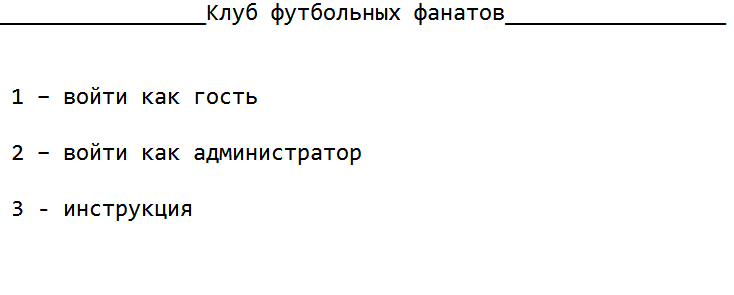
Програмна реалізація завдання за допомогою середовища програмування Visual Studio 2019, а саме реалізація класів та методів.

Особливо наголошую на головних результатах виконання курсової роботи, а саме закріплення знань в області створення алгоритмів, успадкування класів та базової інформації про поліморфізм та інкапсуляцію, навичок програмування на мові C# та створення класів, отриманих під час проходження курсу об’єктно-орієнтованого програмування.

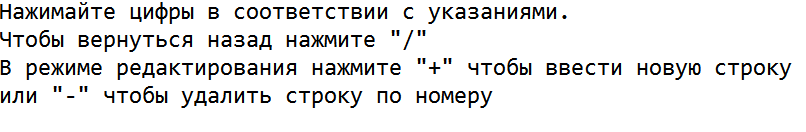
ДОДАТКИ

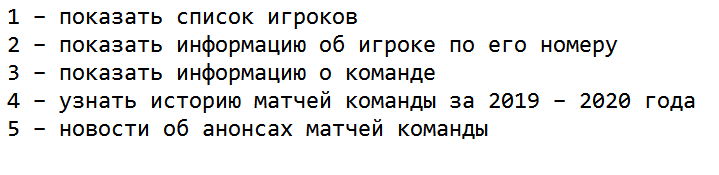
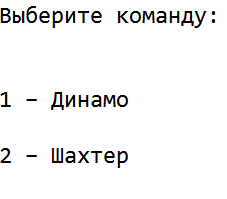
(Додаток 1.2.1)



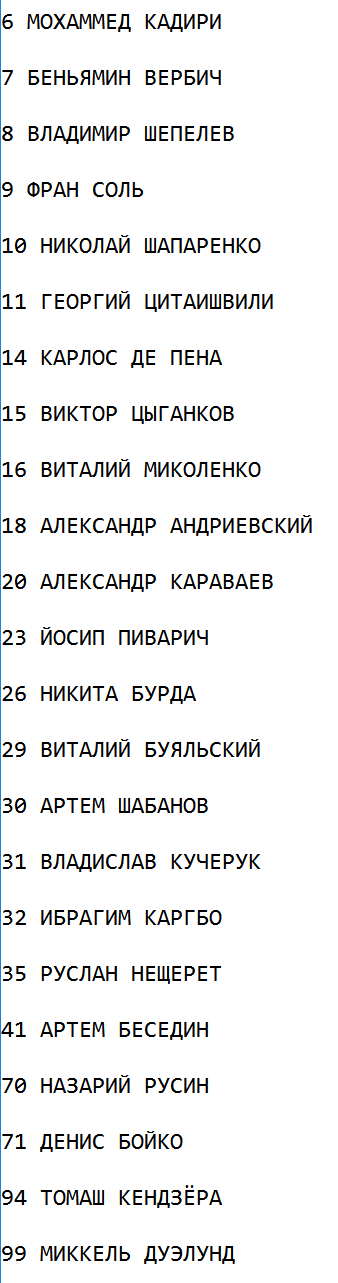
(Додаток 3.3.1)



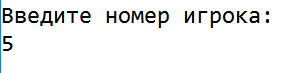
(Додаток 3.3.2)

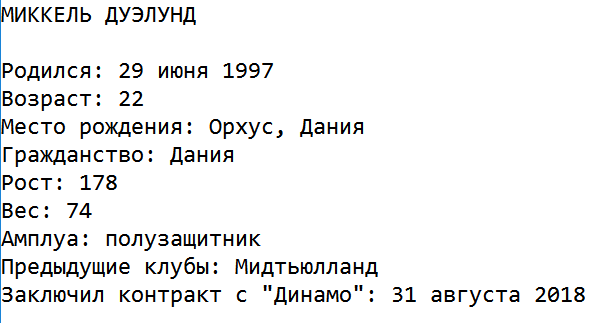


(Додаток 3.3.3)

