

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ  
імені ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ  
СИСТЕМ

### **КУРСОВА РОБОТА**

з дисципліни “Основи програмування”

на тему: *“Довідник фаната”*

Студентки 1 курсу групи КП-23

Спеціальність 121 Інженерія  
*програмного забезпечення*

*Середа Марія Володимирівна*

Київ – 2023 рік

ФПМ НТУУ “КПІ”					
Кафедра		<i>Програмного забезпечення комп’ютерних систем</i>			
Дисципліна		<i>Основи програмування</i>			
Галузь знань		<i>12 Інформаційні технології</i>			
Курс	<i>перший</i>	Група	<i>КП-23</i>	Семестр	<i>другий</i>

### ЗАВДАННЯ

на курсовий проект (роботу) студента

<i>Середа Марія Володимирівна</i>	
1. Тема проекту (роботи)	<i>“Довідник фаната”</i>
2. Строк здачі студентом закінченого проекту (роботи)	<i>08.06.2023 р.</i>
3. Вихідні дані до проекту (роботи)	<i>База спортсменок: анкетні та антропологічні дані, громадянство, вид та категорія спорту, клуб або команда, дані про особисті рекорди та інше. Вибір за довільною ознакою. Пошук рекордсменки в заданому виді спорту.</i>
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)	
<i>1. Постановка задачі.</i>	
<i>2. Метод розв’язку задачі.</i>	
<i>3. Загальна блок-схема алгоритму та опис алгоритму.</i>	
<i>4. Опис програмного продукту.</i>	
<i>5. Результати роботи.</i>	
<i>6. Висновки.</i>	
<i>7. Список використаної літератури.</i>	
<i>Додаток А. Текст програми.</i>	
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)	
<i>1. Загальна блок-схема алгоритму.</i>	
<i>2. Ілюстрації роботи програми.</i>	
6. Дата видачі завдання	<i>13.05.2023 р.</i>

## ЗМІСТ

ВСТУП.....	4
1. РОЗДІЛ 1 Постановка задачі.....	7
1.1. Огляд існуючих підходів до розв’язання поставленої задачі.....	7
1.2. Уточнена постановка задачі на розробку програмного забезпечення.....	7
2. РОЗДІЛ 2 Розробка програмного продукту.....	9
2.1. Метод розв’язку задачі.....	9
2.2. Алгоритм розв’язку задачі.....	13
3. РОЗДІЛ 3 Опис розробленого програмного продукту.....	15
3.1. Опис головних структур і змінних програми.....	15
3.2. Опис головних функцій програми.....	15
3.3. Опис інтерфейсу.....	21
ВИСНОВКИ.....	24
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ.....	25
Додаток А Текст програми	

## **ВСТУП**

### **Актуальність**

Тема “Довідник фаната” є важливою та актуальною з кількох причин.

По-перше, в сучасному спорті дослідження та аналіз даних стають все більш важливими для розвитку спортивних стратегій, тренувань та виявлення потенційних талантів. База спортсменок, що містить анкетні і антропологічні дані, громадянство, вид та категорію спорту, клуб або команду, а також дані про особисті рекорди, є цінним інструментом для проведення досліджень у спортивній сфері.

По-друге, вивчення характеристик спортсменок за довільною ознакою може мати значний вплив на різні аспекти спортивного життя. Наприклад, вибірка спортсменок за громадянством може допомогти зрозуміти динаміку розвитку спорту в певній країні та виявити потенційні області для подальшого розвитку. Вибірка спортсменок за видом спорту може допомогти визначити найефективніші методи тренувань та підвищення результативності в конкретних дисциплінах.

По-третє, пошук рекордсмена в заданому виді спорту є цікавим інструментом для вивчення досягнень спортсменок, встановлення нових стандартів і мотивації молодих спортсменок до досягнення високих результатів. Аналізуючи дані про рекордсменів, можна виявити фактори, які сприяли їх успіху, і використовувати ці знання для покращення тренувань, стратегій та підготовки нового покоління спортсменок.

Таким чином, вибір даної теми є важливим і актуальним у контексті розвитку спортивної науки та практики. Вивчення анкетних і антропологічних даних спортсменок, їх громадянства, виду та категорії спорту, клубу або команди, а також інформації про особисті рекорди, дозволить зрозуміти широкий спектр аспектів спортивного життя та сприяти покращенню тренувань, розвитку стратегій і підготовки спортсменок.

### **Мета**

Метою є розробка та реалізація закінченого програмного продукту для створення та управління базою даних спортсменок. Головним завданням цього програмного продукту є збір, організація та аналіз інформації про спортсменок, включаючи їх анкетні і антропологічні дані, громадянство, вид і категорію спорту, клуб або команду, а також дані про особисті рекорди.

Крім того, програмний продукт має забезпечувати можливість пошуку та фільтрації спортсменок за певними ознаками, що дозволить проводити аналіз та порівняння спортсменок у різних контекстах. Наприклад, можливість знаходити рекордсменів в заданому виді спорту допоможе виявити найкращих представників даної дисципліни та дослідити фактори, які сприяють досягненню високих результатів.

Основною метою програмного продукту є полегшення роботи спортивних організацій, тренерів, науковців та інших зацікавлених осіб, які займаються вивченням, розвитком та підготовкою спортсменок. Завдяки цьому програмному продукту вони зможуть зручно збирати, організовувати та аналізувати дані про спортсменок, що сприятиме покращенню тренувань, розвитку стратегій та здійсненню більш обґрунтованих рішень у спортивній діяльності.

