# Міністерство освіти і науки України

# Національний університет «Львівська політехніка»

**Кафедра ЕОМ**



**Курсова робота**

з предмету: «Програмування, частина 2 (Об’єктно-орієнтованепрограмування)»

на тему:

# «Базові принципи об’єктно-орієнтованого програмування»

Індивідуальне завдання: «Футбольний клуб »

**Виконав:**ст. гр. КІ-15

Гвоздь Ю. А.  
**Прийняв:**викладач

Козак Н. Б.

Львів 2020

## Анотація

Курсова робота складається з опису виконання поставленої задачі та графічної частини. В графічній частині наведено алгоритм виконання завдання, оцінка його складності.

Опис виконання завдання включає розділи: огляд та обґрунтування вибору технологій об’єктно-орієнтованого програмування, аналіз та розробка алгоритму, програмна реалізація алгоритму, відлагодження та тестування розробленої програми, аналіз продуктивності створеної програми, висновок, лістинг програми.

В даній курсовій роботі була виконана розробка програми на мові програмування С++ з використанням об’єктно-орієнтованого програмування, яка включала в себе роботу з файлами, класами, обробку числової інформації і роботу з векторами та об’єктами.

# Summary

Coursework consists of a description of the implementation of the task and the graphic part. In the graphic part the algorithm of the task execution, estimation of its complexity is given.

Description of the task includes sections: review and justification of the choice of technology object-oriented programming, analysis and development of algorithm, software implementation of the algorithm, debugging and testing of the developed program, analysis of the productivity of the program, conclusion, listing program.

In this course work, the development of the program in the programming language C ++ with the use of object-oriented programming, which included work with files, classes, processing of numeric information and work with vectors and objects.

Зміст

[Завдання на курсову роботу 6](#_Toc515462736)

[Вступ 7](#_Toc515462737)

[1. Визначення поняття Об’єктно орієнтованого програмування(ООП), його основні принципи та обгрунтування його технологій. 9](#_Toc515462738)

[1.1 ООП як основа сучасного програмування 9](#_Toc515462739)

[1.2 Визначення ООП 10](#_Toc515462740)

[1.3 Фундаментальні поняття ООП 10](#_Toc515462741)

[1.4 Переваги вибраного методу 11](#_Toc515462742)

[2. Аналіз та розробка алгоритму згідно індивідуального завдання 13](#_Toc515462743)

[2.1 Загальний опис поставленої задачі 13](#_Toc515462744)

[2.2 Польовий гравець 13](#_Toc515462745)

[2.3. Голкіпер 13](#_Toc515462746)

[2.4 Команда 14](#_Toc515462747)

[2.5 Матч 14](#_Toc515462748)

[2.6 Чемпіонат 14](#_Toc515462749)

[2.7 Визначення основних задач покладених в основу програми 15](#_Toc515462750)

[2.8 Алгоритм виконання роботи програми 16](#_Toc515462751)

[3. Реалізація алгоритму у програмному коді 17](#_Toc515462752)

[3.1 Наслідування класів 17](#_Toc515462753)

[4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання 19](#_Toc515462754)

[5. Аналіз структурної складності виконання програми 26](#_Toc515462755)

[Висновок 27](#_Toc515462756)

[Література 28](#_Toc515462757)

[Додаток А. Код програми 29](#_Toc515462758)

[Додаток Б.Блок схема алгоритму 39](#_Toc515462759)

[Додаток В.Діаграмма класів 40](#_Toc515462760)

# Завдання на курсову роботу

У курсовій роботі повинно бути реалізовано на мові С++ таку предметну галузь як “Футбольний клуб”. У ній повинні бути використані основні принципи об’єктно-орієнтованого програмування. Під футбольним клубом мається на увазі реалізація зберігання даних про футболістів.

# Вступ

**Об'є́ктно-орієнто́ване програмува́ння** (ООП) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: інкапсуляція, успадкування та поліморфізм. Одною з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в 1960-тих роках, вона не мала широкого застосування до 1990-тих, коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Однією з таких мов програмування стала мова С++. У 1990-х р. дана мова стала однією з найуживаніших мов програмування загального призначення. ЇЇ розробником є професор Техаського університету А&М Б'ярн Страуструп. На початку свого існування ця мова дістала назву «Сі з класами». Згодом Страуструп перейменував мову на C++ у 1983 р. У 1998 році ратифіковано міжнародний стандарт мови С++: ISO/IEC 14882 «Standard for the C++ Programming Language». Поточна версія цього стандарту — ISO/IEC 14882:2011. На сьогоднішній день цю мову використовують для системного програмування, розробки програмного забезпечення, написання драйверів, потужних серверних та клієнтських програм, а також для розробки розважальних програм таких як відеоігри. Також мова С++ суттєво вплинула на інші, популярні сьогодні, мови програмування такі як С# та Java [1].

[ООП](https://uk.wikipedia.org/wiki/%D0%9E%D0%9E%D0%9F) виникло в результаті розвитку ідеології [процедурного програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), де дані і підпрограми (процедури, функції) їх обробки формально не пов'язані. Для подальшого розвитку об'єктно-орієнтованого програмування велике значення мають поняття події(так зване [подієво-орієнтоване програмування](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D1%96%D1%94%D0%B2%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) і компоненти ([компонентне програмування](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), КОП).

Формування КОП від ООП відбулося, так само як формування [модульного](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) від процедурного програмування: процедури сформувалися в модулі - незалежні частини коду до рівня [збірки програми](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D1%83%D0%B2%D0%B0%D0%BB%D1%8C%D0%BD%D0%B8%D0%BA), так об'єкти сформувалися в компоненти - незалежні частини коду до рівня [виконання програми](https://uk.wikipedia.org/w/index.php?title=%D0%A0%D0%B0%D0%BD%D1%82%D0%B0%D0%B9%D0%BC&action=edit&redlink=1). Взаємодія об'єктів відбувається за допомогою [повідомлень](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F#%D0%9F%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F). Результатом подальшого розвитку ООП, мабуть, буде [агентно-орієнтоване програмування](https://uk.wikipedia.org/wiki/%D0%90%D0%B3%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9_%D0%BF%D1%96%D0%B4%D1%85%D1%96%D0%B4), де *агенти* - незалежні частини коду на рівні виконання. Взаємодія агентів відбувається за допомогою зміни *середовища*, в якій вони знаходяться.

На відміну від традиційних поглядів, коли програму розглядали як набір [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0), або як перелік [інструкцій](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BA%D1%86%D1%96%D1%8F) комп'ютеру, ООП програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожний об'єкт здатний отримувати [повідомлення](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%D0%BC%D1%96%D0%BD_%D0%BF%D0%BE%D0%B2%D1%96%D0%B4%D0%BE%D0%BC%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F%D0%BC%D0%B8), обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт — своєрідний незалежний автомат з окремим призначенням та відповідальністю.

**Метою виконання** цієї курсової роботи є засвоєння основних методів та алгоритмів об’єктно-орієнтованого програмування мови С++, та практичне застосування їх у цій сфері.

# 

# 1. Визначення поняття Об’єктно орієнтованого програмування(ООП), його основні принципи та обгрунтування його технологій.

# 1.1 ООП як основа сучасного програмування

Із розвитком комп’ютерних технологій, коли програми поступово почали ускладнюватися, виникла потреба для нових засобів програмування, які б могли впорядкувати та спростити сприйняття та написання програм, які могли б містити тисячі стрічок коду. Для подолання цієї проблеми було розроблено структурне програмування. Структурне програмування, на відміну від його попередника — стихійного програмування, має певні правила та засоби, які суттєво вдосконалили процес написання програм. Але за допомогою структурного програмування дуже важко писати програми ще більшої складності, які вимагає сучасність. Тому була створена парадигма Об’єктноорієнтованого програмування. Першою мовою програмування, в якій були запропоновані основні поняття, які згодом склалися в парадигму, була Симула, але термін «об'єктна орієнтованість" не використовувався в контексті використання цієї мови. У момент його появи в 1967 році в ньому були запропоновані революційні ідеї: об'єкти, класи, віртуальні методи і ін., проте це все не було сприйнято сучасниками як щось грандіозне. Фактично, Симула була «Алголом з класами», що спрощує вираз в процедурному програмуванні багатьох складних концепцій. Поняття класу в Симулі може бути повністю визначено через композицію конструкцій Алгола (тобто клас в Симулі - це щось складне, що описується за допомогою примітивів).

Погляд на програмування «під новим кутом» (відмінним від процедурного) запропонували Алан Кей і Ден Інгаллс в мові Smalltalk. Тут поняття класу стало основотвірною ідеєю для всіх інших конструкцій мови (тобто клас в Смолток є примітивом, за допомогою якого описані більш складні конструкції). Саме ця мова стала першою широко поширеною об'єктно-орієнтованою мовою програмування.

В даний час кількість прикладних мов програмування (список мов), що реалізують об'єктно-орієнтовану парадигму, є найбільшою по відношенню до інших парадигм. Найбільш поширені в промисловості такі мови як С ++, Delphi, C #, Java які втілюють об'єктну модель Симула. Прикладами мов, що спираються на модель Смолтока, є Objective-C, Python, Ruby.