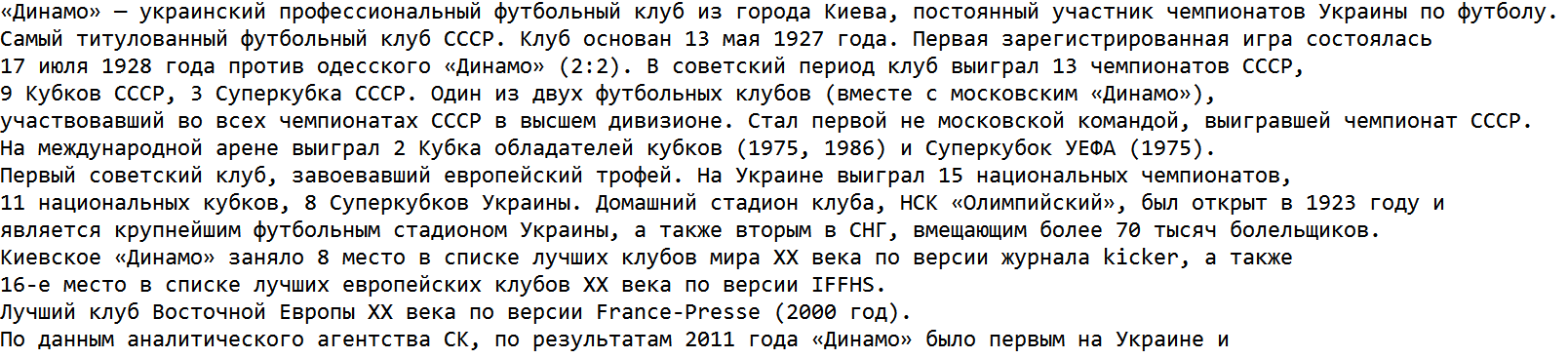


(Додаток 3.3.4)