### **Практичне значення**

Прикладне значення одержаних результатів у випадку довідника фаната з базою спортсменок може мати наступні застосування:

1. Довідник для фанатів: Розроблений програмний продукт може бути використаний як довідник для фанатів спорту. Він надасть користувачам доступ до анкетних та антропологічних даних спортсменок, таких як вік, громадянство, вид та категорія спорту, клуб або команда, особисті рекорди та інше. Фанати зможуть швидко знайти інформацію про своїх улюблених спортсменок і дізнатися більше про їх досягнення.
2. Аналіз спортивних даних: За допомогою програмного продукту можна здійснювати аналіз даних спортсменок та їхніх рекордів. Наприклад, можна провести порівняльний аналіз результатів різних спортсменок у заданому виді спорту, визначити тенденції в розвитку досягнень, встановити нові рекорди тощо. Це може бути корисно для тренерів, науковців або журналістів, які цікавляться статистикою та аналізом результатів у спорті.
3. Пошук рекордсменів: Програмний продукт може надати можливість здійснювати пошук рекордсмена в заданому виді спорту. Користувачі зможуть швидко знайти найкращі досягнення спортсменок у різних категоріях та порівняти їх результати. Це дозволить виявити найуспішніших спортсменок та використовувати цю інформацію для різних цілей, наприклад, для підготовки спортивних звітів, статей або статистичних досліджень.

Використання програмного продукту може значно полегшити доступ до інформації про спортсменок та допомогти в аналізі результатів у спорті.

### **Використане програмне забезпечення**

При виконанні роботи було використане таке програмне забезпечення: стандартне середовище розробки “Visual Studio 2022”; операційна система “Windows 11”; веб-браузер “Google Chrome” для роботи з вебсайтом <https://learn.microsoft.com/en-us/dotnet/csharp>; текстовий редактор “Google Документи” для підготовки та оформлення курсової роботи.

### **Структура роботи**

Робота складається зі вступу, трьох розділів, висновків, додатків та списку використаних джерел.

# **ОСНОВНА ЧАСТИНА**

## **РОЗДІЛ 1 ПОСТАНОВКА ЗАДАЧІ**

### **1.1. Огляд існуючих підходів до розв'язання поставленої задачі**

Огляд існуючих підходів до розв'язання поставленої задачі довідника фаната з базою спортсменок може включати наступні елементи:

- Традиційні довідники та бази даних: Існують традиційні довідники та бази даних, які містять інформацію про спортсменок, включаючи анкетні дані, рекорди та інші деталі. Ці довідники можуть бути представлені у формі паперових книг або електронних таблиць. Однак, пошук і аналіз даних у таких довідниках може бути обмеженим та часомірним.
- Веб-портали та сайти професійних спортивних організацій: Багато професійних спортивних організацій мають веб-портали або сайти, де надається інформація про спортсменок, їхні досягнення та статистика. Ці ресурси можуть містити бази даних з анкетними даними, фотографіями, відеоматеріалами тощо. Однак, доступ до цих даних може бути обмежений, а пошук та аналіз вимагати ручного втручання.
- Спеціалізовані спортивні додатки та платформи: На сьогоднішній день існує значна кількість спеціалізованих спортивних додатків та платформ, які надають інформацію про спортсменок, їхні рекорди та статистику. Деякі з цих додатків використовують технології штучного інтелекту для аналізу та візуалізації даних. Однак, багато з них мають обмежений функціонал та доступність для використання.
- Комбінація власних даних та існуючих ресурсів: Інший підхід полягає у створенні власної бази даних з анкетними та антропологічними даними про спортсменок, а також у поєднанні цих даних з існуючими ресурсами, наприклад, веб-порталами професійних організацій або спортивними додатками. Такий підхід може дати змогу зібрати повнішу та більш розширену базу даних, а також забезпечити більший контроль над процесом пошуку та аналізу даних.

### **1.2. Уточнена постановка задачі на розробку програмного забезпечення**

Вибраний варіант реалізації довідника фаната з базою спортсменок на основі існуючих методів та алгоритмів є доцільним з кількох причин:

- Ефективність: З використанням текстового файлу формату CV та алгоритмів пошуку, сортування та фільтрації довідник фаната може забезпечувати швидкий доступ до інформації про спортсменок. Це дозволяє знайти необхідну інформацію швидко та зручно.
- Гнучкість: Використання текстового файлу формату CV дозволяє легко додавати, оновлювати та видаляти інформацію про спортсменок. Крім того, застосування алгоритмів пошуку та фільтрації дає можливість користувачам знайти спортсменок за різними критеріями, що робить довідник фаната більш універсальним і зручним у використанні.
- Масштабованість: Застосування текстового файлу формату CV дозволяє розширювати довідник фаната, додавати нові види спорту, вносити зміни до анкетних та антропологічних даних спортсменок. Це дає можливість підтримувати актуальну і повну інформацію про спортсменок та їх досягнення.

Детальний опис постановки задачі:

- Створення бази даних: Створити текстовий файл формату CV, що зберігатиме інформацію про спортсменок. Він повинен містити анкетні дані (ім'я, рік народження, громадянство), антропологічні дані (зріст, вага), вид спорту, категорія спорту, клуб або команда, особисті рекорди.
- Реалізація функції пошуку спортсменок: Розробити алгоритми пошуку, що дозволять користувачам знаходити спортсменок за різними критеріями, такими як вид спорту, категорія, клуб, громадянство тощо. Результати пошуку повинні бути точними та швидкими.
- Забезпечення безпеки: Захистити дані спортсменок в текстовому файлі формату CV від несанкціонованих змін.

