КУРСОВА РОБОТА

КР.ІП – 00.00.00.000 ПЗ

Група ІП-20-3

Гуйдаш Владислав

2021

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти та газу
Інститут інформаційних технологій

Курсовий проект

з дисципліни «Об'єктно-орієнтоване програмування»

на тему: Розробка прикладної програми «Моделювання та аналіз АІС спортивних організацій міста засобами С++»

Студента(ки) 1 **курсу групи** ІП-20-3
напряму підготовки 121
інженерія програмного забезпечення
Гуйдаш В.А.
(прізвище та ініціали)
Керівник зав. кафедри ІПЗ,проф.д.т.н.
В.І. Шекета
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: _____ **Оцінка: ECTS** _____

м. Івано-Франківськ — 2021рік

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти та газу
Інститут інформаційних технологій

ЗАТВЕРДЖУЮ

зав. кафедри ІПЗ, проф., д.т.н.

_____В.І. Шекета

«_____» _____ 2021р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

На курсовий проєкт з дисципліни «ООП» студенту Гуйдашу В.А. групи ІП-20-3.

ТЕМА: Розробка прикладної програми з проєктування інформаційних систем згідно завдання.

Постановка задачі.

1. Розробити прикладну програму
2. Розробити інтерфейс користувача
3. Розробити та використати активні елементи
4. Розробити та використати шаблони не менше 5 шт.
5. Розрахувати швидкодію роботи з інтерфейсом користувача
6. Адаптувати розроблене прикладне програмне забезпечення до різних роздільних здатностей цифрових пристроїв.

Дата видачі « 25» лютого 2021р. Керівник _____

Завдання отримав Гуйдаш Владислав Андрійович

Міністерство освіти і науки України
Івано-Франківський національний технічний університет нафти та газу
Інститут інформаційних технологій

ЗАТВЕРДЖУЮ

зав. кафедри ІПЗ, проф., д.т.н.

_____В.І. Шекета

«_____» _____ 2021р.

ТЕХНІЧНЕ ЗАВДАННЯ

На розробку прикладного програмного забезпечення з моделювання та аналізу програмного забезпечення»

1. Область застосування — проєктування інформаційних систем .

2. Основа розробки — робочий навчальний план дисципліни.

3. Мета та експлуатаційне призначення:

a. мета – отримання практичних навичок проєктування та конфігурування інформаційних систем ;

b. призначення розробки — навчальний курсовий проект із дисципліни «Проєктування інформаційних систем»;

4. Джерела розробки — індивідуальне завдання на курсовий проєкт із дисципліни, технічні рекомендації щодо проектування інформаційних систем та інші технічні матеріали для налаштування окремих компонентів програмної системи.

5. Технічні вимоги

Кінцевий термін виконання курсового проєкту «червень» 2021 р

Початок розробки «квітень» 2021 р

Порядок контролю та прийняття.

6.1. Виконання етапів технічної та розрахункової документації курсового проєкту, а також моделювання роботи інформаційної системи контролюється викладачем згідно з графіком виконання проєкту;

6.2. Прийняття проекту здійснюється комісією, затвердженою зав. кафедри згідно графіку захисту.

Розробив студент групи ІП-20-3 Гуйдаш Владислав Андрійович

ЗМІСТ

					КР.ІП-00.00.00.000 ПЗ		
Змн	Лист	№ докум.	Підпис	Дата			
Розроб.		Гуйдаш В.А.			Розробка програмного рішення для консольної бази даних по предметній області « Моделювання та аналіз АІС спортивних організацій міста» засобами С++	Літ.	Арк. 1 / Аркушів
Перевір.		Шекета В. І.					
Реценз.						ІФТУНГ, ІП-20-3	
Н. Контр.							
Затверд.							

ВСТУП

В основі об'єктно-орієнтованої мови програмування лежать два основних поняття: клас та об'єкт. Об'єкт – це базове поняття в ООП, це конкретна реалізація, екземпляр класу. Об'єкти однаковими наборами змінних стану і методів, утворюють клас. Якщо об'єкти мають реалізацію з конкретного світу, то класи є абстракціями. Трохи більш складні об'єкти можуть взагалі не містити даних, а представляти процес і містити тільки функції, які реалізують цей процес. Для формування реального об'єкта необхідно мати шаблон, по прикладу якого і будується даний об'єкт.

Бази даних завжди були найважливішою темою при вивченні інформаційних систем. Однак в останні роки сплеск популярності Інтернету і бурхливий розвиток нових технологій для Інтернету зробили знання технології баз даних для багатьох одним з найактуальніших шляхів кар'єри.

Мета мого курсового проекту: розробити власну базу даних із консольним інтерфейсом, використовуючи класи та об’єкти класу за заданим варіантом з наданням їм відповідних прав. Процес підготовки і рішення задачі складався з кількох етапів:

- постановка задачі;

- бізнес-аналітика вибраної теми;

- побудова моделі класів та їх прототипів;

- розробка алгоритму;

- написання та налагодження програми на мові програмування C++;

- тестування програми.