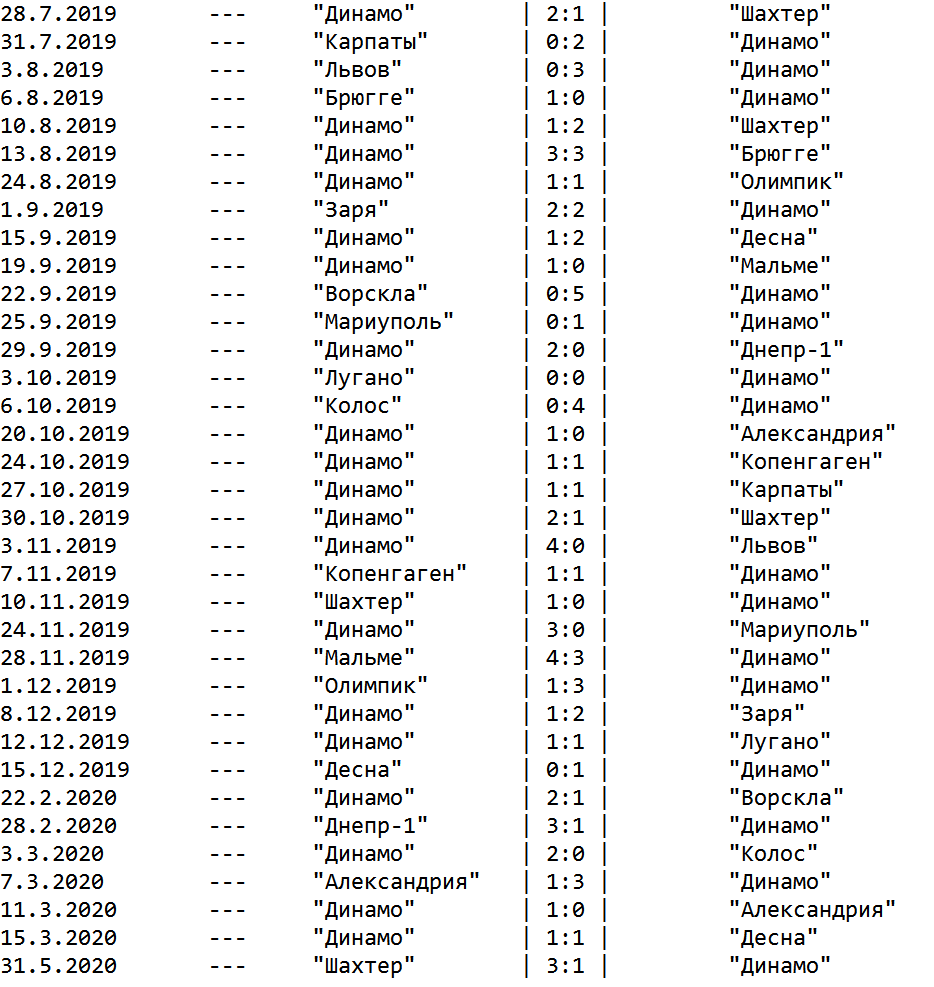




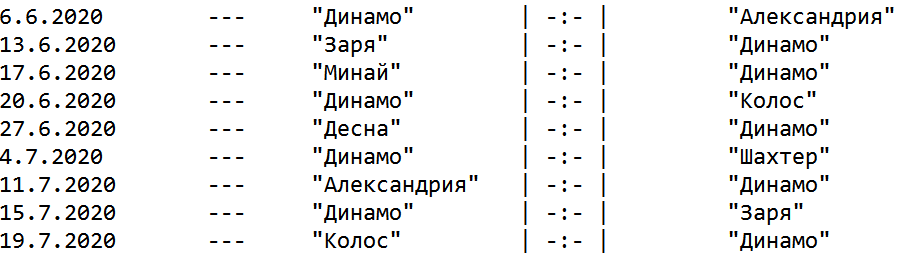
(Додаток 3.3.5)



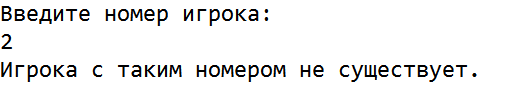
(Додаток 3.3.6)



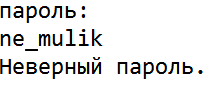
(Додаток 3.3.7)



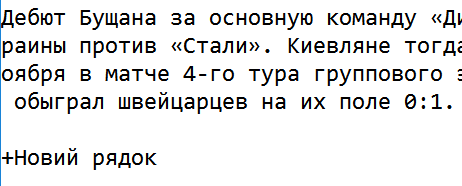
(Додаток 3.3.8)



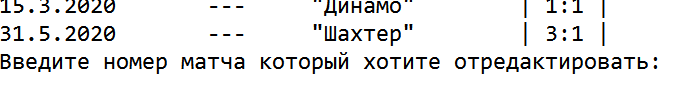
(Додаток 3.3.9)



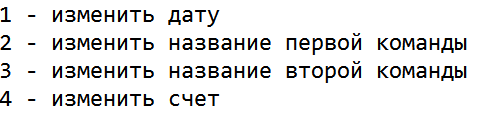
(Додаток 3.3.10)



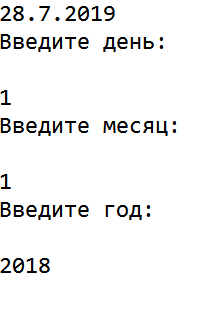
(Додаток 3.3.11, 12)



(Додаток 3.3.13, 14)

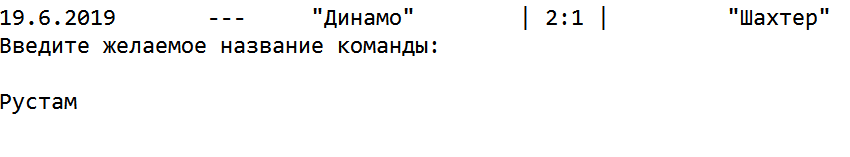


(Додаток 3.3.16)



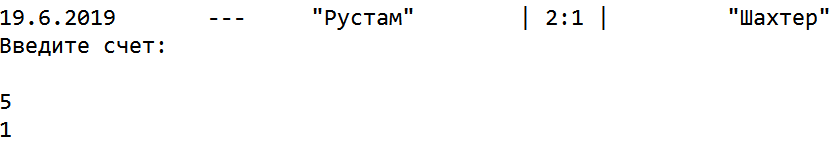


(Додаток 3.3.17)



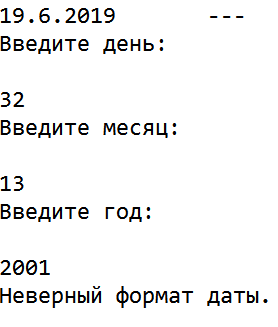


(Додаток 3.3.18, 19)





(Додаток 3.3.20)



(Додаток 3.3.21)

Додаток 2.2.1:

using System;

using System.Collections.Generic;

using System.Text;

namespace football

{

class Player

{

private string name;

private int number;

private string info;

public Player() { }

public Player(string name, int number)

{

this.name = name;

this.number = number;

}

public string Name

{

get { return name; }

}

public int Number

{

get { return number; }

}

public string Info

{

get

{

if (info != null)

return info;

else

return "Информация отсутствует.";

}

set { info = value; }

}

public override string ToString()

{

return "Имя: " + name + "\nНомер: " + number.ToString() + "\n\nИнформация об игроке: " + info.ToString();