Вимоги до програмних та технічних засобів:

- Мова програмування: Реалізація має бути здійснена за допомогою мови програмування C#, оскільки ця мова підтримує об'єктно-орієнтоване програмування.
- База даних: Для збереження інформації про спортсменок рекомендується використовувати текстовий файл формату CV або базу даних MySQL. Вона повинна бути зручною для маніпулювання даними та забезпечувати ефективний доступ до інформації.
- Консольний інтерфейс: Реалізувати консольний інтерфейс, що дає змогу зручно взаємодіяти з довідником фаната.
- Безпека: Забезпечити захист даних спортсменок шляхом використання інкапсуляції даних.



## РОЗДІЛ 2 РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

### 2.1. Метод розв'язку задачі

У програмному коді надано базовий клас *Sportsman* і похідні класи, які представляють різні види спортсменок, а саме: клас *Swimmer* і похідні від нього класи *FiftyMeter\_Swimmer* та *OneHundredMeter\_Swimmer*, клас *FigureSkater* і похідні від нього класи *CombinedTotal\_FigureSkater*, *ShortProgram\_FigureSkater* та *FreeSkating\_FigureSkater*, клас *Athlete* і похідні від нього класи *Decathlete*, *ShotPut\_Athlete*, *PoleVault\_Athlete*, *MarathonRunner* та *Jump\_Athlete* (і похідні від нього класи *LongJump\_Athlete* та *HighJump\_Athlete*). У базовому класі визначені загальні властивості і методи, а похідні класи розширюють їх та надають специфічні властивості та методи для кожного виду спорту. Детальну діаграму з ієрархією класів наведено в файлі *ClassDiagram1.cd*.

Структури даних:

- *List<Sportsman> sportsmen* - це список спортсменок, де кожен елемент є об'єктом класу *Sportsman*. Цей список використовується для збереження та керування інформацією про спортсменок.

Опис алгоритму розв'язку:

1. В конструкторі класу *FanGuide* створюється порожній список *sportsmen*.
2. Метод *LoadFromFile* призначений для завантаження даних про спортсменок з файлу. Він приймає шлях до файлу в якості параметра.
3. У методі *LoadFromFile* відбувається читання файлу рядок за рядком.
4. Кожен рядок розбивається на частини за допомогою роздільника “ , ”, і дані про спортсменок зберігаються у відповідних змінних.
5. За допомогою методу *CreateSportsman* створюється об'єкт спортсменок з використанням зчитаних даних.
6. Створений об'єкт додається до списку *sportsmen*.
7. Метод *CreateSportsman* перевіряє тип спорту та категорію, створює об'єкт відповідного класу-спадкоємця від *Sportsman* і повертає його.
8. Метод *DisplaySportsmenByName* виводить інформацію про спортсменок за вказаним ім'ям. Він приймає ім'я спортсменок як параметр. У циклі перевіряється кожна спортсменка у списку *sportsmen*. Якщо ім'я спортсменки містить вказану фразу, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.

9. Метод *DisplaySportsmenByAge* виводить інформацію про спортсменок за вказаним віком. Для цього методу було використано принцип перевантаження, тому розберемо обидві реалізації методу в залежності від типу переданого параметру:
- 9.1. Метод *DisplaySportsmenByAge(string ageGap)* виводить інформацію про спортсменок за заданим діапазоном віку. Він приймає параметр *ageGap* рядкового типу, який представляє діапазон віку у форматі “*minAge - maxAge*”. Опис алгоритму:
- Розбивається рядок *ageGap* на дві частини за допомогою розділювача “ - ”. Результат зберігається в масиві рядків *ages*.
  - Перевіряється, чи довжина масиву *ages* дорівнює 2. Якщо ні, виводиться повідомлення про невірний формат діапазону віку.
  - Перевіряється, чи можна перетворити елементи масиву *ages* на цілі числа. Якщо ні, виводиться повідомлення про невірний формат діапазону віку.
  - Для кожної спортсменки у списку *sportsmen* виконується наступне:
    - Обчислюється поточний вік спортсменки на основі її року народження і поточного року.
    - Перевіряється, чи поточний вік потрапляє в заданий діапазон. Якщо так, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
- 9.2. Метод *DisplaySportsmenByAge(int age)* виводить інформацію про спортсменок віком, що дорівнює переданому значенню. Він приймає параметр *age* цілочисельного типу даних, який представляє вік спортсменок, яких необхідно вивести. Опис алгоритму:
- Для кожної спортсменки у списку *sportsmen* виконується наступне:
    - Обчислюється поточний вік спортсменки на основі її року народження і поточного року.
    - Перевіряється, чи поточний вік спортсменки дорівнює заданому значенню *age*. Якщо так, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
10. Метод *DisplaySportsmenByHeight* виводить інформацію про спортсменок за заданим діапазоном зросту. Він приймає параметр *heightGap* рядкового типу, який представляє діапазон зросту у форматі “*minHeight - maxHeight*”. Опис алгоритму:

- Розбивається рядок *heightGap* на дві частини за допомогою розділювача “ - ”. Результат зберігається в масиві рядків *heights*.
  - Перевіряється, чи довжина масиву *heights* дорівнює 2. Якщо ні, виводиться повідомлення про невірний формат діапазону зросту.
  - Перевіряється, чи можна перетворити елементи масиву *heights* на цілі числа. Якщо ні, виводиться повідомлення про невірний формат діапазону зросту.
  - Для кожної спортсменки у списку *sportsmen* перевіряється, чи її зріст потрапляє в заданий діапазон. Якщо так, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
11. Метод *DisplaySportsmenByWeight* виводить інформацію про спортсменок за заданим діапазоном ваги. Він приймає параметр *weightGap* рядкового типу, який представляє діапазон ваги у форматі “*minWeight* - *maxWeight*”. Опис алгоритму:
- Розбивається рядок *weightGap* на дві частини за допомогою розділювача “ - ”. Результат зберігається в масиві рядків *weights*.
  - Перевіряється, чи довжина масиву *weights* дорівнює 2. Якщо ні, виводиться повідомлення про невірний формат діапазону ваги.
  - Перевіряється, чи можна перетворити елементи масиву *weights* на цілі числа. Якщо ні, виводиться повідомлення про невірний формат діапазону ваги.
  - Для кожної спортсменки у списку *sportsmen* перевіряється, чи її вага потрапляє в заданий діапазон. Якщо так, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
12. Метод *DisplaySportsmenByCitizenship* виводить інформацію про спортсменок з вказаним громадянством. Він приймає параметр *citizenship* рядкового типу даних, що представляє громадянство спортсменок, яких потрібно вивести. Опис алгоритму:
- Формується новий список *filteredSportsmen*, у якому містяться спортсменки, у яких громадянство (*Citizenship*) співпадає зі значенням *citizenship* (без урахування реєстру).
  - Перевіряється, чи список *filteredSportsmen* не є порожнім. Якщо так, виводиться повідомлення про відсутність спортсменів з вказаним громадянством.
  - Якщо список *filteredSportsmen* не є порожнім, для кожного спортсмена у списку *filteredSportsmen* викликається метод *DisplayDetails*, який виводить деталі спортсмена на екран.

- Лямбда-вираз використовується для фільтрації спортсменів зі списку *sportsmen* на основі їх громадянства. *FindAll* - це метод списку *sportsmen*, який приймає предикат (в даному випадку лямбда-вираз) і повертає новий список, що містить всі об'єкти з початкового списку, для яких предикат повертає *true*. У нашому випадку лямбда-вираз ( $s \Rightarrow s.Citizenship.ToLower() == citizenship.Trim().ToLower()$ ) приймає спортсмена *s* і перевіряє, чи громадянство спортсмена (представлене полем *Citizenship*) співпадає зі значенням *citizenship*, ігноруючи регістр.
13. Метод *DisplaySportsmenBySport* виводить інформацію про спортсменок за вказаним видом спорту. Він приймає вид спорту як параметр. У циклі перевіряється кожна спортсменка у списку *sportsmen*. Якщо спортсменка відповідає вказаному виду спорту, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
  14. Метод *DisplaySportsmenBySportCategory* виводить інформацію про спортсменок за вказаною категорією спорту. Він приймає категорію спортсменки як параметр. У циклі перевіряється кожна спортсменка у списку *sportsmen*. Якщо спортсменка відповідає вказаній категорії, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
  15. Метод *DisplaySportsmenByClub* виводить інформацію про спортсменок за вказаним клубом або командою. Він приймає клуб або команду як параметр. У циклі перевіряється кожна спортсменка у списку *sportsmen*. Якщо спортсменка відповідає вказаному клубу або команді, то викликається метод *DisplayDetails*, який виводить деталі спортсменки на екран.
  16. Метод *DisplayRecordHolder* виводить інформацію про рекордсменку в заданій категорії спорту. Він приймає параметр *sportCategory*, що вказує на категорію спорту, і здійснює наступні кроки:
    - Перевіряє, чи *sportCategory* відповідає “50 m swimming”, “100m swimming” або “marathon”. Якщо це так, то застосовується порядок сортування за зростанням (*OrderBy*) до списку *sportsmen*, фільтруючи тих спортсменок, у яких поле *SportCategory* співпадає з параметром *sportCategory*, що було передано в метод. Перша спортсменка з цього списку, використовуючи метод *FirstOrDefault*, визначається як рекордсменка.
    - Якщо *sportCategory* не відповідає вище переліченим категоріям спорту, то застосовується порядок сортування за спаданням (*OrderByDescending*) до списку *sportsmen*, виконуючи ті ж самі фільтрацію та вибірку першої спортсменки.

- Якщо рекордсменку знайдено (*recordHolder != null*), то викликається метод *DisplayDetails* для виведення деталей рекордсменки.
- Лямбда-вирази використовуються для зручного і компактного опису фільтрації та сортування списку спортсменів. У рядках, де використовуються лямбда-вирази, ми маємо наступну структуру:
  - *.Where(s => s.SportCategory == sportCategory.Trim().ToLower())* - лямбда-вираз використовується для фільтрації списку *sportmen*. Він перебирає кожен елемент *s* у списку і повертає *true*, якщо значення поля *SportCategory* співпадає з переданим у метод параметром *sportCategory*. Це дозволяє відфільтрувати тільки спортсменів, які належать до заданої категорії спорту.
  - *.OrderBy(s => s.PersonalRecord)* - лямбда-вираз використовується для сортування відфільтрованого списку спортсменів за полем *PersonalRecord* у порядку зростання. Тут *s* представляє кожен елемент списку, і за допомогою *s.PersonalRecord* отримується значення поля *PersonalRecord*. Це дозволяє впорядкувати спортсменів за їхнім особистим рекордом.
  - *.FirstOrDefault()* - лямбда-вираз використовується для отримання першого елемента з відсортованого списку. Він повертає першу спортсменку з відфільтрованого і відсортованого списку або *null*, якщо список порожній. Це дозволяє знайти рекордсменку в заданій категорії спорту.

Цей код дозволяє завантажити дані про спортсменок з файлу і виконувати різні операції з відображенням цих даних на екрані за різними параметрами, а також відображенням рекордсменки в заданій категорії спорту.

## 2.2. Алгоритм розв'язку задачі

Загальний алгоритм роботи програми містить наступні пункти:

1. Створення списку *List<Sportsman> sportmen* для зберігання інформації про спортсменок.
2. Завантаження даних про спортсменів з джерела (наприклад, текстового файлу в форматі CV або бази даних) і додавання їх до списку *sportmen*.