Об’єктом дослідження є розробка системи управління базою даних консольним інтерфейсом на прикладі спортивної інфраструктури міста Івано-Франківська, моделювання та аналіз спортивних організацій міста.

Предметом роботи є програмна реалізація мовою С++ об’єктно-класової моделі.

Теоретико-методологічною основою розробки є парадигма ООП.

Робота є рукописом об’ємом 77 сторінок машинописного тексту, список використаних джерел на 7 позицій, містить додатки, в т. ч код програми та результати виконання контрольного прикладу.

1. 1 Опис сучасного стану динамічних та статичних властивостей предметної області

Ключовим поняттям в об’єктно-орієнтованому програмуванні на C++ є поняття класу. Загалом, певному класу притаманні ті чи інші властивості. Вони представляють собою ту чи іншу змінну, однак при різних умовах, на відмінну від змінної, властивість може повертати різні значення.

Класи - це розширене поняття структур даних : як і структури даних, вони можуть містити члени даних, але вони також можуть містити функції як члени. Об'єкт є конкретизацією класу. З точки зору змінних, клас буде типом, а об'єкт - змінною. Фактично – це визначений програмістом нестандартний тип даних, тому поняття полів і методів класу повністю співпадають з аналогічними поняттями абстрактного типу даних.

Для аналізу предметної області вибрано узагальнення спортивних організацій міста Івано-Франківська, тобто такі організації, яким притаманні всі основні ознаки і атрибути інфраструктури міста. Дана організація займається обробкою спортивних споруд міста, організацією змагань. Організація спорту представлена спортивними клубами для певного виду спорту, спортсменами і їхніми тренерами та спонсорами змагань. Споруди та змагання, які на них проводяться, знаходяться виключно в Івано-Франківській області.

Основними спортивними спорудами є:

- ✓ Стадіони,
- ✓ Спортивні зали,
- ✓ Манежі,
- ✓ Корти.

Кожна категорія споруд має атрибути, специфічні тільки для неї.

Спортсмен тренується у певній споруді, призначена для певного виду спорту. За одним спортсменом може бути закріплено декілька тренерів, залежності від кількості вибраних видів спорту.

Спортсмен може одночасно займатися декількома видами спорту, а саме:

- Бокс,
- Футбол,
- Баскетбол,
- Волейбол,
- Теніс,
- Легка атлетика,
- Настільний теніс.

Декілька разів на рік організовуються змагання з певного виду спорту, де виступають спортсмени і за результатами участі проводиться нагородження.

Спортсмени під керівництвом тренерів займаються окремими видами спорту, при цьому один і той же спортсмен може займатися кількома видами спорту, і в рамках одного і того ж виду спорту може тренуватися у кількох тренерів. Всі спортсмени об'єднуються в спортивні клуби, при цьому кожен з них може виступати тільки за один клуб.

Організації постійно змінюються, зростають, адаптовуються. Змінюються також і бізнес-процеси. Відповідно з’являється необхідність створити якісну систему обліку та управління, яка дає змогу контролювати дані про кожну споруду, організацю та змагань. Основним призначенням програмного забезпечення є автоматизація ведення обліку. За кожною спорудою закріплені змагання та тренера, які проводяться в певні дати. Тому ведеться облік кількості проведених змагань за певний період, які спортсмени приймали участь з певного виду спорту та під керівництвом якого тренера. Також береться до уваги спонсори, які фінансують змагання або той чи інший клуб.

Спеціалізоване програмне забезпечення локального масштабу повинне автоматизувати та контролювати бізнес-процеси, зберігати інформацію про спортсменів, тренерів та об’єкти спортивної індустрії. Дане програмне забезпечення сприятиме подальшому розвитку та масштабуванню організацій міста, а також збільшенню прибутку. Це – комплексне рішення для спорт індустрії , яке охоплює оперативний облік в управлінні молоді та спорту Івано-Франківська.

Наступні переваги автоматизації:

- ❖ контроль за ефективністю проведення змагань;
- ❖ оцінка якості роботи споруд;
- ❖ Ведення обліку діючих інфраструктур;
- ❖ оптимізація кількості спортсменів;
- ❖ аналіз ефективності роботи спонсорського комітету;
- ❖ контроль за викладацької роботи тренерів;
- ❖ покращення якості тренування спортсменів;
- ❖ скорочення витрат;
- ❖ можливість проводити складні розрахунки автоматично;
- ❖ локалізація всієї інформації про діяльність організацій в одному місці та можливість використовувати її для прийняття поточних і стратегічних рішень.

Автоматизація особлива необхідна для автопідприємства за

наявності таких факторів:

- вся інформація знаходиться в окремих файлах;

- існує залежність виконання поставленого плану на рік;
- постійне дублювання одних і тих самих дій;

Отже, спортивні організації потребують якісного програмного забезпечення, для того щоб автоматизувати та покращити якість праці. Ці продукти вже в базовій версії повинні покривати всі бізнес-процеси , а також гнучко підлаштовуватися під певні умови.