}

}

}

Додаток 2.2.2:

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

namespace football

{

class Team

{

private LinkedList<Player> team = new LinkedList<Player>();

private string name;

private string info;

public Team(string name)

{

this.name = name;

}

public Team(string name, LinkedList<Player> players)

{

this.name = name;

team = players;

}

public void InitPlayers(string way)

{

for (int i = 0; i < 100; i++)

{

string path = way + i + ".txt";

if (File.Exists(path))

{

using (StreamReader sr = new StreamReader(path))

{

team.push\_back(new Player(sr.ReadLine().ToString(), i));

}

}

}

}

public void SetInfo(string way)

{

info = new Text(way).ToString();

}

public void SetPlayersInfo(string way)

{

for (int i = 0; i < team.Size; i++)

{

for (int j = 0; j < 100; j++)

{

string path = way + j + ".txt";

if (team[i].Number == j || File.Exists(path))

{

team[i].Info = new Text(path).ToString();

}

}

}

}

public string Name

{

get { return name; }

}

public string Info

{

get

{

if (info != null)

return info;

else

return "Информация отсутствует.";

}

set { info = value; }

}

public string GetPlayers()

{

string s = null;

for (int i = 0; i < team.Size; i++)

{

s += "\n" + team[i].Number + " " + team[i].Name + "\n";

}

return s;

}

public Player this[int index]

{

get

{

int num = 1000;

for (int i = 0; i < team.Size; i++)

{

if(team[i].Number == index)

{

num = i;

}

}

if (num == 1000)

throw new IndexOutOfRangeException();

else

return team[num];

}

}

public override string ToString()

{

string s = "Название команды: " + name + "\n\nИнформация о команде:\n\n" + info.ToString() + "\n\nПолный список игроков:\n";

for(int i = 0; i < team.Size; i++)

{

s += "\n" + team[i].Number + " " + team[i].Name + "\n";

}

return s;

}

}

}

Додаток 2.2.3:

class Match

{

protected Team team1;

protected Team team2;

protected Date date;

protected int goal1;

protected int goal2;

public Match() { }

public void SetDate(int dd, int mm, int yy)

{

date = new Date(dd, mm, yy);

}

public static Was MatchToWas(Match match, int goal1, int goal2)

{

Was was = new Was(match.Date, match.Team1, goal1, goal2, match.Team2);

return was;

}

public Date Date

{

get { return date; }

set { date = value; }

}

public Team Team1

{

get { return team1; }

set { team1 = value; }

}

public Team Team2

{

get { return team2; }

set { team2 = value; }

}

public int Goal1

{

get { return goal1; }

set { goal1 = value; }

}

public int Goal2

{

get { return goal2; }

set { goal2 = value; }

}

}

Додаток 2.2.4:

class Was : Match

{

public Was() { }

public Was(Date dat, Team tem1, int gol1, int gol2, Team tem2)

{

date = dat;

team1 = tem1;

goal1 = gol1;

goal2 = gol2;

team2 = tem2;

}

public override string ToString()

{

return date.ToString() + "\t---\t" + team1.Name + " \t| " + goal1 + ":" + goal2 + " | \t" + team2.Name;

}

}

Додаток 2.2.5:

class Will : Match

{

public Will(Date dat, Team tem1, Team tem2)

{

date = dat;

team1 = tem1;

team2 = tem2;

}

public override string ToString()

{

return date.ToString() + "\t---\t" + team1.Name + " \t| -:- | \t" + team2.Name;

}

}

Додаток 2.2.6:

class Board

{

protected static LinkedList<Match> matches = new LinkedList<Match>();

protected static int size;

public int Size

{

get { return size; }

}

}

Додаток 2.2.7:

class History : Board

{

public static LinkedList<Match> Init(string way)

{

matches = new LinkedList<Match>();

size = 0;

using (StreamReader sr = new StreamReader(way))

{

string line;

while ((line = sr.ReadLine()) != null)

{

string[] splitLine = line.Split('.');

Date date = new Date(Convert.ToInt32(splitLine[0]), Convert.ToInt32(splitLine[1]), Convert.ToInt32(splitLine[2]));

Team team1 = new Team(sr.ReadLine());

string[] count = sr.ReadLine().Split(" : ");

int goal1 = Convert.ToInt32(count[0]);

int goal2 = Convert.ToInt32(count[1]);

Team team2 = new Team(sr.ReadLine());

matches.push\_back(new Was(date, team1, goal1, goal2, team2));

size++;

}

}

return matches;

}

}

Додаток 2.2.8:

class Announcement : Board

{

public static LinkedList<Match> Init(string way)

{

matches = new LinkedList<Match>();

size = 0;

using (StreamReader sr = new StreamReader(way))