# 1.2 Визначення ООП

Об'єктно-орієнтоване програмування (ООП) - методологія програмування, заснована на представленні програми у вигляді сукупності об'єктів, кожен з яких є екземпляром певного класу, а класи утворюють ієрархію спадкування. Лука Карделла і Мартін Абаді побудували теоретичне обгрунтування ООП і класифікацію на основі цього обгрунтування. Вони відзначають, що виділені ними поняття і категорії разом зустрічаються далеко не у всіх ОО-мовах, більшість мов підтримують лише підмножини теорії, а часом і своєрідні відхилення від неї. [2]

Ключові поняття:

* квантифікація змінних типу (універсальна, екзистенційна, обмежена);
* підтипізація (англ. subtyping - відносини «супертип-підтип»);
* включення (англ. subsumption) - окремий випадок підтипізації;
* об'єкт;
* об'єктний тип (фактично, спеціальна форма «запису», в число полів якої входять функції першого класу);
* клас.

[5] Клас описує абстрактну поведінку. Об'єктні типи будуються на основі класу за допомогою додавання різних приватних полів і методів. Об'єкт (тобто значення об'єктного типу, традиційно називається «екземпляром класу») породжується конструктором на основі початкових параметрів.

# 1.3 Фундаментальні поняття ООП

До фундаментальних понять (принципів), використаних у переважній більшості визначень об'єктно-орієнтованого програмування належать:

* *Абстракція даних* – виділення значущої інформації і виключення її з розгляду незначною. В ООП розглядають лише абстракцію даних (нерідко називаючи її просто «абстракцією»), маючи на увазі набір значущих характеристик об'єкта, доступних решті частини програми.
* *Інкапсуляція –* властивість системи, що дозволяє об'єднати дані і методи, що працюють з ними, в класі. Одні мови (наприклад, С ++, Java або Ruby) ототожнюють інкапсуляцію з приховуванням, але інші (Smalltalk, Eiffel, OCaml) розрізняють ці поняття.
* *Спадкування –* властивість системи, що дозволяє описати новий клас на основі вже існуючого з частково або повністю запозичиною функціональністю. Клас, від якого відбувається спадкування, називається базовим, батьківським або суперкласом. Новий клас – нащадком, спадкоємцем, дочірнім або похідним класом.
* *Поліморфізм підтипів –* властивість системи, що дозволяє використовувати об'єкти з однаковим інтерфейсом без інформації про тип і внутрішню структуру об'єкта. Інший вид поліморфізму – параметричний - в ООП називають узагальненим програмуванням.
* *Клас –* абстрактні характеристики деякої сутності, включаючи

характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінки, методи або можливості). Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Як правило, клас має бути зрозумілим для не-програмістів, що знаються на предметній області, що, у свою чергу, значить, що клас повинен мати значення в контексті. Також, код реалізації класу має бути досить самодостатнім. Властивості та методи класу, разом називаються його членами.

* *Об’єкт –* сутність в адресному просторі обчислювальної системи, що з'являється при створенні екземпляра класу (наприклад, після запуску результатів компіляції і зв'язування вихідного коду на виконання).

# 1.4 Переваги вибраного методу

Від будь-якого методу програмування ми чекаємо, що він допоможе нам у вирішенні наших проблем. Але однією з найбільш значних проблем у програмуванні є складність. Чим більша і складніша програма, тим важливіше стає розбити її на невеликі, чітко окреслені частини. Щоб побороти складність, ми повинні абстрагуватися від дрібних деталей. У цьому сенсі класи являють собою дуже зручний інструмент. Класи дозволяють проводити конструювання з корисних компонент, що володіють простими інструментами, що дає можливість абстрагуватися від деталей реалізації. Також ООП дає можливість створювати розширювані системи. Це одна з найбільш значних переваг ООП і саме вона відрізняє даний підхід від традиційних методів програмування. Розширюваність означає, що існуючу систему можна змусити працювати з новими компонентами, причому без внесення до неї будь-яких змін. Компоненти можуть бути додані на етапі виконання. Багаторазового використання програмного забезпечення на практиці досягти не вдається з-за того, що існуючі компоненти вже не відповідають новим вимогам. ООП допомагає досягти цього без порушення роботи вже наявних клієнтів, що дозволяє нам витягти максимум з багаторазового використання компонент. Завдяки чому ми скорочуємо час на розробку, яке з вигодою може бути віддано іншим проектам. Також компоненти багаторазового використання звичайно містять набагато менше помилок, ніж знову розроблені, адже вони вже не раз піддавалися перевірці.

# 2. Аналіз та розробка алгоритму згідно індивідуального завдання

# 2.1 Загальний опис поставленої задачі

Задачею моєї курсової роботи є футбольний клуб. Перед усім його слід розглядати як взаємодію кількох важливих його складових, зокрема польових гравців та голкіперів.

# 2.2 Польовий гравець

Польовий гравець є частиною команди. Польові гравці беруть участь безпосередньо у грі рухаючись по всьому футбольному полі, намагаючись забити м’яч у ворота суперника. У моїй програмі польові гравці мають такий набір властивостей:

* Ім'я
* Прізвище
* Вік
* Номер
* Сила
* Забиті голи
* Ціна

Польовий гравець має на багато більше властивостей, але я виділив основні з них.

# 2.3. Голкіпер

Голкіпер є також частиною команди, але на відміну від польового гравця голкіпер не рухається по всьому полю. Він захищає ворота від атак суперника. Для голкіпера я виділив також декілька властивостей:

* Ім'я
* Прізвище
* Вік
* Номер
* Сила
* Забиті голи
* Ціна

# 2.4 Команда

#### Команда – набір гравців. До команди повинно входити мінімум 11 польових гравців та мінімум 1 голкіпер. В моїй роботі команда реалізована з такими властивостями:

* Ім'я
* Сила
* Бюджет
* Набір польових гравців
* Набір голкіперів

# 2.5 Матч

У моїй роботі матч є додатковим об’єктом . Матч – це одна гра двох команд.

Результати матчу є двоякі. Може перемогти одна з команд відповідно інша програє.

У програмі матч реалізований з такими властивостями:

* Ім'я команд
* Голи команд
* Імя стадіону

# 2.6 Чемпіонат

Чемпіонат також є додатковим об’єктом, на відміну від матчу в чемпіонаті беруть участь декілька команд . Під час чемпіонату команди заробляють очки .

У своїй програмі я виділив декілька в властивостей чемпіонату а саме:

* Назва чемпіонату
* Кількість команд що беруть участь
* Набір команд
* Набір матчів
* Набір очків команд

# 2.7 Визначення основних задач покладених в основу програми

Дана програма написана для вирішення певного роду задач які повністю пов’язані з такими об’єктами як гравці.

Основною задачею, яку повинна виконувати ця програма є операції над даними про гравців, забезпечення користувачу швидкого та зручного доступу до інформації, пошук відповідної назви футболіста у базі даних та швидке додавання нового в базу. В результаті я отримав набір функцій, які мені потрібно реалізувати для виконання поставленої перед мною задачі:

* Запис і зберігання даних у файлах;
* Зчитування даних з файлу;
* Додавання або видалення даних, додатковий параметр очищення;
* Сортування даних для виведення їх екран, для зручності автоматичне сортування;
* Автоматичний пошук потрібних даних;
* Порівняння даних;
* Виконання трансферу;
* Проведення матчу;
* Проведення чемпіонату;

Проаналізувавши мою область програмування, можна вивести основні об’єкти, відносно яких можна сформувати властивості

* Польовий гравець
* Голкіпер
* Команда

Як додаткові об’єкти я розглядаю змагання, матч.

# 2.8 Алгоритм виконання роботи програми

Тип користувача – Менеджер – власне, це той тип користувача для якого і розроблялася ця програма. Він матиме наступний список можливостей:

* Додавання нових даних;
* Перегляд;

1. Автоматичний пошук тхт. файлів з даними та можливість їх сортування
2. Вибір файлу для перегляду

* Видалення даних та їх очищення;

1. Автоматичний пошук тхт. файлів з даними та можливість їх сортування
2. Вибір файлу для видалення чи очищення

* Порівнняня двох даних;

1. Автоматичний пошук тхт. файлів з даними та можливість їх сортування
2. Вибір першого та другого файлу

* Список гравців

1. Створення
2. Показ
3. Очищення
4. Повернення назад

* Вихід;

Cхема роботи усіх вище вказаних процесів зображена у додатку Б.

# 3. Реалізація алгоритму у програмному коді

Після складання алгоритму роботи моєї програми, я приступив до її реалізації програмним кодом. У своїй програмі я використовую 6 класів, а саме: Player (батьківський клас), Field\_Players (клас що описує польових гравців), Goalkeepers (клас де описано голкіперів), Team (клас що описує команду), Match (клас що описує матч), Tournament (клас що описує змагання)

Вся програма побудована на роботі функцій. Нижче вони є наведені основні з них:

void create\_goalkeepers()-створення об'єкта типу голкіпер.

void create\_field\_players()-створення об'єкта типу польовий гравець.

void enter\_team\_in\_file()- запис даних про команду у файл.

void create\_team()- створити команду.

void show\_team\_from\_file(string fTeam\_name)-вивести дані про команду.

void redag\_player()-змінення окремих даних гравців.

void delete\_players()-видалення гравців.

void sort\_team()-сортувати дані.

void transfer()-зробити трансфер.

void add\_players()-додати гравця.

void Hold\_tournament()-почати чемпіонат.