Також важливі такі характеристики як

- ❖ зручний інтерфейс,
- ❖ можливість масштабування,
- ❖ подальша підтримка та розвиток продукту,
- ❖ вартість володіння продуктом,
- ❖ висока продуктивність,
- ❖ можливість інтеграції з іншими інформаційними системами.

1.2 Опис об’єктів в предметній області згідно прийнятої стилістики

Реалізація баз даних ґрунтується на виділенні об'єктів предметної області та структуризації документів, де ці об'єкти, ґрунтуючись на логічному аналізі атрибутів, можна виділити.

Предметна область (бази даних) — це сфера пов’язаних між собою функцій, за допомогою яких досягається виконання поставлених цілей і застосування конкретної бази даних. Кожна предметна область має фрагменти, а самі фрагменти містять в собі процеси і об’єкти, що використовують об'єкти, а також безліччю користувачів, що характеризуються різними поглядами на предметну область.

Проектування баз даних включає кілька етапів:

- Проектування инфологічної концептуальної моделі БД.
- Проектування даталогічної моделі БД.
- Проектування фізичної моделі БД.
- Реалізація БД.

Кожен етап має на увазі послідовне проектування бази даних і включає ряд особливостей.

Інфологічна модель відображає реальний світ у деякі зрозумілі людині концепції, повністю незалежні від параметрів середовища зберігання даних. Це узагальнене неформальний опис створюваної бази даних.

Даталогічна модель відображає логічні зв'язки між елементами даних, незалежно від їх змісту і середовища зберігання. Даталогічне моделювання передбачає опис, створюване за інфологічної моделі даних.

Фізична модель даних - модель, яка визначає розміщення даних на зовнішніх носіях, методи доступу і техніку індексування. Вона так само називається внутрішньою моделлю системи, яка визначає і оперує розміщенням даних і їх взаємозв'язками на запам'ятовуючих пристроях.

Згідно вибраного варіанту, для аналізу моєї предметної області вибрано спортивні споруди міста Івано-Франківська. Спортивна інфраструктура представлена спортивними спорудами різного типу: спортивні зали, манежі, стадіони, корти. Кожна споруда має специфічні для неї атрибути: стадіон характеризується місткістю, корт- покриттям.

Спортсмени, які займаються в спортивній споруді певного виду спорту відповідно. Кожен спортсмен займається під керівництвом тренера і може вибрати кілька видів спорту і звичайно ж тренерів. Всі спортсмени об’єднюються в спортивні клуби, але кожен з них може виступати тільки за один клуб.

Для реалізації поставленої задачі потрібно ввести систему класів. Клас можна реалізувати 2 способами:

1) створити окремі класи для кожного виду запиту і використовувати наслідування класів;

2) створити один клас, який буде зберігати інформацію про різні види об’єктів.

У даному курсовому проекті я другий спосіб, оскільки це дає можливість ефективно побудувати складні ієрархічні класи з можливістю їх зручної модифікації. Роботу класів в ієрархії можна змінювати шляхом додавання нових успадкованих класів у потрібному місці ієрархії.

Новий клас можна отримати від декількох базових класів. Такий процес називається множинним наслідуванням. Наслідування – це спосіб реалізації нових класів, основою якого є основний клас. Клас, з якого унаслідується

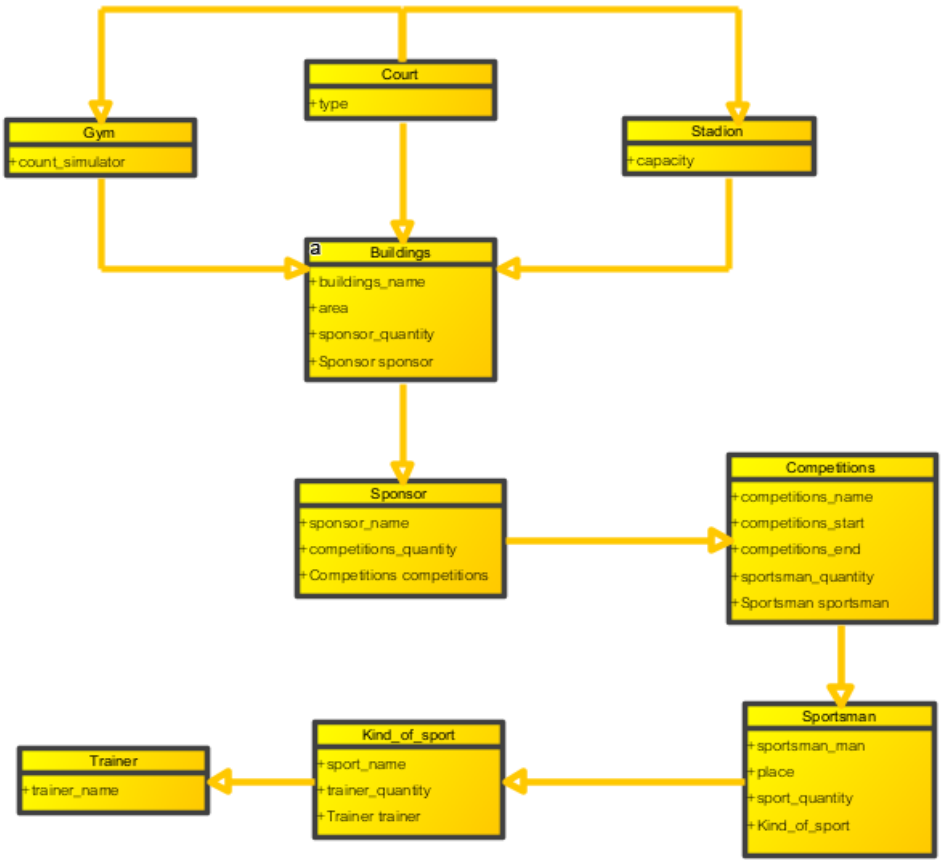
новий клас успадковує функціонал та параметри «батьківського» класу. При успішній реалізації похідного класу, він може бути доповненим власними властивостями та методами.