{

string line;

while ((line = sr.ReadLine()) != null)

{

string[] splitLine = line.Split('.');

Date date = new Date(Convert.ToInt32(splitLine[0]), Convert.ToInt32(splitLine[1]), Convert.ToInt32(splitLine[2]));

Team team1 = new Team(sr.ReadLine());

sr.ReadLine();

Team team2 = new Team(sr.ReadLine());

matches.push\_back(new Will(date, team1, team2));

size++;

}

}

return matches;

}

}

Додаток 2.2.9:

using System;

using System.Collections.Generic;

using System.Text;

using System.IO;

namespace football

{

class Text

{

private static string way;

private static string way2 = "../../../files/copy/temp.txt";

private string output;

public Text()

{

output = "Информация отсутствует.";

}

public Text(string wayy)

{

way = wayy;

}

public static void AddText(string input)

{

using (StreamWriter sw = new StreamWriter(way, true))

{

sw.WriteLine(input);

}

}

public static void DeleteStr(int line\_to\_delete)

{

string line = null;

int line\_number = 0;

using (StreamReader reader = new StreamReader(way))

{

using (StreamWriter writer = new StreamWriter(way2))

{

while ((line = reader.ReadLine()) != null)

{

line\_number++;

if (line\_number == line\_to\_delete)

continue;

writer.WriteLine(line);

}

}

}

using (StreamReader reader = new StreamReader(way2))

{

using (StreamWriter writer = new StreamWriter(way))

{

while ((line = reader.ReadLine()) != null)

{

writer.WriteLine(line);

}

}

}

File.Delete(way2);

}

public override string ToString()

{

output = null;

using (StreamReader sr = new StreamReader(way))

{

string line;

while ((line = sr.ReadLine()) != null)

{

output += line + "\n";

}

}

return output;

}

}

}

Додаток 2.2.10:

using System;

using System.Collections.Generic;

using System.Text;

namespace football

{

class Date

{

private int day;

private int month;

private int year;

public Date(int dd, int mm, int yy)

{

if (dd < 32 && mm < 13)

{

day = dd;

month = mm;

year = yy;

}

else

throw new Exception("Неверный формат даты.");

}

public override string ToString()

{

return day.ToString() + "." + month.ToString() + "." + year.ToString();

}

}

}

Додаток 2.2.11:

using System;

using System.Collections.Generic;

using System.Text;

namespace football

{

class LinkedList<T>

{

private LinkedListNode<T> head;

private int size;

public LinkedList()

{

size = 0;

head = null;

}

public int Size

{

get { return size; }

}

public void push\_back(T data)

{

if (head == null)

{

head = new LinkedListNode<T>(data);

}

else

{

LinkedListNode<T> current = this.head;

while (current.pNext != null)

{

current = current.pNext;

}

current.pNext = new LinkedListNode<T>(data);

}

size++;

}

public T this[int index]

{

get

{

int counter = 0;

T result = default(T);

LinkedListNode<T> current = this.head;

while (current != null)

{

if (counter == index)

{

result = current.data;

}

current = current.pNext;

counter++;

}

return result;

}

}

}

}

Додаток 2.2.12:

using System;

using System.Collections.Generic;

using System.Text;

namespace football

{

class LinkedListNode<T>

{

public T data;

public LinkedListNode<T> pNext;

public LinkedListNode(T data, LinkedListNode<T> pNext = null)

{

this.data = data;

this.pNext = pNext;

}

}

}

Додаток 2.2.13:

using System;

using System.Dynamic;

using System.Threading;

namespace football

{

class Program

{

static void Main(string[] args)

{

try

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

bool x = true;

do

{

Console.Clear();

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Клуб футбольных фанатов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");

Console.WriteLine(" 1 – войти как гость\n\n 2 – войти как администратор\n\n 3 - инструкция");

char t1 = Console.ReadKey().KeyChar;

if (t1 == '1')

{

do

{

Console.Clear();

Console.WriteLine("Выберите команду:\n\n\n1 – Динамо\n\n2 – Шахтер");

char t2 = Console.ReadKey().KeyChar;

if (t2 == '1')

{

do

{

Console.Clear();

Team dynamo = new Team("Динамо");

dynamo.InitPlayers("../../../files/dyn/");

dynamo.SetInfo("../../../files/dynamo.txt");

dynamo.SetPlayersInfo("../../../files/dyn/");

Console.WriteLine(" 1 – показать список игроков\n 2 – показать информацию об игроке по его номеру\n 3 – показать информацию о команде\n 4 – узнать историю матчей команды за 2019 – 2020 года\n 5 – новости об анонсах матчей команды");

char t3 = Console.ReadKey().KeyChar;

if (t3 == '1')