Решта коду знаходиться у додатку А.

# 3.1 Наслідування класів

Також у своїй програмі я використав таку особливість обєктно-орієнтовного програмування як наслідування класів . Це дуже спрощує роботу програми нище наведена гілка наслідування (діаграмма класів).

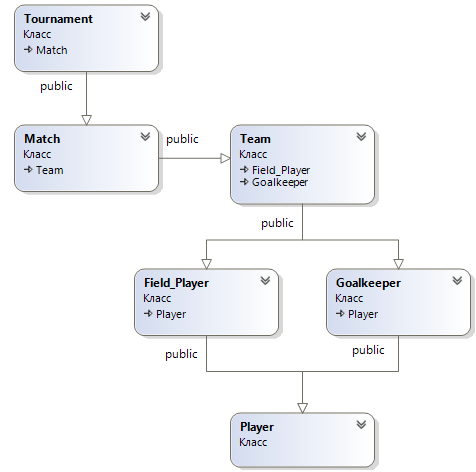


Рис 1. Діаграмма класів

В додатку В наведено повну діаграмму класів.

# 4. Відлагодження та тестування реалізованої програми згідно індивідуального завдання

Під час тестування окремих частин програм з’являлись певні недоліки та помилки. Щоб все правильно працювало потрібно було не дати можливості користувачу змушувати програму некоректно працювати. Тож всі можливі неполадки були ізольовані і опрацьовані так, щоб з’являлось певне повідомлення з певною інформацією, але лише в тих місцях де це дійсно необхідно.

Подібні кроки були виконані і з перевіркою всієї програми. Нижче наведені скріншоти, які запевнюють в коректній роботі програми:

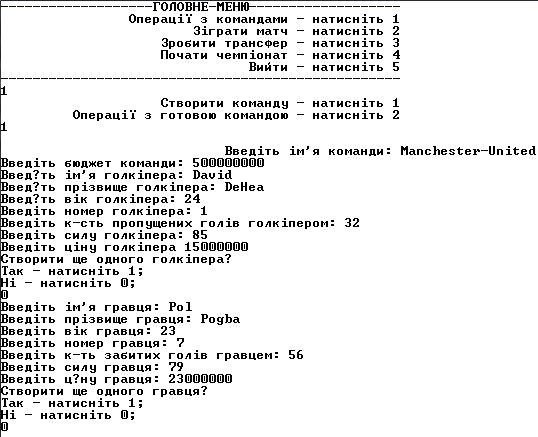


Рис.2 Запис даних про гравців в новий файл

Використовуючи функції void create\_goalkeepers(),void create\_field\_players (), void enter\_team\_in\_file() ми створюємо txt. файл куди записуємо дані про всіх гравців, для подальшої роботи з ним. Файл отримуємо назву команди яку туди записали.

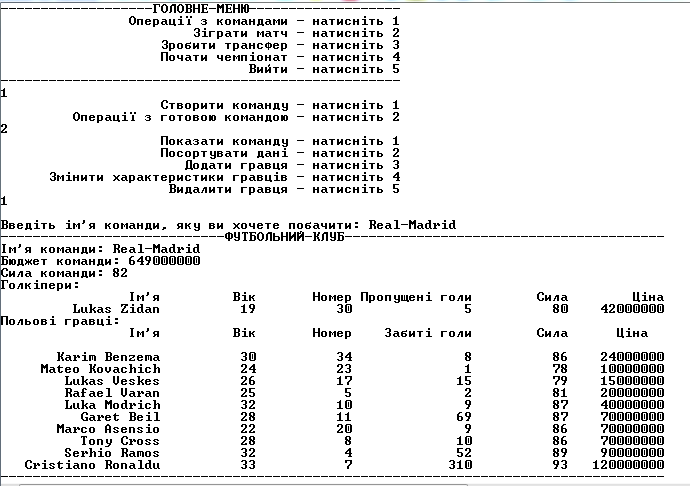


Рис. 3 Виведення інформації

Використовуючи попередньо записаний файл за допомогою функції void show\_team\_from\_file(string fTeam\_name) виводимо дані на екран.

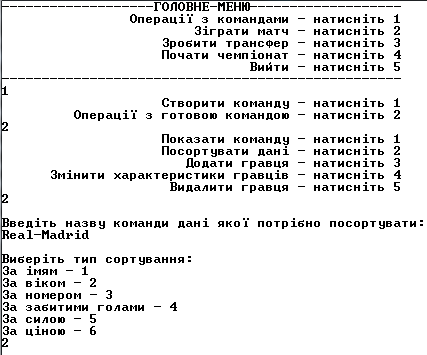


Рис. 4 Сортування даних

За допомогою функції void sort\_team() зчитуємо дані з файлу сортуємо їх

та перезаписуємо в файл.

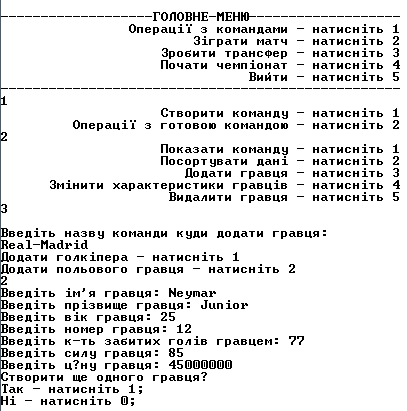


Рис. 5 Додавання нових гравців

Використовуючи функцію void add\_players() ми вводимо ще одного гравця в команду , зчитуючи з файла , додаючи в кінець та переписування файла.

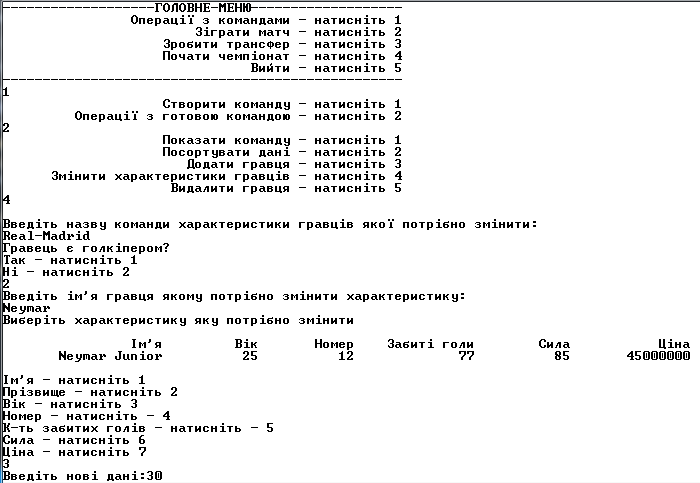


Рис. 6 Редагування властивостей гравців

Використовуючи функцію void redag\_player() ми вводимо імя гравця і вибираючи одну з його характеристик, ми можемо змінити її

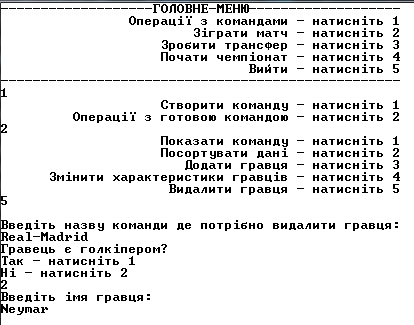


Рис. 7 Видалення гравця

Користуючись функцією void delete\_players() ми вибираємо команду де потрібно видалити гравця , вводимо його імя та видаляємо його.

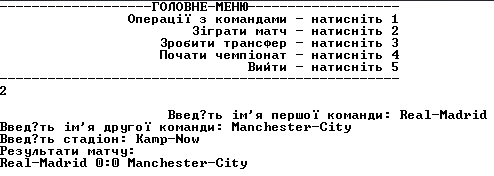


Рис. 8 Проведення матчу

Використовуючи функцію проведення матчу вибираємо команди що беруть участь у матчі вводимо назву стадіону, та отримуємо рахунок матчу.Дані про матч записуються у файл.

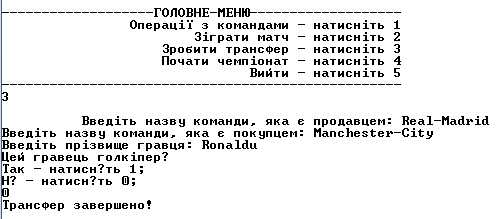


Рис. 9 Проведення трансферу

Використовуючи функцію void transfer() ми зможемо продати якійсь команді гравця.