Одним з таких є клас **Building,** який містить поля даних для зберігання назви споруди, площі споруди та підтримки спонсора. Від нього наслідуються класи **Stadion**, який окрім об’єктів класу **Stadion** містить атрибут «місткість», клас **Gym** містить атрибут «Кількість тренажерів», а клас корт отримує атрибут «покриття». У загальному класі інфраструктур створив вказівник на клас **Sponsor,** який дає доступ до об’єктів даного класу. Кожен похідний клас може мати властивості та методи, які притаманні лише йому.

Також створив класи:

- Trainer (містить ім’я тренера);
- Kindsport (містить назву спорту і вказівник на клас Trainer і кількість тренерів даного виду);
- Sportsman (містить ім’я спортсмена, розряд, кількість вибраних дисциплін спорту та тренерів відповідно);
- Competitions (містить назву змагання, в якому році почалось і закінчилось і кількість спортсменів, які брали участь з вказівником на клас Sportsman);
- Sponsor (містить ім’я спонсора, кількість змагань та вказівник на клас Competition).

Таблиця 1.1. Структура класів



C++ дозволяє інкапсулювати дані. Інкапсуляція (або «приховування інформації») – це процес прихованого зберігання деталей реалізації об'єкта. Користувачі звертаються до об'єкта через відкритий інтерфейс. Абстракція даних – це механізм викриття тільки інтерфейсів і приховування деталей реалізації від користувача. інкапсуляція реалізована через специфікатор

доступу. Як правило, всі змінні-члени класу є закритими, а більшість методів є відкритими (з відкритим інтерфейсом для користувача).

Інкапсуляція має багато переваг:

1. інкапсульовані класи простіші у використанні і зменшують складність програм;
2. інкапсульовані класи допомагають захистити дані і запобігають їх неправильному використанню;
3. інкапсульовані класи легше змінити;
4. інкапсульовані класи легше налагодити.

Інкапсуляція здійснюється специфікаторами доступу. Специфікатор доступу визначає, хто має доступ до членів цього специфікатору. Кожен з членів має свій рівень доступу відповідно до специфікатора доступу (або, якщо він не вказаний, відповідно до специфікатора доступу за замовчуванням).

Класи мають той самий формат, що і звичайні структури даних , за винятком того, що вони також можуть включати функції та мати ці нові речі, які називаються специфікаторами доступу . Специфікатор доступу є один з наступних трьох ключових слів: private, public або protected. Ці специфікатори змінюють права доступу для членів, які слідують за ними:

- У розділах із заголовками **public** розміщуються загальнодоступні поля та методи, які використовуються для інтерфейсу об’єктів даного класу з програмою.
- У розділах із заголовками **private** розміщуються закриті дані, доступ до яких може бути здійснений лише за допомогою методів самого класу або за допомогою друзів класу.

- У розділах із заголовками **protected** учасники доступні від інших членів того ж класу (або від їх *"друзів"*), а також від членів їх похідних класів.

Якщо зробити члени класу Buildings публічними, то до них можна буде безпосередньо звертатися з функції main(). Рекомендується встановлювати специфікатор доступу private змінним-членам класу і специфікатор доступу public – методам класу (якщо немає вагомих підстав робити інакше). Оскільки для правильного функціонування програми необхідно передати поля дочірнім класам, то для цих полів, варто встановити специфікатор protected. Специфікатор доступу protected відкриває доступ до членів класу дружнім і дочірнім класами. Доступ до protected-члену поза тілом класу закритий.

Клас Building – абстрактний клас, тобто він не містить членів змінних, які повинні належати лише цьому класу. Отже, усі поля класу Building захищені (protected), тобто доступні лише методам цього класу, та класам, які походять від Vehicle. Специфікатор protected у цьому випадку дуже корисний, оскільки кількість класів, що походять від класу Building є відносно невеликою, тобто якщо доведеться внести зміни в реалізацію батьківського класу і відповідно похідних класів, то це не займе багато часу. На думку автора, варто спочатку перерахувати protected-члени класу Building, а потім його public-члени, оскільки public-члени використовуватимуть protected-члени.