{

Console.Clear();

Console.WriteLine(dynamo.GetPlayers());

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '2')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

Console.WriteLine(dynamo[Convert.ToInt32(Console.ReadLine())].Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '3')

{

Console.Clear();

Console.WriteLine(dynamo.Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '4')

{

Console.Clear();

LinkedList<Match> matches = History.Init("../../../files/match\_dynamo\_was.txt");

for (int i = 0; i < matches.Size; i++)

{

Console.WriteLine(matches[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '5')

{

Console.Clear();

LinkedList<Match> matches = Announcement.Init("../../../files/match\_dynamo\_will.txt");

for (int i = 0; i < matches.Size; i++)

{

Console.WriteLine(matches[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '/')

break;

} while (x);

}

else if (t2 == '2')

{

do

{

Console.Clear();

Team shahter = new Team("Шахтер");

shahter.InitPlayers("../../../files/shah/");

shahter.SetInfo("../../../files/shahter.txt");

shahter.SetPlayersInfo("../../../files/shah/");

Console.WriteLine(" 1 – показать список игроков\n 2 – показать информацию об игроке по его номеру\n 3 – показать информацию о команде\n 4 – узнать историю матчей команды за 2019 – 2020 года\n 5 – новости об анонсах матчей команды");

char t3 = Console.ReadKey().KeyChar;

if (t3 == '1')

{

Console.Clear();

Console.WriteLine(shahter.GetPlayers());

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '2')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

Console.WriteLine(shahter[Convert.ToInt32(Console.ReadLine())].Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '3')

{

Console.Clear();

Console.WriteLine(shahter.Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '4')

{

Console.Clear();

LinkedList<Match> matches = History.Init("../../../files/match\_shahter\_was.txt");

for (int i = 0; i < matches.Size; i++)

{

Console.WriteLine(matches[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '5')

{

Console.Clear();

LinkedList<Match> matches = Announcement.Init("../../../files/match\_shahter\_will.txt");

for (int i = 0; i < matches.Size; i++)

{

Console.WriteLine(matches[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '/')

break;

} while (x);

}

else if (t2 == '/')

break;

} while (x);

}

else if (t1 == '2')

{

do

{

Console.Clear();

Console.WriteLine("пароль: ");

string password = Console.ReadLine();

if (password == "mulik")

{

do

{

Console.Clear();

Console.WriteLine("Выберите команду:\n\n\n1 – Динамо\n\n2 – Шахтер");

char t2 = Console.ReadKey().KeyChar;

if (t2 == '1')

{

do

{

Console.Clear();

Team dynamo = new Team("Динамо");

string dyn = "../../../files/dyn/"; //!!!!!!!!!!!!!!!!

string info = "../../../files/dynamo.txt";

dynamo.InitPlayers(dyn);

dynamo.SetInfo(info);

dynamo.SetPlayersInfo(dyn);

LinkedList<Match> matches\_was = History.Init("../../../files/match\_dynamo\_was.txt");

LinkedList<Match> matches\_will = Announcement.Init("../../../files/match\_dynamo\_will.txt");

Console.WriteLine(" 1 – показать список игроков\n 2 – показать информацию об игроке по его номеру\n 3 - редактировать информацию об игроке по его номеру\n 4 – показать информацию о команде\n 5 – редактировать информацию о команде\n 6 – узнать историю матчей команды за 2019 – 2020 года\n 7 – новости об анонсах матчей команды\n 8 – редактировать историю матчей команды\n 9 – редактировать анонсы матчей команды");

char t3 = Console.ReadKey().KeyChar;

if (t3 == '1')

{

Console.Clear();

Console.WriteLine(dynamo.GetPlayers());

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '2')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

Console.WriteLine(dynamo[Convert.ToInt32(Console.ReadLine())].Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '3')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

string num = Console.ReadLine();

do

{

Console.Clear();

Console.WriteLine(new Text(dyn + num + ".txt"));

char t4 = Console.ReadKey().KeyChar;

if (t4 == '-')

{

Console.WriteLine("Введите номер строки которую хотите удалить: ");

Text.DeleteStr(Convert.ToInt32(Console.ReadLine()));

continue;

}

else if (t4 == '+')

{

do

{

Text.AddText(Console.ReadLine());

}

while (Console.ReadKey().KeyChar != '/');

Console.WriteLine(" ");

continue;

}

else if (t4 == '/')

break;

} while (x);

}

else if (t3 == '4')

{

Console.Clear();

Console.WriteLine(dynamo.Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '5')

{

do

{

Console.Clear();

Console.WriteLine(new Text(info));

char t4 = Console.ReadKey().KeyChar;

if (t4 == '-')