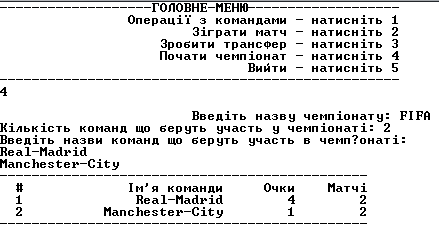


Рис. 10 Проведення чемпіонату

Користуючись функцією void Hold\_tournament() ми зможемо провести чемпіонат.

# 5. Аналіз структурної складності виконання програми

Оцінюючи структуру програми, потрібно наголосити, що наявні 3 конструкції switch, які полегшують не тільки користування програмою в ролі користувача, але як на мене, полегшують власне кодування. Ці конструкції начебто уявно відділяють блок коду від усієї програми, що покращує сприйняття й розуміння виконаних та запланованих цілей (функцій, операцій). Як висновок, можу сказати що, виконання програми на структурному рівні не є складним.

/

# Висновок

В ході курсової роботи я на практиці застосував свої знання з об’єктно-орієнтованого програмування. Поглибив розуміння основних принципів об’єктно-орієнтованої ідеології програмування. Відпрацював на практиці основні методи та засоби об’єктно-орієнтованого програмування. Навчився розробляти ієрархію класів. Практично застосував такі поняття, як інкапсуляція, наслідування, перевантаження та перевизначення функцій. Набув практичних навиків роботи.

# Література

Електронні ресурси:

Джерело№1. <https://uk.wikipedia.org/wiki/Об%27єктно-орієнтоване_програмування>

Джерело№2. <https://uk.wikipedia.org/wiki/C%2B%2B>

Джерело№3. https://www.sports.ua

Джерело№4. <https://code-live.ru/post/cpp-class-inheritance/>

* Основы информатики и программирования: Учебное пособие// Е. А. Роганов — М.: МГИУ, 2001. — Вузол VI.(рос.)
* С++. Теорія та практика : Навч. посібник / [О. Г. Трофименко, Ю. В.

Прокоп, І. Г. Швайко, Л. М. Буката та ін.] ; за ред.О. Г.Трофименко.

# Додаток А. Код програми

#include "stdafx.h"

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <vector>

#include <time.h>

using namespace std;

class Player {

protected:

string name, surname;

int age, number, strength;

long price;

public:

Player() : name("NoName"), surname("NoSurname"), number(0), age(0), strength(0), price(0) {}

Player(string fname, string fsurname, int fage, int fnumber, int fstrength, long fprice) : name(fname), surname(fsurname), number(fnumber), age(fage), strength(fstrength), price(fprice) {}

string get\_name() { return name; }

string get\_surname() { return surname; }

int get\_number() { return number; }

int get\_age() { return age; }

int get\_strength() { return strength; }

long get\_price() { return price; }

void redag\_name() {

cout << "Введiть нові данi:";

cin >> name;

}

void redag\_surname() {

cout << "Введiть новi данi:";

cin >> surname;

}

void redag\_age() {

cout << "Введiть новi данi:";

cin >> age;

}

void redag\_number() {

cout << "Введiть новi данi:";

cin >> number;

}

void redag\_strength() {

cout << "Введiть новi данi:";

cin >> strength;

}

void redag\_price() {

cout << "Введiть новi данi:";

cin >> price;

}

void delete\_player() {

name = ' ';

surname = ' ';

age = NULL;

number = NULL;

strength = NULL;

price = NULL;

}

};

class Goalkeeper : public Player {

protected:

int missed\_goal;

public:

Goalkeeper() : missed\_goal(0), Player() {}

Goalkeeper(string fname, string fsurname, int fage, int fnumber, int fstrength, long fprice, int fmissed\_goal) : Player(fname, fsurname, fage, fnumber, fstrength, fprice), missed\_goal(fmissed\_goal) {}

int get\_missed\_goal() { return missed\_goal; }

void redag\_missed\_goal() {

cout << "Введiть новi данi:";

cin >> missed\_goal;

}

void print\_data\_goalkeepers() {

cout << endl;

cout.setf(ios::right);

cout.width(20);

cout << "Iм'я"; cout.width(12);

cout << "Вiк"; cout.width(12);

cout << "Номер"; cout.width(15);

cout << "Пропущенi голи"; cout.width(12);

cout << "Cила"; cout.width(15);

cout << "Цiна" << '\n';

cout.setf(ios::right);

cout << name + ' ' + surname; cout.width(12);

cout << age; cout.width(12);

cout << number; cout.width(15);

cout << missed\_goal; cout.width(12);

cout << strength; cout.width(15);

cout << price ;

cout << endl;

}

void delete\_goalkeeper() {

missed\_goal = NULL;

delete\_player();

}

};

class Field\_Player : public Player {

protected:

int scored\_goal;

public:

Field\_Player() : scored\_goal(0), Player() {}

Field\_Player(string fname, string fsurname, int fage, int fnumber, int fstrength, long fprice, int fscored\_goal) : Player(fname, fsurname, fage, fnumber, fstrength, fprice), scored\_goal(fscored\_goal) {}

int get\_scored\_goal() { return scored\_goal; }

void redag\_scored\_goal() {

cout << "Введiть новi данi:";

cin >> scored\_goal;

}

void print\_data\_player() {

cout << endl;

cout.setf(ios::right);

cout.width(20);

cout << "Iм'я"; cout.width(12);

cout << "Вiк"; cout.width(12);

cout << "Номер"; cout.width(15);

cout << "Забитi голи"; cout.width(12);

cout << "Cила"; cout.width(15);

cout << "Цiна" << '\n';

cout.setf(ios::right);

cout.width(20);

cout << name + ' ' + surname; cout.width(12);

cout << age; cout.width(12);

cout << number; cout.width(15);

cout << scored\_goal; cout.width(12);

cout << strength; cout.width(15);

cout << price ;

cout << endl;

}

void delete\_field\_player() {

scored\_goal = NULL;

delete\_player();

}

};

class Team : public Field\_Player, public Goalkeeper {

protected:

string Team\_name;

long budget;

int team\_strength;

vector<Goalkeeper> goalkeepers;

vector<Field\_Player> players;

public:

Team() : Field\_Player(), Goalkeeper(), Team\_name("Noname"), budget(0) {}

Team(string fTeam\_name, long fbudget) : Team\_name(fTeam\_name), budget(fbudget) {}

string get\_team\_name() { return Team\_name; }

long get\_budget() { return budget; }

void set\_team\_strength() {

int goalkeepers\_strength = 0, players\_strength = 0;

if (goalkeepers.size() > 0) {

for (int i = 0; i < goalkeepers.size(); i++) {

goalkeepers\_strength += goalkeepers[i].get\_strength();

}

goalkeepers\_strength = goalkeepers\_strength / goalkeepers.size();

}

if (players.size() > 0) {

for (int i = 0; i < players.size(); i++) {

players\_strength += players[i].get\_strength();

}

players\_strength = players\_strength / players.size();

}

team\_strength = (players\_strength + goalkeepers\_strength) / 2;

}

int get\_team\_strength() { return team\_strength; }

void create\_goalkeepers() {

string fname, fsurname;

int fage, fnumber, fstrength, fmissed\_goal, ch;

long fprice;

do {

cout << "Введіть iм'я голкiпера: "; cin >> fname;

cout << "Введіть прiзвище голкiпера: "; cin >> fsurname;

cout << "Введіть вiк голкiпера: "; cin >> fage;

cout << "Введiть номер голкiпера: "; cin >> fnumber;

cout << "Введiть к-сть пропущених голiв голкiпером: "; cin >> fmissed\_goal;

cout << "Введiть силу голкiпера: "; cin >> fstrength;

cout << "Введiть цiну голкiпера "; cin >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

cout << "Створити ще одного голкiпера?\nТак - натиснiть 1;\nНi - натиснiть 0;\n"; cin >> ch;

} while (ch == 1);

}

void create\_field\_players() {

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, ch;

long fprice;

do {

cout << "Введiть iм'я гравця: "; cin >> fname;

cout << "Введiть прiзвище гравця: "; cin >> fsurname;

cout << "Введiть вiк гравця: "; cin >> fage;

cout << "Введiть номер гравця: "; cin >> fnumber;

cout << "Введiть к-ть забитих голiв гравцем: "; cin >> fscored\_goal;

cout << "Введiть силу гравця: "; cin >> fstrength;

cout << "Введiть ціну гравця: "; cin >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

cout << "Створити ще одного гравця?\nТак - натиснiть 1;\nНi - натиснiть 0;\n"; cin >> ch;

} while (ch == 1);

}

void enter\_team\_in\_file() {

fstream outfile(Team\_name + ".txt", ios::out);

if (!outfile) {

cout << "Hе можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(10);

outfile << Team\_name << '\n';

outfile.width(10);

outfile << budget << '\n';

outfile.width(10);

outfile << team\_strength << '\n';

outfile.width(10);

outfile << goalkeepers.size() << '\n';

for (int i = 0; i < goalkeepers.size(); i++) {

outfile << goalkeepers[i].get\_name() << ' ';

outfile << goalkeepers[i].get\_surname() << ' ';

outfile << goalkeepers[i].get\_age() << ' ';

outfile << goalkeepers[i].get\_number() << ' ';

outfile << goalkeepers[i].get\_missed\_goal() << ' ';

outfile << goalkeepers[i].get\_strength() << ' ';

outfile << goalkeepers[i].get\_price() << endl;