В головному файлі DataBase_sport_IF**.cpp** розробив інтерфейс головного меню, запрограмував клавіші для відкриття того чи іншого пункту, передбачив некоректний ввід клавіш. За допомогою наступних прототипів та методів запрограмував додавання, сортування, зміну та збереження даних.

Прототип методів записав в файлі **Functions.cpp**.

Використовуються методи, розміщені в заголовковому файлі

Functions.h. Зокрема:

- функції виводу помилки на екран

- функції виводу меню

- функції зчитування даних із файла

- допоміжні функції сортування.

Оголошення функцій можна помістити в заголовкові файли, щоб потім мати можливість використовувати ці функції в декількох файлах або навіть в декількох проектах. Класи в цьому плані нічим не відрізняються від функцій. Визначення класів можуть бути поміщені в заголовки для полегшення їх повторного використання в декількох файлах або проектах. Зазвичай, визначення класу поміщається в заголовки з тим же ім'ям, що у класу, а методи, визначені поза тілом класу, поміщаються в файл .cpp з тим же ім'ям, що у класу. В даній роботі оголосив класи в заголовковому файлі Data.h , а саму реалізацію класів з методами в файлі Data.cpp.

Отже, робота програми забезпечується саме такими структурами даних.

2. ОСНОВНА ТЕХНІЧНА ЧАСТИНА

2.1. Розробка моделі бази даних.

Меню консольної бази даних C++ передбачає такі основні алгоритми:

1. Додавання даних в базу.

Генерування коду відбувається під час додавання даних про організацію, перевіркою наявності файлу Sport.txt , при його наявності, відбувається його читання. Перед початком роботи з базою користувач може сам вибрати спосіб додавання даних з файлу, або вручну. Дійшовши до кінця файлу, код генерується таким чином: останній наявний код + 1. Якщо файл пустий і не містить жодної інформації, програма виводить на екран повідомлення «Немає даних». Тоді користувач повинен вибрати метод «Ввести дані вручну», ввівши відповідну клавішу з клавіатури.

Метод додавання даних про спортсмена чи спортивний комплекс (DataEntry( )) вміщує у собі функцію генерації ідентифікаційного коду, при записі даних у файл.

З клавіатури вводиться назви інфраструктур, їхнє призначення, місткість, тип покриття, кількість тренажерів (відповідно до категорії споруди та їхніх атрибутів). Також потрібно ввести дані про спортсмена, у якого тренера і у скількох вони тренуються, їхні види спорту та кількість. Виходячи з цього, проводяться спортивні змагання. Вводиться назва змагання, проміжок часу проведення, які спортсмени зайняли місце і яке. Організацією змагань займаються спонсори, тому потрібно ввести їхні імена. Ці дані виводяться в консоль для їх огляду і підтвердження правильності вводу, після чого записуються у файл. Для реалізації завдання з врахуванням

результатів аналізу предметної області було створено об’єктно-класову модель.

2. Видалення записів з бази даних.

Видалення реалізовано лише для стрічки даних з файлу. Розглянемо цей алгоритм на прикладі класу «Sportsman». Метод видалення даних (DeleteData()): виконується пошук спортсмена за попередньо-введеним ідентифікаційним кодом, якщо його знайдено, рядок із усією інформацією про нього пропускається, при продовженні, наступні ідентифікаційні коди спортсмена зменшуються на 1, якщо не знайдено – то просто переписує рядок у проміжну змінну. Після завершення видалення змінені дані перезаписуються у файл Sport.txt.

3. Зміна окремих даних.

Метод редагування даних: з клавіатури вводиться кількість даних, які потрібно змінити. Відкривається файл «Sport.txt» і по одному першому слову (коду) кожного рядку зчитуються дані, при знаходженні потрібного коду, виводиться і система запитує, що саме потрібно змінити, при підтвердженні нововведених змін, із початку файлу по одному першому слову (коду) кожного рядку знову зчитуються дані, якщо код не збігається з введеним – рядок записується в проміжну змінну, якщо збігається – заміняє необхідну інформацію і записується в змінну. Після завершення редагування змінені дані перезаписуються у файл.

4. Сортування даних за алфавітним порядком.

Сортування даних відбувається методом “ бульбашки ”. Сортування даних відбувається за прізвищем в алфавітному порядку. При цьому має виконуватись умова, якщо ці елементи є елементами класа, то якщо прізвище першого більше ніж прізвище наступного, то дані, які зберігаються в поточному, присвоїти тимчасовому елементу.

5. Виведення інформації.

Виведення інформації з бази даних створене для усіх файлів. Розглянемо цей алгоритм на прикладі класу «Sportsman».