{

Console.WriteLine("Введите номер строки которую хотите удалить: ");

Text.DeleteStr(Convert.ToInt32(Console.ReadLine()));

continue;

}

else if (t4 == '+')

{

do

{

Text.AddText(Console.ReadLine());

}

while (Console.ReadKey().KeyChar != '/');

Console.WriteLine(" ");

continue;

}

else if (t4 == '/')

break;

} while (x);

}

else if (t3 == '6')

{

Console.Clear();

for (int i = 0; i < matches\_was.Size; i++)

{

Console.WriteLine(matches\_was[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '7')

{

Console.Clear();

for (int i = 0; i < matches\_will.Size; i++)

{

Console.WriteLine(matches\_will[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '8')

{

Console.Clear();

for (int i = 0; i < matches\_was.Size; i++)

{

Console.WriteLine(matches\_was[i]);

}

Console.WriteLine("Введите номер матча который хотите отредактировать: ");

int num = Convert.ToInt32(Console.ReadLine()) - 1;

do

{

Console.Clear();

Console.WriteLine("1 - изменить дату \n2 - изменить название первой команды \n3 - изменить название второй команды \n4 - изменить счет\n");

char t4 = Console.ReadKey().KeyChar;

if (t4 == '1')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите день: \n");

int dd = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите месяц: \n");

int mm = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите год: \n");

int yy = Convert.ToInt32(Console.ReadLine());

matches\_was[num].SetDate(dd, mm, yy);

Console.Clear();

Console.WriteLine(matches\_was[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '2')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_was[num].Team1 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_was[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '3')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team2 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '4')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите счет: \n");

matches\_was[num].Goal1 = Convert.ToInt32(Console.ReadLine());

matches\_was[num].Goal2 = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '/')

break;

} while (x);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '9')

{

Console.Clear();

for (int i = 0; i < matches\_will.Size; i++)

{

Console.WriteLine(matches\_will[i]);

}

Console.WriteLine("Введите номер матча который хотите отредактировать: ");

int num = Convert.ToInt32(Console.ReadLine()) - 1;

do

{

Console.Clear();

Console.WriteLine("1 - изменить дату \n2 - изменить название первой команды \n3 - изменить название второй команды \n4 - записать счет\n");

char t4 = Console.ReadKey().KeyChar;

if (t4 == '1')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите день: \n");

int dd = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите месяц: \n");

int mm = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите год: \n");

int yy = Convert.ToInt32(Console.ReadLine());

matches\_will[num].SetDate(dd, mm, yy);

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '2')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team1 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '3')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team2 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '4')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите счет: \n");

Was was = Match.MatchToWas(matches\_will[num], Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine()));

Console.Clear();

Console.WriteLine(was);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '/')

break;

} while (x);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '/')

break;

} while (x);

}

else if (t2 == '2')

{

do

{

Console.Clear();

Team shahter = new Team("Шахтер");

string shah = "../../../files/shah/"; //!!!!!!!!!!!!!!!!

string info = "../../../files/shahter.txt";

shahter.InitPlayers(shah);

shahter.SetInfo(info);

shahter.SetPlayersInfo(shah);

LinkedList<Match> matches\_was = History.Init("../../../files/match\_shahter\_was.txt");

LinkedList<Match> matches\_will = Announcement.Init("../../../files/match\_shahter\_will.txt");

Console.WriteLine(" 1 – показать список игроков\n 2 – показать информацию об игроке по его номеру\n 3 - редактировать информацию об игроке по его номеру\n 4 – показать информацию о команде\n 5 – редактировать информацию о команде\n 6 – узнать историю матчей команды за 2019 – 2020 года\n 7 – новости об анонсах матчей команды\n 8 – редактировать историю матчей команды\n 9 – редактировать анонсы матчей команды");

char t3 = Console.ReadKey().KeyChar;

if (t3 == '1')

{

Console.Clear();

Console.WriteLine(shahter.GetPlayers());

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '2')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

Console.WriteLine(shahter[Convert.ToInt32(Console.ReadLine())].Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '3')

{

Console.Clear();

Console.WriteLine("Введите номер игрока: ");

string num = Console.ReadLine();

do

{

Console.Clear();

Console.WriteLine(new Text(shah + num + ".txt"));

char t4 = Console.ReadKey().KeyChar;

if (t4 == '-')

{

Console.WriteLine("Введите номер строки которую хотите удалить: ");

Text.DeleteStr(Convert.ToInt32(Console.ReadLine()));

continue;

}

else if (t4 == '+')

{

do

{

Text.AddText(Console.ReadLine());

}

while (Console.ReadKey().KeyChar != '/');

Console.WriteLine(" ");

continue;