}

outfile.width(10);

outfile << players.size() << '\n';

for (int i = 0; i < players.size(); i++) {

outfile << players[i].get\_name() << ' ';

outfile << players[i].get\_surname() << ' ';

outfile << players[i].get\_age() << ' ';

outfile << players[i].get\_number() << ' ';

outfile << players[i].get\_scored\_goal() << ' ';

outfile << players[i].get\_strength() << ' ';

outfile << players[i].get\_price() << endl;

}

}

void create\_team() {

cout << "Введiть iм'я команди: "; cin >> Team\_name;

cout << "Введiть бюджет команди: "; cin >> budget;

create\_goalkeepers();

create\_field\_players();

set\_team\_strength();

enter\_team\_in\_file();

}

void show\_team\_from\_file(string fTeam\_name) {

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, fmissed\_goal, goalkeepers\_count, players\_count;

long fprice;

ifstream infile;

infile.open(fTeam\_name + ".txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> Team\_name;

infile >> budget;

infile >> team\_strength;

infile >> goalkeepers\_count;

for (int i = 0; i < goalkeepers\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fmissed\_goal;

infile >> fstrength;

infile >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

}

infile >> players\_count;

for (int i = 0; i < players\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fscored\_goal;

infile >> fstrength;

infile >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

}

for (int i = 0; i < 81; i++)

{

cout << "-";

if (i == 27)

{

cout << "ФУТБОЛЬНИЙ-КЛУБ";

i += 13;

}

}

cout << endl << "Iм'я команди: " << Team\_name << '\n';

cout << "Бюджет команди: " << budget << '\n';

cout << "Сила команди: " << team\_strength << '\n';

cout << "Голкiпери:\n";

cout.setf(ios::right);

cout.width(20);

cout << "Iм'я"; cout.width(12);

cout << "Вiк"; cout.width(12);

cout << "Номер"; cout.width(15);

cout << "Пропущенi голи"; cout.width(12);

cout << "Cила"; cout.width(12);

cout << "Цiна" << '\n';

for (int i = 0; i < goalkeepers.size(); i++) {

cout.setf(ios::right);

cout.width(20);

cout << goalkeepers[i].get\_name() + ' ' + goalkeepers[i].get\_surname(); cout.width(12);

cout << goalkeepers[i].get\_age(); cout.width(12);

cout << goalkeepers[i].get\_number(); cout.width(15);

cout << goalkeepers[i].get\_missed\_goal(); cout.width(12);

cout << goalkeepers[i].get\_strength(); cout.width(12);

cout << goalkeepers[i].get\_price() << '\n';

}

cout << "Польовi гравцi:\n";

cout.setf(ios::right);

cout.width(20);

cout << "Iм'я"; cout.width(12);

cout << "Вiк"; cout.width(12);

cout << "Номер"; cout.width(15);

cout << "3абитi голи"; cout.width(12);

cout << "Cила"; cout.width(12);

cout << "Цiна\n\n";

for (int i = 0; i < players.size(); i++) {

cout.setf(ios::right);

cout.width(20);

cout << players[i].get\_name() + ' ' + players[i].get\_surname(); cout.width(12);

cout << players[i].get\_age(); cout.width(12);

cout << players[i].get\_number(); cout.width(15);

cout << players[i].get\_scored\_goal(); cout.width(12);

cout << players[i].get\_strength(); cout.width(12);

cout << players[i].get\_price() << '\n';

}

}

for (int i = 0; i < 83; i++)

{

cout << "-";

}

infile.close();

}

void redag\_player() {

cout << "Введiть назву команди характеристики гравцiв якої потрiбно змiнити:" << endl;

string fTeam\_name;

cin >> fTeam\_name;

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, fmissed\_goal, goalkeepers\_count, players\_count;

long fprice;

ifstream infile;

infile.open(fTeam\_name + ".txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> Team\_name;

infile >> budget;

infile >> team\_strength;

infile >> goalkeepers\_count;

for (int i = 0; i < goalkeepers\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fmissed\_goal;

infile >> fstrength;

infile >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

}

infile >> players\_count;

for (int i = 0; i < players\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fscored\_goal;

infile >> fstrength;

infile >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

}

}

cout << "Гравець є голкiпером?\n" << "Так - натиснiть 1\n" << "Hi - натиснiть 2\n";

int choice;

cin >> choice;

if (choice == 2) {

cout << "Введiть iм'я гравця якому потрiбно змiнити характеристику:" << endl;

string name;

cin >> name;

cout << "Виберiть характеристику яку потрiбно змiнити\n";

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].print\_data\_player();

}

}

cout << "\nIм'я - натиснiть 1\n" << "Прiзвище - натиснiть 2\n" << "Вiк - натиснiть 3\n" << "Номер - натиснiть - 4\n" << "К-ть забитих голiв - натиснiть - 5\n" << "Сила - натиснiть 6\n" << "Цiна - натиснiть 7\n";

int choice;

cin >> choice;

switch (choice) {

case 1: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_name();

}

}

break;

}

case 2: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_surname();

}

}

break;

}

case 3: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_age();

}

}

break;

}

case 4: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_number();

}

}

break;

}

case 5: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_scored\_goal();

}

}

break;

}

case 6: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_strength();

}

}

break;

}

case 7: {

for (int i = 0; i < players\_count; i++) {

if (players[i].get\_name() == name) {

players[i].redag\_price();

}

}

break;

}

}

}

if (choice == 1) {

cout << "Введiть iм'я голкiпера якому потрiбно змiнити характеристику:" << endl;

string name;

cin >> name;

cout << "Виберiть характеристику яку потрiбно змiнити\n";

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].print\_data\_goalkeepers();

}

}

cout << "Iм'я - натиснiть 1\n" << "Прiзвище - натиснiть 2\n" << "Вiк - натиснiть 3\n" << "Номер - натиснiть - 4\n" << "К-ть пропущених голiв - натиснiть - 5\n" << "Сила - натиснiть 6\n" << "Цiна - натиснiть 7\n";

int choice;

cin >> choice;

switch (choice) {

case 1: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_name();

}

}

break;

}

case 2: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_surname();

}

}

break;

}

case 3: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_age();

}

}

break;

}

case 4: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_number();

}

}

break;

}

case 5: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_missed\_goal();

}

}

break;

}

case 6: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_strength();

}

}

break;

}

case 7: {

for (int i = 0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == name) {

goalkeepers[i].redag\_price();

}

}

break;

}

}

}

{

fstream outfile(fTeam\_name + ".txt", ios::out);

if (!outfile) {

cout << "Hе можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(10);

outfile << Team\_name << '\n';

outfile.width(10);

outfile << budget << '\n';

outfile.width(10);

outfile << team\_strength << '\n';

outfile.width(10);

outfile << goalkeepers.size() << '\n';

for (int i = 0; i<goalkeepers.size(); i++) {

outfile << goalkeepers[i].get\_name() << ' ';

outfile << goalkeepers[i].get\_surname() << ' ';

outfile << goalkeepers[i].get\_age() << ' ';

outfile << goalkeepers[i].get\_number() << ' ';

outfile << goalkeepers[i].get\_missed\_goal() << ' ';

outfile << goalkeepers[i].get\_strength() << ' ';

outfile << goalkeepers[i].get\_price() << endl;

}

outfile.width(10);

outfile << players.size() << '\n';

for (int i = 0; i<players.size(); i++) {

outfile << players[i].get\_name() << ' ';

outfile << players[i].get\_surname() << ' ';

outfile << players[i].get\_age() << ' ';

outfile << players[i].get\_number() << ' ';

outfile << players[i].get\_scored\_goal() << ' ';

outfile << players[i].get\_strength() << ' ';

outfile << players[i].get\_price() << endl;

}

}

}

void delete\_players() {

cout << "Введiть назву команди де потрiбно видалити гравця:" << endl;

string fTeam\_name;

cin >> fTeam\_name;

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, fmissed\_goal, goalkeepers\_count, players\_count;

long fprice;

ifstream infile;

infile.open(fTeam\_name + ".txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> Team\_name;

infile >> budget;

infile >> team\_strength;

infile >> goalkeepers\_count;

for (int i = 0; i < goalkeepers\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fmissed\_goal;

infile >> fstrength;

infile >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

}

infile >> players\_count;

for (int i = 0; i < players\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fscored\_goal;

infile >> fstrength;

infile >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

}

}

cout << "Гравець є голкiпером?\n" << "Так - натиснiть 1\n" << "Hi - натиснiть 2\n";

int choice;

cin >> choice;

switch (choice) {

case 1: {

cout << "Введiть iмя голкiпера:" << endl;

string player;

cin >> player;

for (int i=0; i < goalkeepers\_count; i++) {

if (goalkeepers[i].get\_name() == player)

goalkeepers.erase(goalkeepers.begin() + i);