Метод виведення інформації спортсмена : з клавіатури вводиться спосіб пошуку потрібної інформації : усіх спортсменів, у яких змаганнях брали участь, кількість тренерів та види спорту. Через введення ідентифікаційного коду, відкривається файл «Sport.txt», у якому, перевіряється перше слово (код), якщо він ідентичний введеному – то виводиться стрічка інформації про спортсмена у певному порядку.

Крім того, є запити відповідно варіанту:

1. Отримати перелік спортивних споруд зазначеного типу в цілому або такі що задовольняють заданим характеристикам (наприклад, стадіони, що вміщають не менше вказаного числа глядачів).

2. Отримати список спортсменів, що займаються зазначеним видом спорту в цілому або не нижче певного розряду.

3. Отримати список спортсменів, що тренуються у якогось тренера в цілому або не нижче певного розряду.

4. Отримати список спортсменів, що займаються більш ніж одним видом спорту із зазначенням цих видів спорту.

5. Отримати список тренерів зазначеного спортсмена.

6. Отримати перелік змагань, проведених протягом заданого періоду часу в цілому або зазначеним організатором.

7. Отримати список призерів зазначеного змагання.

8. Отримати перелік змагань, проведених у зазначеній спортивній споруді в цілому або з певного виду спорту.

9. Отримати перелік спортивних клубів і число спортсменів цих клубів,

що брали участь у спортивних змаганнях протягом заданого інтервалу часу.

10. Отримати список тренерів за певним видом спорту.

11. Отримати список спортсменів, які не брали участь в жодних змаганнях

протягом певного періоду часу.

12. Отримати список організаторів змагань і число проведених ними змагань протягом певного періоду часу.

13. Отримати перелік спортивних споруд та дати проведення на них змагань протягом певного періоду часу.

2.2. Розміщення активних та пасивних елементів меню на робочому столі програми.

Дана інформаційна система є, по суті, реляційною базою даних. Реляційна база даних — база даних, заснована на реляційній моделі даних. Слово «реляційний» походить від англ. relation.

Можна провести аналогію між елементами реляційної моделі даних і елементами моделі «сутність-зв’язок». Реляційні відносини відповідають наборам сутностей, а кортежі — сутностям. Тому, як і в моделі «сутність-зв’язок», стовпці в таблиці, що представляє реляційне відношення, називають атрибутами.

Кожен атрибут визначений на домені, тому домен можна розглядати як множина допустимих значень даного атрибуту. Кілька атрибутів одних відношень і навіть атрибути різних відношень можуть бути визначені на одному і тому ж домені.

Отже, дані будуть виведені в консоль у вигляді таблиць. Для того, щоб

таблиці коректно відображалися в консолі, необхідно застосувати керуючі символи.

Керуючі символи (або як їх ще називають – escape-послідовність) – символи які виштовхуються в потік виводу, з метою форматування виводу або друку деяких керуючих знаків C++ . Найчастіше використовуються такі керуючі символи: \t (горизонтальна табуляція), \n (новий рядок).

Також було б добре встановити вирівнювання по лівому краю за допомогою setiosflags(ios::left).
Щоб дані, що виводяться в консоль мали справді вигляд таблиці варто застосувати символи | та _.

2.3 Прикладне програмування АІС та взаємодія користувача з нею.

Користування та використання базою даних не є складною процедурою. Візьмемо реляційну базу даних згідно мого варіанту (тобто та, яка представлена у вигляді таблиці). Кожен запис в реляційній базі даних розкладається на кілька осередків.

При першому запуску нас вітає меню заданої бази даних. Кожному блоку меню присвоєно клавішу від 0 до 6. Якщо Ви вводите з клавіатури цифру 1- ви маєте змогу зчитати дані з файлу або ввести їх вручну, тобто занести дані в середовище для роботи з ними. При введені цифри 2 можна переглянути, які саме дані Ви занесли до системи. Пункт меню під номером 3 запрограмований для надання вам змоги змінити введені дані і вибрати, під яким номером знаходяться ті дані, які потрібно змінити або видалити. Клавіша 4 може додати нові дані, якщо їх немає в базі даних. 5 та 6 відповідають методам сортування та видалення даних. Якщо дані не є на даний час актуальні, то ви можете їх видалити або просто очистити повністю дані з бази, а також відсортувати всі дані за алфавітним порядком. Коли

робота з базою даних закінчена, Ви можете вийти з неї, нажавши клавішу 0 на клавіатурі, при цьому дані будуть збережені у файлі програми.

Також передбачені методи для виводу окремих даних за умовою. Нижче головних пунктів меню можна отримати інформацію окремо про тренерів, спортсменів, змагань, спортивної інфраструктури Івано-Франківська, змагань та спонсорів, ввівши латинські літери від a до f відповідно. В кожному пункті можна переглянути дані за певним видом запиту. Якщо це стосується змагань, то програма виведе на екран змагання за інтервал часу, який ви введете або список спортсменів, які займаються певним видом спорту або зайняли призові місця на змаганнях.

Програма передбачає введення неправильних запитів або введення клавіші, яка не запрограмована в коді програми. Якщо користувач введе цифру, більшу за 6 або літеру, яка не відповідає заданому інтервалу, то програма видасть вам помилку та запропонує зробити ваш вибір ще раз.