3. Передбачення можливості виконання різних операцій зі списком спортсменок, які можуть бути запитані користувачем.
4. Надання користувачу можливості вибору операції, яку вони хочуть виконати.
5. Залежно від обраної операції користувачем, виконання відповідної функціональності:
6. Для методу *DisplaySportsmenByName*: зчитування введеного користувачем імені, перебір списку *sportsmen*, перевірка збігу імен та виведення деталей спортсменок, які підходять до введеного імені.
7. Для методу *DisplaySportsmenByAge(string ageGap)*: розбиття рядка *ageGap* на мінімальний та максимальний вік, перебір списку *sportsmen*, перевірка відповідності віку спортсменок до вказаного діапазону та виведення деталей спортсменок, які підходять до вказаного діапазону віку.
8. Для методу *DisplaySportsmenByAge(int age)*: зчитування введеного користувачем віку, перебір списку *sportsmen*, перевірка збігу віку спортсменок та виведення деталей спортсменок, які мають вказаний вік.
9. Для методу *DisplaySportsmenByHeight*: розбиття рядка *heightGap* на мінімальний та максимальний зріст, перебір списку *sportsmen*, перевірка відповідності зросту спортсменок до вказаного діапазону та виведення деталей спортсменок, які підходять до вказаного діапазону зросту.
10. Для методу *DisplaySportsmenByWeight*: розбиття рядка *weightGap* на мінімальну та максимальну вагу, перебір списку *sportsmen*, перевірка відповідності ваги спортсменок до вказаного діапазону та виведення деталей спортсменок, які підходять до вказаного діапазону ваги.
11. Для методу *DisplaySportsmenByCitizenship*: зчитування введеного користувачем громадянства, фільтрування списку *sportsmen* за вказаним громадянством та виведення деталей спортсменок, які мають вказане громадянство.
12. Для методу *DisplaySportsmenBySport*: зчитування введеного користувачем спорту, перебір списку *sportsmen*, перевірка збігу спорту та виведення деталей спортсменок, які підходять до введеного спорту.
13. Для методу *DisplaySportsmenBySportCategory*: зчитування введеної користувачем категорії спорту, перебір списку *sportsmen*, перевірка збігу категорії спорту та виведення деталей спортсменок, які підходять до введеної категорії спорту.
14. Для методу *DisplayRecordHolder*: зчитування введеної користувачем категорії спорту, пошук запису про спортсменку, яка є рекордсменкою у вказаній категорії, та виведення її деталей.
15. Повторення кроків 4-5, доки користувач не вибере вихід з програми.

## РОЗДІЛ 3 ОПИС РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

### 3.1. Опис головних структур і змінних програми — 3.2. Опис головних функцій програми

#### 1. Клас *Sportsman* (спортсменка)

- Опис даних:
  - *FullName* (тип: *string*) - повне ім'я спортсменки
  - *BirthYear* (тип: *int*) - рік народження спортсменки
  - *Height* (тип: *int*) - зріст спортсменки в сантиметрах
  - *Weight* (тип: *int*) - вага спортсменки в кілограмах
  - *Citizenship* (тип: *string*) - громадянство спортсменки
  - *Sport* (тип: *string*) - вид спорту, яким займається спортсменка
  - *SportCategory* (тип: *string*) - категорія спорту, в якій виступає спортсменка
  - *Club* (тип: *string*) - назва спортивного клубу, до якого належить спортсменка
  - *PersonalRecord* (тип: *string*) - особистий рекорд спортсменки
- Конструктор:
  - *Sportsman(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - ініціалізує всі поля спортсменки
- Метод:
  - *DisplayDetails()* - виводить на екран деталі про спортсменки

#### 2. Клас *Swimmer* (плавчиня) (похідний від класу *Sportsman*)

- Конструктор:
  - *Swimmer(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
- Метод:
  - *DisplayDetails()* - перевизначений метод, виводить деталі про плавчиню

#### 3. Клас *FiftyMeter\_Swimmer* (плавчиня на 50 метрів) (похідний від класу *Swimmer*)

- Конструктор:
  - *FiftyMeter\_Swimmer(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string*

- sportCategory, string club, string personalRecord*) - викликає конструктор базового класу для ініціалізації полів
4. Клас *OneHundredMeter\_Swimmer* (плавчиня на 100 метрів) (похідний від класу *Swimmer*)
    - Конструктор:
      - *OneHundredMeter\_Swimmer(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  5. Клас *FigureSkater* (фігуристка) (похідний від класу *Sportsman*)
    - Конструктор:
      - *FigureSkater(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
    - Метод:
      - *DisplayDetails()* - перевизначений метод, виводить деталі про фігуристку
  6. Клас *CombinedTotal\_FigureSkater* (фігуристка загальної суми) (похідний від класу *FigureSkater*)
    - Конструктор:
      - *CombinedTotal\_FigureSkater(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  7. Клас *ShortProgram\_FigureSkater* (фігуристка короткої програми) (похідний від класу *FigureSkater*)
    - Конструктор:
      - *ShortProgram\_FigureSkater(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  8. Клас *FreeSkating\_FigureSkater* (фігуристка довільної програми) (похідний від класу *FigureSkater*)
    - Конструктор:
      - *FreeSkating\_FigureSkater(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* -