}

}

case 2: {

cout << "Введiть iмя гравця:" << endl;

string player;

cin >> player;

for (int i=0; i < players\_count; i++) {

if (players[i].get\_name() == player)

players.erase(players.begin() + i);

}

}

}

fstream outfile(fTeam\_name + ".txt", ios::out);

if (!outfile) {

cout << "Hе можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(10);

outfile << Team\_name << '\n';

outfile.width(10);

outfile << budget << '\n';

outfile.width(10);

outfile << team\_strength << '\n';

outfile.width(10);

outfile << goalkeepers.size() << '\n';

for (int i = 0; i<goalkeepers.size(); i++) {

outfile << goalkeepers[i].get\_name() << ' ';

outfile << goalkeepers[i].get\_surname() << ' ';

outfile << goalkeepers[i].get\_age() << ' ';

outfile << goalkeepers[i].get\_number() << ' ';

outfile << goalkeepers[i].get\_missed\_goal() << ' ';

outfile << goalkeepers[i].get\_strength() << ' ';

outfile << goalkeepers[i].get\_price() << endl;

}

outfile.width(10);

outfile << players.size() << '\n';

for (int i = 0; i<players.size(); i++) {

outfile << players[i].get\_name() << ' ';

outfile << players[i].get\_surname() << ' ';

outfile << players[i].get\_age() << ' ';

outfile << players[i].get\_number() << ' ';

outfile << players[i].get\_scored\_goal() << ' ';

outfile << players[i].get\_strength() << ' ';

outfile << players[i].get\_price() << endl;

}

}

void sort\_team() {

cout << "Введiть назву команди данi якої потрiбно посортувати:" << endl;

string fTeam\_name;

cin >> fTeam\_name;

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, fmissed\_goal, goalkeepers\_count, players\_count;

long fprice;

ifstream infile;

infile.open(fTeam\_name + ".txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> Team\_name;

infile >> budget;

infile >> team\_strength;

infile >> goalkeepers\_count;

for (int i = 0; i < goalkeepers\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fmissed\_goal;

infile >> fstrength;

infile >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

}

infile >> players\_count;

for (int i = 0; i < players\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fscored\_goal;

infile >> fstrength;

infile >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

}

}

cout <<endl<< "Виберiть тип сортування:\n" << "За iмям - 1\n" << "За вiком - 2\n" << "За номером - 3\n" << "За забитими голами - 4\n" << "За силою - 5\n" << "За цiною - 6\n";

int choice;

cin >> choice;

switch (choice) {

case 1: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_name() > players[i + 1].get\_name())

swap(players[i], players[i + 1]);

}

}

break;

}

case 2: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_age() > players[i + 1].get\_age())

swap(players[i], players[i + 1]);

}

}

break;

}

case 3: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_number() > players[i + 1].get\_number())

swap(players[i], players[i + 1]);

}

}

break;

}

case 4: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_scored\_goal() > players[i + 1].get\_scored\_goal())

swap(players[i], players[i + 1]);

}

}

break;

}

case 5: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_strength() > players[i + 1].get\_strength())

swap(players[i], players[i + 1]);

}

}

break;

}

case 6: {

for (int j = 0; j < players\_count; j++) {

for (int i = 0; i < players\_count - 1; i++) {

if (players[i].get\_price() > players[i + 1].get\_price())

swap(players[i], players[i + 1]);

}

}

break;

}

}

{

fstream outfile(fTeam\_name + ".txt", ios::out);

if (!outfile) {

cout << "Hе можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(10);

outfile << Team\_name << '\n';

outfile.width(10);

outfile << budget << '\n';

outfile.width(10);

outfile << team\_strength << '\n';

outfile.width(10);

outfile << goalkeepers.size() << '\n';

for (int i = 0; i<goalkeepers.size(); i++) {

outfile << goalkeepers[i].get\_name() << ' ';

outfile << goalkeepers[i].get\_surname() << ' ';

outfile << goalkeepers[i].get\_age() << ' ';

outfile << goalkeepers[i].get\_number() << ' ';

outfile << goalkeepers[i].get\_missed\_goal() << ' ';

outfile << goalkeepers[i].get\_strength() << ' ';

outfile << goalkeepers[i].get\_price() << endl;

}

outfile.width(10);

outfile << players.size() << '\n';

for (int i = 0; i<players.size(); i++) {

outfile << players[i].get\_name() << ' ';

outfile << players[i].get\_surname() << ' ';

outfile << players[i].get\_age() << ' ';

outfile << players[i].get\_number() << ' ';

outfile << players[i].get\_scored\_goal() << ' ';

outfile << players[i].get\_strength() << ' ';

outfile << players[i].get\_price() << endl;

}

}

}

void transfer() {

string p\_surname = "", p\_name = "", t1\_name, t2\_name, fname1, fsurname1, fname2, fsurname2;

int p\_number = 0, p\_age = 0, p\_goals = 0, p\_strength = 0, ch, g\_count1, p\_count1, g\_count2, p\_count2, fage1, fage2, fnumber1, fnumber2, fmissed\_goal1, fmissed\_goal2, fscored\_goal1, fscored\_goal2, fstrength1, fstrength2;

long p\_price = 0, fprice1, fprice2;

string Team\_name1;

long budget1;

int team\_strength1;

vector<Goalkeeper> goalkeepers1;

vector<Field\_Player> players1;

string Team\_name2;

long budget2;

int team\_strength2;

vector<Goalkeeper> goalkeepers2;

vector<Field\_Player> players2;

cout << "Введiть назву команди, яка є продавцем: "; cin >> t1\_name;

cout << "Введiть назву команди, яка є покупцем: "; cin >> t2\_name;

cout << "Введiть прiзвище гравця: "; cin >> p\_surname;

cout << "Цей гравець голкiпер?\nТак - натисніть 1;\nНі - натисніть 0;\n"; cin >> ch;

ifstream infile1;

infile1.open(t1\_name + ".txt");

if (!infile1) cout << "Не можу вiдкрити файл\n";

if (infile1.is\_open()) {

infile1 >> Team\_name1;

infile1 >> budget1;

infile1 >> team\_strength1;

infile1 >> g\_count1;

for (int i = 0; i<g\_count1; i++) {

infile1 >> fname1;

infile1 >> fsurname1;

infile1 >> fage1;

infile1 >> fnumber1;

infile1 >> fmissed\_goal1;

infile1 >> fstrength1;

infile1 >> fprice1;

goalkeepers1.push\_back(Goalkeeper(fname1, fsurname1, fage1, fnumber1, fstrength1, fprice1, fmissed\_goal1));

}

infile1 >> p\_count1;

for (int i = 0; i<p\_count1; i++) {

infile1 >> fname1;

infile1 >> fsurname1;

infile1 >> fage1;

infile1 >> fnumber1;

infile1 >> fscored\_goal1;

infile1 >> fstrength1;

infile1 >> fprice1;

players1.push\_back(Field\_Player(fname1, fsurname1, fage1, fnumber1, fstrength1, fprice1, fscored\_goal1));

}

if (ch == 1) {

for (int i = 0; i<goalkeepers.size(); i++) {

if (p\_surname == goalkeepers1[i].get\_surname()) {

p\_name = goalkeepers1[i].get\_name();

p\_age = goalkeepers1[i].get\_age();

p\_number = goalkeepers1[i].get\_number();

p\_strength = goalkeepers1[i].get\_strength();

p\_goals = goalkeepers1[i].get\_missed\_goal();

p\_price = goalkeepers1[i].get\_price();

goalkeepers1.erase(goalkeepers1.begin() + i);

}

}

}

if (ch == 0) {

for (int i = 0; i<players1.size(); i++) {

if (p\_surname == players1[i].get\_surname()) {

p\_name = players1[i].get\_name();

p\_age = players1[i].get\_age();

p\_number = players1[i].get\_number();

p\_strength = players1[i].get\_strength();

p\_goals = players1[i].get\_scored\_goal();

p\_price = players1[i].get\_price();

players1.erase(players1.begin() + i);

}

}

}

}infile1.close();

fstream outfile1(t1\_name + ".txt", ios::out);

if (!outfile1) {

cout << "Не можу вiдкрити файл" << endl;

}

outfile1.setf(ios::left);

outfile1.width(10);

outfile1 << Team\_name1 << '\n';

budget1 += p\_price;

outfile1.width(10);

outfile1 << budget1 << '\n';

outfile1.width(10);

outfile1 << team\_strength1 << '\n';

outfile1.width(10);

outfile1 << goalkeepers1.size() << '\n';

for (int i = 0; i<goalkeepers1.size(); i++) {

outfile1 << goalkeepers1[i].get\_name() << ' ';

outfile1 << goalkeepers1[i].get\_surname() << ' ';

outfile1 << goalkeepers1[i].get\_age() << ' ';

outfile1 << goalkeepers1[i].get\_number() << ' ';

outfile1 << goalkeepers1[i].get\_missed\_goal() << ' ';

outfile1 << goalkeepers1[i].get\_strength() << ' ';

outfile1 << goalkeepers1[i].get\_price() << endl;