2.4 Особливості розробки активних елементів системи

У даній курсовій роботі активними програмними елементами є об’єкти класів. Щоб створити якісні класи, необхідно чітко розуміти принципи ООП (об’єктно-орієнтованого програмування). ООП – це одна з парадигм програмування, яка розглядає програму як множину об'єктів, що взаємодіють між собою.

Інкапсуляція — один з трьох основних механізмів об'єктно-орієнтованого програмування. Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів (методів).

Успадкування (наслідування) — механізм утворення нових класів на основі використання вже існуючих. При цьому властивості та

функціональність батьківського класу переходять до класу нащадка (дочірнього).
Поліморфізм — концепція в програмуванні та теорії типів, в основі якої лежить використання єдиного інтерфейсу для різнотипних сутностей або у використанні однакового символу для маніпуляцій над даними різного типу.
Абстракція – це додання об'єкту характеристик, які відрізняють його від всіх об'єктів, чітко визначаючи його концептуальні межі.

Отже, у цій курсовій роботі буде створено кілька базових загальних класів, від яких будуть походити інші, більш спеціалізовані класи-нащадки. Всі змінні будуть інкапсульованими, тобто захищеними від будь-яких випадкових змін, а методи (функції) будуть відкритими і служитимуть для зв’язку з захищеними членами. Інкапсуляція здійснюється за допомогою одного з трьох специфікаторів доступу (public, protected, public), вибір яких обгрунтовано в наступних розділах.

2.5 Особливості розробки активних шаблонів

Шаблони проектування програмного забезпечення — ефектні способи вирішення задач проектування програмного забезпечення. Шаблон не є закінченим зразком, який можна безпосередньо транслювати в програмний код. Об'єктно-орієнтований шаблон найчастіше є зразком вирішення проблеми і відображає відношення між класами та об'єктами, без вказівки на те, як буде зрештою реалізоване це відношення. Шаблон призначений для інформування інших користувачів про свою активність.

Робота будь-якої інформаційної системи забезпечується наявністю базових функцій. Наприклад, функції сортування, пошуку, заповнення бази даних, виведення повідомлень про помилки, виведення на екран окремих елементів та їхню кількість, додавання нових елементів та виведення інших, тих, які вже не потрібні.

Для зберігання цих методів можна створити окремий клас в окремому файлі і назвати його, наприклад, Functions, а можна створити ці методи у відповідних їм класах.

Отже, окрім властивостей, класи мають ще й методи. Метод — підпрограма (процедура, функція), що використовується виключно разом із класом (методи класу) або з об'єктом (методи екземпляра).

Окрім цих методів інформаційна система має мати ще й методи обробки помилок введення даних, для можна створити окремий загальний клас Exceptions, що має похідні клас

2.6 Розрахунок якості інтерфейсу користувача

Інтерфейс користувача, з точки зору дизайну — це програмне середовище, де відбувається обмін інформації між людьми та технікою. Метою створення інтерфейсу є спрощення роботи користувача з тією чи іншою системою, щоб забезпечити бажаний результат.

Інтерфейс повинен володіти такими якостями:

- ясність;
- виразність
- естетика;
- відповідність;
- продуктивність.

На даний момент ми маємо справу з консольною базою даних, для якої потрібно розробити інтерфейс, який буде зрозумілий користувачу.

Найперше, що варто зробити – це ввести SetConsoleCP(1251) SetConsoleOutputCP(1251)для того, щоб при введені і виводі(функції cout і

cin) інформації на українській мові вивелись українські літери. Якщо записувати дані англійською мовою, то кодування нам не знадобиться.

Наступним кроком буде реалізація індивідуального інтерфейсу. Для цього я використав функцію system("color 7c"). Отже, щоб змінити фон, будемо використовувати функцію system, в яку будемо передавати рядок наступного виду: "color <A> <B>", де <A> і <B> - шістнадцяткові цифри - перша задає колір фону, а друга - колір переднього плану (колір шрифту). В моєму випадку колір фону буде білий, а колір шрифту буде світло - червоний.

Також, для гарного вигляду та розміщення меню я використовую табуляцію. В даному проекті я використав **\t-** горизонтальна вкладка. Він використовується для надання місця табуляції горизонтально у вашому виході. **\n** – виводить текст на нову строку. Також є функція endl, яку я також використав. Вона означає, що текст, який буде виводитися далі, почнеться на наступному рядку. Через що часто вважається, що endl еквівалентно '\ n' в кінці тексту. Ну і для кращого вигляду я використав символи | та _ для оформлення меню та чітких його границь.

2.7 Розрахунок швидкості заповнення форм даних GOMS.

Одним з кращих підходів до кількісного аналізу моделей інтерфейсів є класична модель GOMS - «правила для цілей, об'єктів, методів і виділення» (the model of goals, objects, methods and selection rules). Моделювання GOMS дозволяє передбачити, скільки часу буде потрібно досвідченому користувачеві на виконання конкретної операції при використанні даної моделі інтерфейсу.