викликає конструктор базового класу для ініціалізації полів

9. Клас *Athlete* (легкоатлетистка) (похідний від класу *Sportsman*)
  - Конструктор:
    - *Athlete(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  - Метод:
    - *DisplayDetails()* - перевизначений метод, виводить деталі про легкоатлетистку
10. Клас *Decathlete* (легкоатлетистка, що займається десятиборством) (похідний від класу *Athlete*)
  - Конструктор:
    - *Decathlete(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
11. Клас *ShotPut\_Athlete* (легкоатлетистка, що займається штовханням ядра) (похідний від класу *Athlete*)
  - Конструктор:
    - *ShotPut\_Athlete(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  - Метод:
    - *DisplayDetails()* - перевизначений метод, виводить деталі про легкоатлетистку, що займається штовханням ядра
12. Клас *PoleVault\_Athlete* (легкоатлетистка, що займається стрибками з жердиною) (похідний від класу *Athlete*)
  - Конструктор:
    - *PoleVault\_Athlete(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)* - викликає конструктор базового класу для ініціалізації полів
  - Метод:
    - *DisplayDetails()* - перевизначений метод, виводить деталі про легкоатлетистку, що займається стрибками з жердиною
13. Клас *MarathonRunner* (легкоатлетистка, що займається марафонами) (похідний від класу *Athlete*)

- Конструктор:
    - *MarathonRunner* (*string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord*) - викликає конструктор базового класу для ініціалізації полів
  - Метод:
    - *DisplayDetails()* - перевизначений метод, виводить деталі про легкоатлетистку, що займається марафонами
14. Клас *Jump\_Athlete* (легкоатлетистка, що займається стрибками) (похідний від класу *Athlete*)
- Конструктор:
    - *Jump\_Athlete* (*string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord*) - викликає конструктор базового класу для ініціалізації полів
  - Метод:
    - *DisplayDetails()* - перевизначений метод, виводить деталі про легкоатлетистку, що займається стрибками
15. Клас *LongJump\_Athlete* (легкоатлетистка, що займається стрибками у довжину) (похідний від класу *Jump\_Athlete*)
- Конструктор:
    - *LongJump\_Athlete* (*string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord*) - викликає конструктор базового класу для ініціалізації полів
16. Клас *HighJump\_Athlete* (легкоатлетистка, що займається стрибками у висоту) (похідний від класу *Jump\_Athlete*)
- Конструктор:
    - *HighJump\_Athlete* (*string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord*) - викликає конструктор базового класу для ініціалізації полів
17. Клас *FanGuide* (представляє основну програму для керування спортивною інформацією та надає користувацький інтерфейс для взаємодії зі списком спортсменів)
- Змінна:
    - *List<Sportsman> sportsmen*: Список спортсменок, які будуть зберігатись та керуватись програмою
  - Конструктор:

- *FanGuide()*: Конструктор класу *FanGuide*, ініціалізує порожній список спортсменок
- Методи:
  - *LoadFromFile(string filePath)*: Завантажує дані про спортсменів з текстового файлу формату CV. Параметр *filePath* вказує шлях до файлу, з якого треба завантажити дані. Метод зчитує дані з файлу, розбирає їх та створює об'єкти спортсменок за заданими параметрами, після чого додає їх до списку *sportsmen*. У випадку помилки завантаження або обробки даних виводиться відповідне повідомлення.
  - *CreateSportsman(string fullName, int birthYear, int height, int weight, string citizenship, string sport, string sportCategory, string club, string personalRecord)*: Приватний метод, який створює об'єкт спортсменки з заданими параметрами. Метод перевіряє тип спорту та категорію спорту та створює відповідний об'єкт на основі цих даних. Повертає створений об'єкт спортсменки.
  - *DisplaySportsmenByName(string fullName)*: Відображає інформацію про спортсменок за заданим ім'ям або його частиною. Параметр *fullName* вказує повне ім'я або частину імені спортсмена. Метод виводить деталі про спортсменів, імена яких містять задану частину імені.
  - *DisplaySportsmenByAge(string ageGap)*: Відображає інформацію про спортсменок у заданому діапазоні віку. Параметр *ageGap* приймає діапазон віку у форматі "*minAge - maxAge*". Метод розбиває переданий рядок на два числа, використовуючи роздільник "-" і перевіряє правильність формату. Потім він порівнює вік кожної спортсменки з заданим діапазоном та виводить інформацію про спортсменок, які потрапляють у цей діапазон
  - *DisplaySportsmenByAge(int age)*: Відображає інформацію про спортсменок певного віку. Параметр *age* вказує необхідний вік спортсменок. Метод порівнює вік кожної спортсменки з вказаним значенням та виводить деталі про спортсменок, чий вік відповідає переданому значенню
  - *DisplaySportsmenByHeight(string heightGap)*: Відображає інформацію про спортсменок у заданому діапазоні зросту. Параметр *heightGap* приймає діапазон зросту у форматі "*minHeight - maxHeight*". Метод розбиває

переданий рядок на два числа, використовуючи роздільник “ - ” і перевіряє правильність формату. Потім він порівнює зріст кожної спортсменки з заданим діапазоном та виводить інформацію про спортсменок, чий зріст потрапляє у цей діапазон

- *DisplaySportsmenByWeight(string weightGap)*: Відображає інформацію про спортсменок у заданому діапазоні ваги. Параметр *weightGap* приймає діапазон ваги у форматі “*minWeight - maxWeight*”. Метод розбиває переданий рядок на два числа, використовуючи роздільник “ - ” і перевіряє правильність формату. Потім він порівнює вагу кожної спортсменки з заданим діапазоном та виводить інформацію про спортсменок, чия вага потрапляє у цей діапазон
- *DisplaySportsmenByCitizenship(string citizenship)*: відображає спортсменок за громадянством. Вхідним параметром є рядок *citizenship*, який представляє громадянство спортсменок, що потрібно відобразити. Метод шукає спортсменів у колекції *sportsmen*, у яких значення властивості *Citizenship* співпадає з заданим громадянством. Всі відповідні спортсмени зберігаються в списку *filteredSportsmen*. Якщо список *filteredSportsmen* не містить жодну спортсменку, виводиться повідомлення про те, що не знайдено спортсменок з вказаною національністю. У протилежному випадку виводиться заголовок з шуканим громадянством, а потім виводяться деталі кожного спортсмена.
- *DisplaySportsmenBySport(string sport)*: Відображає інформацію про спортсменок за заданим видом спорту. Параметр *sport* вказує вид спорту. Метод виводить деталі про спортсменів, які займаються заданим видом спорту.
- *DisplaySportsmenBySportCategory(string sportCategory)*: Відображає інформацію про спортсменок за заданою категорією спорту. Параметр *sportCategory* вказує категорію спорту. Метод виводить деталі про спортсменів, які займаються заданим видом спорту.
- *DisplaySportsmenByClub(string club)*: Відображає інформацію про спортсменок за заданим спортивним клубом або командою. Параметр *club* вказує назву спортивного клубу або команди. Метод виводить деталі про спортсменок, які представляють заданий клуб або команду.