}

outfile1.width(10);

outfile1 << players1.size() << '\n';

for (int i = 0; i<players1.size(); i++) {

outfile1 << players1[i].get\_name() << ' ';

outfile1 << players1[i].get\_surname() << ' ';

outfile1 << players1[i].get\_age() << ' ';

outfile1 << players1[i].get\_number() << ' ';

outfile1 << players1[i].get\_scored\_goal() << ' ';

outfile1 << players1[i].get\_strength() << ' ';

outfile1 << players1[i].get\_price() << endl;

}

outfile1.close();

ifstream infile2;

infile2.open(t2\_name + ".txt");

if (!infile2) cout << "Не можу вiдкрити файл\n";

if (infile2.is\_open()) {

infile2 >> Team\_name2;

infile2 >> budget2;

infile2 >> team\_strength2;

infile2 >> g\_count2;

for (int i = 0; i<g\_count2; i++) {

infile2 >> fname2;

infile2 >> fsurname2;

infile2 >> fage2;

infile2 >> fnumber2;

infile2 >> fmissed\_goal2;

infile2 >> fstrength2;

infile2 >> fprice2;

goalkeepers2.push\_back(Goalkeeper(fname2, fsurname2, fage2, fnumber2, fstrength2, fprice2, fmissed\_goal2));

}

infile2 >> p\_count2;

for (int i = 0; i<p\_count2; i++) {

infile2 >> fname2;

infile2 >> fsurname2;

infile2 >> fage2;

infile2 >> fnumber2;

infile2 >> fscored\_goal2;

infile2 >> fstrength2;

infile2 >> fprice2;

players2.push\_back(Field\_Player(fname2, fsurname2, fage2, fnumber2, fstrength2, fprice2, fscored\_goal2));

}

if (ch == 1) {

goalkeepers2.push\_back(Goalkeeper(p\_name, p\_surname, p\_age, p\_number, p\_strength, p\_price, p\_goals));

}

if (ch == 0) {

players2.push\_back(Field\_Player(p\_name, p\_surname, p\_age, p\_number, p\_strength, p\_price, p\_goals));

}

}infile2.close();

fstream outfile2(t2\_name + ".txt", ios::out);

if (!outfile2) {

cout << "Не можу вiдкрити файл" << endl;

}

outfile2.setf(ios::left);

outfile2.width(10);

outfile2 << Team\_name2 << '\n';

budget2 -= p\_price;

outfile2.width(10);

outfile2 << budget2 << '\n';

outfile2.width(10);

outfile2 << team\_strength2 << '\n';

outfile2.width(10);

outfile2 << goalkeepers2.size() << '\n';

for (int i = 0; i<goalkeepers2.size(); i++) {

outfile2 << goalkeepers2[i].get\_name() << ' ';

outfile2 << goalkeepers2[i].get\_surname() << ' ';

outfile2 << goalkeepers2[i].get\_age() << ' ';

outfile2 << goalkeepers2[i].get\_number() << ' ';

outfile2 << goalkeepers2[i].get\_missed\_goal() << ' ';

outfile2 << goalkeepers2[i].get\_strength() << ' ';

outfile2 << goalkeepers2[i].get\_price() << endl;

}

outfile2.width(10);

outfile2 << players2.size() << '\n';

for (int i = 0; i<players2.size(); i++) {

outfile2 << players2[i].get\_name() << ' ';

outfile2 << players2[i].get\_surname() << ' ';

outfile2 << players2[i].get\_age() << ' ';

outfile2 << players2[i].get\_number() << ' ';

outfile2 << players2[i].get\_scored\_goal() << ' ';

outfile2 << players2[i].get\_strength() << ' ';

outfile2 << players2[i].get\_price() << endl;

}

outfile2.close();

cout << "Трансфер завершено!\n";

}

void add\_players() {

cout << "Введiть назву команди куди додати гравця:" << endl;

string fTeam\_name;

cin >> fTeam\_name;

string fname, fsurname;

int fage, fnumber, fstrength, fscored\_goal, fmissed\_goal, goalkeepers\_count, players\_count;

long fprice;

ifstream infile;

infile.open(fTeam\_name + ".txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> Team\_name;

infile >> budget;

infile >> team\_strength;

infile >> goalkeepers\_count;

for (int i = 0; i < goalkeepers\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fmissed\_goal;

infile >> fstrength;

infile >> fprice;

goalkeepers.push\_back(Goalkeeper(fname, fsurname, fage, fnumber, fstrength, fprice, fmissed\_goal));

}

infile >> players\_count;

for (int i = 0; i < players\_count; i++) {

infile >> fname;

infile >> fsurname;

infile >> fage;

infile >> fnumber;

infile >> fscored\_goal;

infile >> fstrength;

infile >> fprice;

players.push\_back(Field\_Player(fname, fsurname, fage, fnumber, fstrength, fprice, fscored\_goal));

}

}

cout << "Додати голкiпера - натиснiть 1" << endl;

cout << "Додати польового гравця - натиснiть 2" << endl;

int choice;

cin >> choice;

switch (choice) {

case 1: {

create\_goalkeepers();

}

case 2: {

create\_field\_players();

}

}

fstream outfile(fTeam\_name + ".txt", ios::out);

if (!outfile) {

cout << "Hе можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(10);

outfile << Team\_name << '\n';

outfile.width(10);

outfile << budget << '\n';

outfile.width(10);

outfile << team\_strength << '\n';

outfile.width(10);

outfile << goalkeepers.size() << '\n';

for (int i = 0; i<goalkeepers.size(); i++) {

outfile << goalkeepers[i].get\_name() << ' ';

outfile << goalkeepers[i].get\_surname() << ' ';

outfile << goalkeepers[i].get\_age() << ' ';

outfile << goalkeepers[i].get\_number() << ' ';

outfile << goalkeepers[i].get\_missed\_goal() << ' ';

outfile << goalkeepers[i].get\_strength() << ' ';

outfile << goalkeepers[i].get\_price() << endl;

}

outfile.width(10);

outfile << players.size() << '\n';

for (int i = 0; i<players.size(); i++) {

outfile << players[i].get\_name() << ' ';

outfile << players[i].get\_surname() << ' ';

outfile << players[i].get\_age() << ' ';

outfile << players[i].get\_number() << ' ';

outfile << players[i].get\_scored\_goal() << ' ';

outfile << players[i].get\_strength() << ' ';

outfile << players[i].get\_price() << endl;

}

}

};

class Match : public Team {

protected:

string team1\_name, team2\_name, stadium;

int team1\_point, team2\_point, team1\_goal, team2\_goal, team1\_match, team2\_match;

public:

Match() : Team(), team1\_name("NoName"), team2\_name("NoName"), stadium("NoName"), team1\_point(0), team2\_point(0), team1\_goal(0), team2\_goal(0), team1\_match(0), team2\_match(0) {}

Match(string t1n, string t2n, string stad) : team1\_name(t1n), team2\_name(t2n), stadium(stad) {}

string get\_team1\_name() { return team1\_name; }

string get\_team2\_name() { return team2\_name; }

string get\_stadium() { return stadium; }

int get\_team1\_point() { return team1\_point; }

int get\_team2\_point() { return team2\_point; }

int get\_team1\_goal() { return team1\_goal; }

int get\_team2\_goal() { return team2\_goal; }

int get\_team1\_match() { return team1\_match; }

int get\_team2\_match() { return team2\_match; }

void play\_match() {

srand(time(NULL));

long budget1, budget2, t1\_koef, t2\_koef;

int t1\_strength, t2\_strength;

cout << "Введіть iм'я першої команди: "; cin >> team1\_name;

cout << "Введіть iм'я другої команди: "; cin >> team2\_name;

cout << "Введіть стадiон: "; cin >> stadium;

ifstream infile1;

infile1.open(team1\_name + "\_matches.txt");

if (!infile1) {

ifstream infile11;

infile11.open(team1\_name + ".txt");

if (!infile11) cout << "Не можу вдiкрити файл\n";

if (infile11.is\_open()) {

infile11 >> team1\_name;

infile11 >> budget1;

infile11 >> t1\_strength;

team1\_match = 0;

team1\_point = 0;

}

}

if (infile1.is\_open()) {

infile1 >> team1\_name;

infile1 >> t1\_strength;

infile1 >> team1\_match;

infile1 >> team1\_point;

}

ifstream infile2;

infile2.open(team2\_name + "\_matches.txt");

if (!infile2) {

ifstream infile22;

infile22.open(team2\_name + ".txt");

if (!infile22) cout << "Не можу вiдкрити файл\n";

if (infile22.is\_open()) {

infile22 >> team2\_name;

infile22 >> budget2;

infile22 >> t2\_strength;

team2\_match = 0;

team2\_point = 0;

}

}

if (infile2.is\_open()) {

infile2 >> team2\_name;

infile2 >> t2\_strength;

infile2 >> team2\_match;

infile2 >> team2\_point;

}

t1\_koef = t1\_strength \* 1 + rand() % 10;

t2\_koef = t2\_strength \* 1 + rand() % 10;

if (t1\_koef > t2\_koef) {

team1\_goal = 1 + rand() % 7;

team2\_goal = 0 + rand() % (team1\_goal - 1);

team1\_point += 3;