}

else if (t4 == '/')

break;

} while (x);

}

else if (t3 == '4')

{

Console.Clear();

Console.WriteLine(shahter.Info);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '5')

{

do

{

Console.Clear();

Console.WriteLine(new Text(info));

char t4 = Console.ReadKey().KeyChar;

if (t4 == '-')

{

Console.WriteLine("Введите номер строки которую хотите удалить: ");

Text.DeleteStr(Convert.ToInt32(Console.ReadLine()));

continue;

}

else if (t4 == '+')

{

do

{

Text.AddText(Console.ReadLine());

}

while (Console.ReadKey().KeyChar != '/');

Console.WriteLine(" ");

continue;

}

else if (t4 == '/')

break;

} while (x);

}

else if (t3 == '6')

{

Console.Clear();

for (int i = 0; i < matches\_was.Size; i++)

{

Console.WriteLine(matches\_was[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '7')

{

Console.Clear();

for (int i = 0; i < matches\_will.Size; i++)

{

Console.WriteLine(matches\_will[i]);

}

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '8')

{

Console.Clear();

for (int i = 0; i < matches\_was.Size; i++)

{

Console.WriteLine(matches\_was[i]);

}

Console.WriteLine("Введите номер матча который хотите отредактировать: ");

int num = Convert.ToInt32(Console.ReadLine()) - 1;

do

{

Console.Clear();

Console.WriteLine("1 - изменить дату \n2 - изменить название первой команды \n3 - изменить название второй команды \n4 - изменить счет\n");

char t4 = Console.ReadKey().KeyChar;

if (t4 == '1')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите день: \n");

int dd = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите месяц: \n");

int mm = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите год: \n");

int yy = Convert.ToInt32(Console.ReadLine());

matches\_was[num].SetDate(dd, mm, yy);

Console.Clear();

Console.WriteLine(matches\_was[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '2')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_was[num].Team1 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_was[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '3')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team2 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '4')

{

Console.Clear();

Console.WriteLine(matches\_was[num]);

Console.WriteLine("Введите счет: \n");

matches\_was[num].Goal1 = Convert.ToInt32(Console.ReadLine());

matches\_was[num].Goal2 = Convert.ToInt32(Console.ReadLine());

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '/')

break;

} while (x);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '9')

{

Console.Clear();

for (int i = 0; i < matches\_will.Size; i++)

{

Console.WriteLine(matches\_will[i]);

}

Console.WriteLine("Введите номер матча который хотите отредактировать: ");

int num = Convert.ToInt32(Console.ReadLine()) - 1;

do

{

Console.Clear();

Console.WriteLine("1 - изменить дату \n2 - изменить название первой команды \n3 - изменить название второй команды \n4 - записать счет\n");

char t4 = Console.ReadKey().KeyChar;

if (t4 == '1')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите день: \n");

int dd = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите месяц: \n");

int mm = Convert.ToInt32(Console.ReadLine());

Console.WriteLine("Введите год: \n");

int yy = Convert.ToInt32(Console.ReadLine());

matches\_will[num].SetDate(dd, mm, yy);

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '2')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team1 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '3')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите желаемое название команды: \n");

matches\_will[num].Team2 = new Team("\"" + Console.ReadLine() + "\"");

Console.Clear();

Console.WriteLine(matches\_will[num]);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '4')

{

Console.Clear();

Console.WriteLine(matches\_will[num]);

Console.WriteLine("Введите счет: \n");

Was was = Match.MatchToWas(matches\_will[num], Convert.ToInt32(Console.ReadLine()), Convert.ToInt32(Console.ReadLine()));

Console.Clear();

Console.WriteLine(was);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t4 == '/')

break;

} while (x);

if (Console.ReadKey().KeyChar == '/')

continue;

}

else if (t3 == '/')

break;

} while (x);

}

else if (t2 == '/')

break;

} while (x);

}

else if (password == "/")

break;

else

{

Console.WriteLine("Неверный пароль.");

Thread.Sleep(1000);

}

} while (x);

}

else if (t1 == '3')

{

do

{

Console.Clear();

Console.WriteLine("Нажимайте цифры в соответствии с указаниями. \nЧтобы вернуться назад нажмите \"/\" \nВ режиме редактирования нажмите \"+\" чтобы ввести новую строку \nили \"-\" чтобы удалить строку по номеру");

if (Console.ReadKey().KeyChar == '/')

break;

} while (x);

}

else if (t1 == '/')

break;

} while (x);

}

catch (IndexOutOfRangeException)

{

Console.WriteLine("Игрока с таким номером не существует.");

}

catch (Exception e)

{

Console.WriteLine(e.Message);

}

}

}

}