- *DisplayRecordHolder(string sportCategory)*: Відображає інформацію про рекордсменку у заданій категорії спорту. Параметр *sportCategory* вказує категорію спорту. В залежності від категорії, метод знаходить спортсменку з найкращим особистим рекордом у цій категорії (враховуючи чи найнижче значення є кращим або найвище), і виводить інформацію про цю спортсменку. У випадку, якщо не знайдено рекордсменку в заданій категорії, виводиться відповідне повідомлення.
  - *DisplayAllSportsmen()*: Відображає інформацію про всіх спортсменок. Метод виводить деталі про всіх спортсменок, які знаходяться у списку.
18. Клас *Program* (відповідає за взаємодію з користувачем та виконання відповідних методів для відображення даних про спортсменів залежно від обраних опцій)
- Метод:
    - *Main*: Представляє точку входу в програму. В методі ньому створюється об'єкт класу *FanGuide* з назвою *fanGuide*. Потім викликається метод *LoadFromFile*, який завантажує дані про спортсменів з файлу. Після завантаження даних виводиться привітальне повідомлення та список доступних опцій для користувача. Використовуючи цикл *while*, програма очікує введення користувача та виконує відповідні дії в залежності від обраної опції. Користувач вводить номер вибраної опції, а потім програма перевіряє цей вибір за допомогою оператора *switch*. Залежно від введеного номера, програма викликає відповідний метод об'єкта *fanGuide* для відображення даних про спортсменів. Після виходу з циклу виводиться повідомлення про завершення використання програми.

### 3.3. Опис інтерфейсу

Після завантаження даних (див. попередній пункт, клас *Program*, метод *Main*) виводиться привітальне повідомлення та список доступних опцій для користувача:

```
C:\Users\Mapia\KPI1 kypcOI x + v
Welcome to the Sportsmen Fan Guide!
Please select an option:
1. Search sportsmen's data by sportsmen name
2. Search sportsmen's data by sportsmen age
3. Search sportsmen's data by sportsmen height
4. Search sportsmen's data by sportsmen weight
5. Search sportsmen's data by sportsmen citizenship
6. Search sportsmen's data by sportsmen sport
7. Search sportsmen's data by sportsmen sport category
8. Search sportsmen's data by sportsmen club
9. Search record holder in specific category
10. Exit

Enter the number of your choice: |
```

Щоб скористатися будь-якою опцією, користувачу необхідно ввести відповідний номер та натиснути клавішу Enter. Користувачу будуть запропоновані додаткові питання або інструкції, які потрібно виконати, щоб отримати потрібні дані.

Наприклад, якщо користувач оберете опцію 1, йому/їй буде запропоновано ввести ім'я спортсмена, за яким потрібно здійснити пошук. Після введення програма відобразить дані про спортсменів з відповідним іменем:

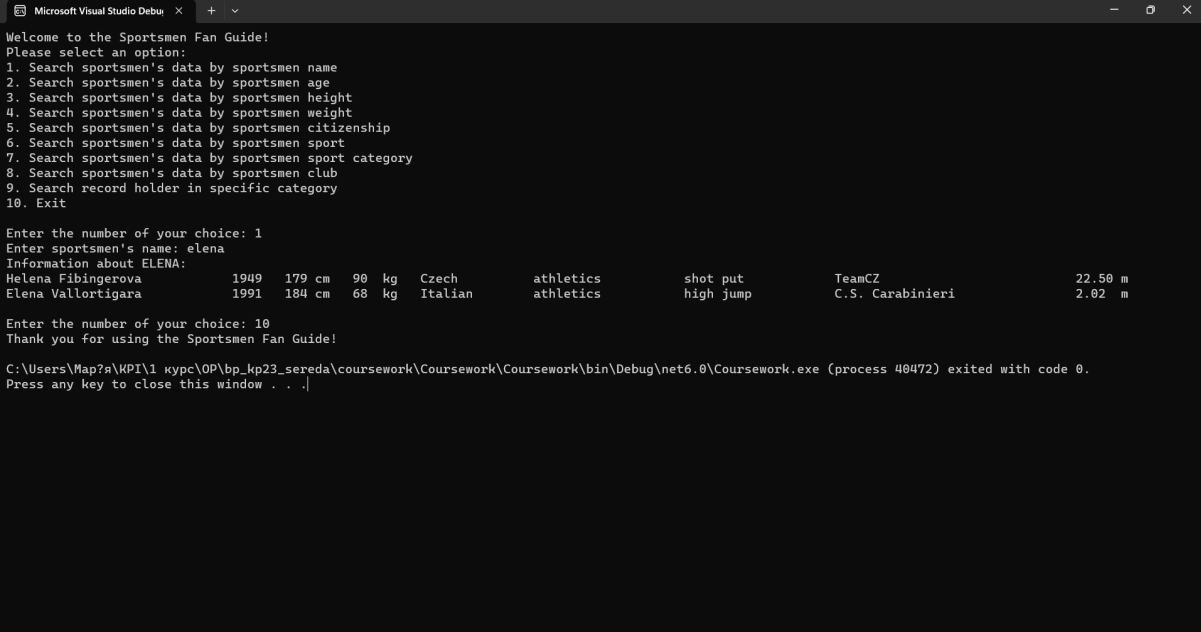
```
C:\Users\Mapia\KPI1 kypcOI x + v
Welcome to the Sportsmen Fan Guide!
Please select an option:
1. Search sportsmen's data by sportsmen name
2. Search sportsmen's data by sportsmen age
3. Search sportsmen's data by sportsmen height
4. Search sportsmen's data by sportsmen weight
5. Search sportsmen's data by sportsmen citizenship
6. Search sportsmen's data by sportsmen sport
7. Search sportsmen's data by sportsmen sport category
8. Search sportsmen's data by sportsmen club
9. Search record holder in specific category
10. Exit

Enter the number of your choice: 1
Enter sportsmen's name: elena
Information about ELENA:
Helena Fibingerova      1949   179 cm   90 kg   Czech   athletics   shot put   TeamCZ   22.50 m
Elena Vallortigara      1991   184 cm   68 kg   Italian athletics   high jump   C.S. Carabinieri   2.02 m

Enter the number of your choice: |
```