}

if (t1\_koef == t2\_koef) {

team1\_goal = 0 + rand() % 7;

team2\_goal = team1\_goal;

team1\_point += 1;

team2\_point += 1;

}

if (t1\_koef < t2\_koef) {

team2\_goal = 1 + rand() % 7;

team1\_goal = 0 + rand() % (team2\_goal - 1);

team2\_point += 3;

}

team1\_match++;

team2\_match++;

cout << "Результати матчу:\n" << team1\_name << " " << team1\_goal << ":" << team2\_goal << " " << team2\_name << endl;

fstream outfile1(team1\_name + "\_matches.txt", ios::out);

if (!outfile1) {

cout << "Не можу вiдкрити файл" << endl;

}

outfile1.setf(ios::left);

outfile1.width(10);

outfile1 << team1\_name << '\n';

outfile1.width(10);

outfile1 << t1\_strength << '\n';

outfile1.width(10);

outfile1 << team1\_match << '\n';

outfile1.width(10);

outfile1 << team1\_point << endl;

fstream outfile2(team2\_name + "\_matches.txt", ios::out);

if (!outfile2) {

cout << "Не можу вiдкрити файл" << endl;

}

outfile2.setf(ios::left);

outfile2.width(10);

outfile2 << team2\_name << '\n';

outfile2.width(10);

outfile2 << t2\_strength << '\n';

outfile2.width(10);

outfile2 << team2\_match << '\n';

outfile2.width(10);

outfile2 << team2\_point << endl;

}

};

class Tournament : public Match {

protected:

string tournament\_name;

int team\_count;

public:

Tournament() : Match(), tournament\_name("NoName"), team\_count(0) {}

void Hold\_tournament() {

string temp;

string tteam\_name;

int tteam\_strength, tteam\_matches, tteam\_points;

vector<string> Teams;

vector<int> Teams\_matches;

vector<int> Teams\_points;

cout << "Введiть назву чемпiонату: "; cin >> tournament\_name;

cout << "Кiлькiсть команд що беруть участь у чемпiонатi: "; cin >> team\_count;

cout << "Введiть назви команд що беруть участь в чемпіонатi:\n";

for (int i = 0; i<team\_count; i++) {

cin >> tteam\_name;

Teams.push\_back(tteam\_name);

}

for (int i = 0; i<team\_count; i++) {

ifstream infile;

infile.open(Teams[i] + "\_matches.txt");

if (!infile) cout << "Не можу вiдкрити файл\n";

if (infile.is\_open()) {

infile >> temp;

infile >> tteam\_strength;

infile >> tteam\_matches;

infile >> tteam\_points;

}

Teams\_matches.push\_back(tteam\_matches);

Teams\_points.push\_back(tteam\_points);

}

for (int i = team\_count; i>0; i--) {

for (int j = 0, k = 1; k<i; j++, k++) {

if (Teams\_points[j] < Teams\_points[k]) {

int swap1 = Teams\_points[j];

Teams\_points[j] = Teams\_points[k];

Teams\_points[k] = swap1;

int swap2 = Teams\_matches[j];

Teams\_matches[j] = Teams\_matches[k];

Teams\_matches[k] = swap2;

string swap3 = Teams[j];

Teams[j] = Teams[k];

Teams[k] = swap3;

}

}

}

for (int i = 0; i<46; i++)

{

cout << "-";

}

cout << endl << setw(3) << "#" << setw(25) << "Iм'я команди" << setw(9) << "Очки" << setw(9) << "Матчi" << endl;

for (int i = 0; i<team\_count; i++) {

cout.width(3);

cout << i + 1;

cout.width(25);

cout << Teams[i];

cout.width(9);

cout << Teams\_points[i];

cout.width(9);

cout << Teams\_matches[i];

cout << endl;

}

for (int i = 0; i<46; i++)

{

cout << "-";

}

fstream outfile(tournament\_name + "\_tournament.txt", ios::out);

if (!outfile) {

cout << "Не можу вiдкрити файл" << endl;

}

outfile.setf(ios::left);

outfile.width(9);

outfile << tournament\_name << endl;

outfile.width(9);

outfile << team\_count << endl;

for (int i = 0; i<team\_count; i++) {

outfile.width(9);

outfile << Teams[i] << ' ';

outfile.width(9);

outfile << Teams\_points[i] << ' ';

outfile.width(9);

outfile << Teams\_matches[i] << endl;

}

}

};

int main() {

setlocale(LC\_ALL, "Ukrainian");

cout.setf(ios::right);

while (true) {

label:

for (int i = 0; i < 19; i++) {

cout << "-";

}

cout << "ГОЛОВНЕ-МЕНЮ";

for (int i = 0; i < 19; i++) {

cout << "-";

}

cout << endl;

cout.width(50);

cout << "Операцiї з командами - натиснiть 1" << endl; cout.width(50);

cout << "Зiграти матч - натиснiть 2" << endl; cout.width(50);

cout << "Зробити трансфер - натиснiть 3" << endl; cout.width(50);

cout << "Почати чемпiонат - натиснiть 4" << endl; cout.width(50);

cout << "Вийти - натиснiть 5" << endl;

for (int i = 0; i < 50; i++) {

cout << "-";

}

cout << endl;

cout.width(50);

int choice;

cin >> choice;

if (choice == 1) {

cout << "Створити команду - натиснiть 1" << endl; cout.width(50);

cout << "Операцiї з готовою командою - натиснiть 2" << endl; cout.width(50);

cin >> choice;

if (choice == 1) {

cout << endl;

Team obj;

obj.create\_team();

cout << endl;

}

if (choice == 2) {

cout << "Показати команду - натиснiть 1" << endl; cout.width(50);

cout << "Посортувати данi - натиснiть 2" << endl; cout.width(50);

cout << "Додати гравця - натиснiть 3" << endl; cout.width(50);

cout << "Змiнити характеристики гравцiв - натиснiть 4" << endl; cout.width(50);

cout << "Видалити гравця - натиснiть 5" << endl;

cin >> choice;

if (choice == 1) {

string fTeam\_name;

cout << endl;

cout << "Введiть iм'я команди, яку ви хочете побачити: "; cin >> fTeam\_name;

Team obj;

obj.show\_team\_from\_file(fTeam\_name);

cout << endl;

}

if (choice == 2) {

cout << endl;

Team obj;

obj.sort\_team();

cout << endl;

}

if (choice == 3) {

cout << endl;

Team obj;

obj.add\_players();

cout << endl;

}

if (choice == 4) {

cout << endl;

Team obj;

obj.redag\_player();

cout << endl;

}

if (choice == 5) {

cout << endl;

Team obj;

obj.delete\_players();

cout << endl;

}

}

goto label;

}

if (choice == 2) {

cout << endl;

Match obj;

obj.play\_match();

cout << endl;

}

if (choice == 3) {

cout << endl;

Team obj;

obj.transfer();

cout << endl;

}

if (choice == 4) {

cout << endl;

Tournament obj;

obj.Hold\_tournament();

cout << endl;

}

if (choice == 5) {

break;

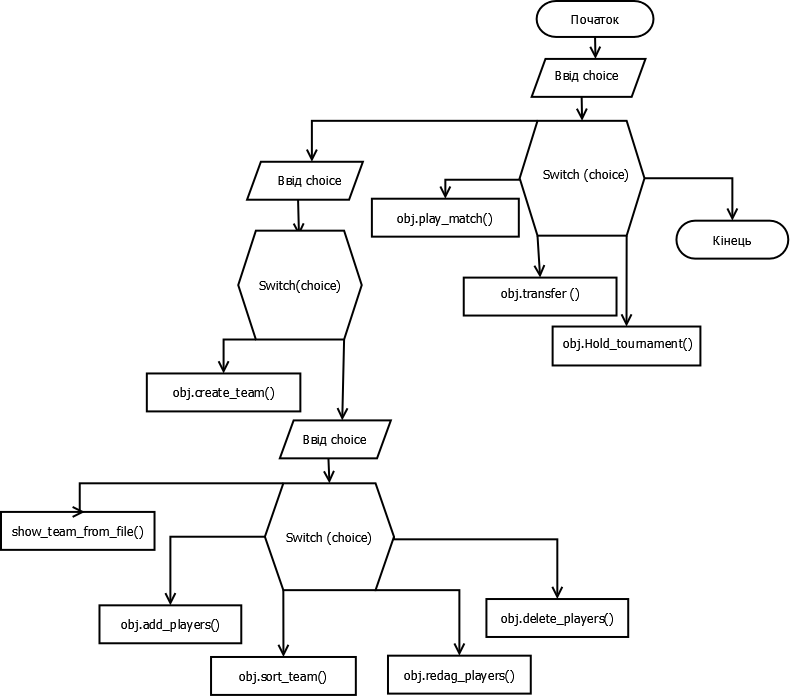
}

}

return 0;

}

# Додаток Б. Блок схема алгоритму



# Додаток В. Діаграмма класів