Користувач може продовжити вводити відповідні номери опцій та виконувати інструкції, доки не завершить використання програми.

Якщо користувач вибирає опцію “10”, програма завершує роботу. Після завершення використання програми буде виведено повідомлення “Thank you for using the Sportsmen Fan Guide!”.



```
Microsoft Visual Studio Debu x + v
Welcome to the Sportsmen Fan Guide!
Please select an option:
1. Search sportsmen's data by sportsmen name
2. Search sportsmen's data by sportsmen age
3. Search sportsmen's data by sportsmen height
4. Search sportsmen's data by sportsmen weight
5. Search sportsmen's data by sportsmen citizenship
6. Search sportsmen's data by sportsmen sport
7. Search sportsmen's data by sportsmen sport category
8. Search sportsmen's data by sportsmen club
9. Search record holder in specific category
10. Exit

Enter the number of your choice: 1
Enter sportsmen's name: elena
Information about ELENA:
Helena Fibingerova      1949  179 cm  90 kg  Czech      athletics      shot put      TeamCZ      22.50 m
Elena Vallortigara      1991  184 cm  68 kg  Italian    athletics      high jump     C.S. Carabinieri  2.02 m

Enter the number of your choice: 10
Thank you for using the Sportsmen Fan Guide!

C:\Users\Map?n\KPI\1 курс\OD\bp_kp23_sereda\coursework\Coursework\bin\Debug\net6.0\Coursework.exe (process 40472) exited with code 0.
Press any key to close this window . . .
```

Цей інтерфейс дозволяє користувачу швидко та зручно знаходити та переглядати дані про спортсменок відповідно до його/її вибору.

## ВИСНОВКИ

У процесі розробки програмного продукту “Довідник фаната” вдалось успішно реалізувати систему, яка надає користувачам зручний спосіб отримання інформації про спортсменок за допомогою консольного інтерфейсу. Деякі здібності, які вдалось реалізувати, включають:

- Пошук спортсменів за різними критеріями: ім'я, вік, зріст, вага, громадянство, вид спорту, категорія спорту та клуб.
- Пошук рекордсменки у певній категорії.
- Завантаження даних про спортсменок з файлу.

Переваги розробленого програмного продукту включають простий та зрозумілий інтерфейс для користувача, який дозволяє швидко знаходити потрібну інформацію. Користувачі можуть використовувати різні критерії пошуку для отримання точних результатів. Крім того, можливість завантаження даних з файлу робить програму гнучкою та легко змінюваною.

Проте, розроблений програмний продукт має кілька недоліків:

- Відсутність перевірки коректності введених даних виключно залежить від коректності введення користувача. Недостатньо стійка обробка помилок може призвести до некоректних результатів або аварійного завершення програми.
- Обмежена функціональність: розроблений продукт враховує лише деякі критерії пошуку та не надає можливість додавання або видалення даних про спортсменів під час роботи програми.

Можливі шляхи подальшого покращення програмного продукту:

- Впровадження механізмів перевірки коректності введених даних для запобігання помилкам та некоректним результатам.
- Розширення функціональності шляхом додавання нових критеріїв пошуку.
- Вдосконалення інтерфейсу, забезпечення більш зручного та інтуїтивно зрозумілого способу взаємодії з користувачем.
- Оптимізація алгоритмів пошуку та обробки даних для забезпечення швидкості та ефективності роботи програми.

Ці поліпшення можуть сприяти поліпшенню користувацького досвіду, розширенню функціональності та забезпеченню більш надійної та стійкої роботи програмного продукту “Довідник фаната”.



## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Івченко І. Ю. Математичне програмування : навч. посіб. для студентів вищ. навч. закл. / І. Ю. Івченко. – Київ : Центр учб. літ., 2007. – 232 с.
2. Казимир В. В. Об'єктно-орієнтоване програмування : навч. посіб. для студентів вищ. навч. закл. / В. В. Казимир. – Київ : Слово, 2008. – 192 с.
3. Ковалюк Т. В. Основи програмування : підруч. для студентів ВНЗ / Т. В. Ковалюк. – Київ : ВНУ, 2005. – 384 с.
4. C# docs - get started, tutorials, reference. [Електронний ресурс] // Microsoft Learn: Build skills that open doors in your career. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/csharp> (дата звернення: 03.06.2023). – Назва з екрана.