Тут необхідно обговорити тільки один найпростіший, але досить цінний аспект методу GOMS - модель, заснована на оцінці швидкості друку. Розробники, які знайомі з методом GOMS, рідко проводять детальний і формальний аналіз моделі інтерфейсу. Мета GOMS – підвищення

ефективності взаємодії між людиною та комп’ ютером шляхом усунення непотрібних дій.

GOMS розшифровується як:

G → Цілі

O → Оператори

M → Методи

S → Вибір

Розглянемо суть даної системи на прикладі бази даних. Цілі – це завдання, яке потрібно виконати студенту. У нашому випадку – розробити власну базу даних та проаналізувати її функціонал. Оператори – це дії, необхідні для реалізації цілі. В програмному середовищі це класи, об’єкти, методи і файли. Наступною категорією є методи, які передбачені програмістом, щоб користувачеві був зрозумілий функціонал програми. Це можуть бути вказівки при неправильному вводі з клавіатури, яку клавішу потрібно нажати, щоб додати або зчитати дані. Виходячи з вище перелічених дій, користувач обирає дію, яка надасть йому потрібну інформацію. Це може бути вибір списку всіх спортсменів за заданим параметром, або вибір змагань і їхніх спонсорів.GOMS базується на етапі дослідження з кінцевими користувачами, і це може стати потужним еталоном аналізу поведінки користувачів. Це допомагає усунути непотрібні дії, тому це економить час і витрати.

2.8 Моделювання та тестування роботи АІС

Ієрархія класів - це структура численних пов'язаних класів, яка визначає, які класи успадковують функції від інших класів. Спадкування функцій породжується спадкуванням класів. Ієрархія класів дозволяє визначати нові класи на основі вже наявних. Наявні класи зазвичай називають базовими (іноді породжують), а нові класи, формуються на основі базових, - похідними (породженими, класами-нащадками або спадкоємцями).

Похідні класи «отримують спадок» - дані і методи своїх класів, і можуть поповнюватися власними компонентами (даними та власними методами). Успадковані компоненти не переміщаються в похідний клас, а залишаються в базових класах. Повідомлення, обробку якого не можуть виконати методи похідного класу, автоматично передається в базовий клас. Якщо для обробки повідомлення потрібні дані, відсутні в похідному класі, то їх намагаються відшукати автоматично в базовому класі. На прикладі моєї курсової роботи першим класом буде Trainer, який вміщує в собі ім’я та прізвище тренера. Далі йде клас Kind_of_Sport, який містить вид спорту та тренера, який тренує даний вид спорту. Наступний клас називається Sportsman, зберігає дані, такі як ім’я спортсмена, зайняте місце у змаганнях, вид спорту. Клас Competitions: назва змагання, початок і кінець змагання, які спортсмени беруть участь. Клас Sponsor містить ім’я спонсора і змагання, які він фінансує. Головний (батьківський) клас Buildings містить ім’я інфраструктури, площу, підтримку спонсора. Від даного класу є похідні Court з атрибутом «Тип покриття», Stadion «Місткість глядачів», Gym з кількістю тренажерів. Всі вище клас зв’язані між собою вказівником на адресу класу попереднього і дає змогу отримати доступ до даних при формуванні запитів та логічно пов’язати їх між собою.

Тестування програмного забезпечення - це процес визначення працездатності програмного забезпечення відповідно до очікувань, в нашому випадку – правильний функціонал бази даних. Цей конкретний процес є перевіркою зіставлень полів з точки зору кінцевого користувача. У цьому конкретному сценарії тестер буде виконувати операцію на рівні бази даних, а потім переходити до відповідного елементу призначеного для користувача інтерфейсу, щоб спостерігати і перевіряти, чи були виконані правильні перевірки полів чи ні.

Навпаки, умова, при якому тестер спочатку виконує операцію на призначеному для користувача інтерфейсі, а потім перевіряється на бекенда, також вважається допустимим варіантом.

При написанні коду, я програмував методи в невеликих кількостях або в окремому проекті. Наприклад, щоб протестувати, чи працює метод введення даних про спортсмена, я створив окремий проект, де реалізував клас і прописав код для нього. Якщо все працювало коректно, то додавав його в готовий проект. Так бажано робити поступово, щоб уникнути помилок в написанні коду. Набагато краще писати невеликі функції, а потім відразу їх компілювати і тестувати. Таким чином, якщо ви допустили помилку, ви будете знати, що вона знаходиться в невеликій кількості коду, який ви тільки що написали / змінили.

Коли проект повністю готовий, потрібно його протестувати на правильність вводу та виводу інформації, перевірити, чи всі методи працюють коректно. З даного тестування ми можемо розширити функціонал програми, виправити баги в системі або покращити інтерфейс власної програми,

ВИСНОВКИ

Виконання курсової роботи була із метою освоєння усіх теоретичних знань, отриманих з дисципліни «Об’єктно-орієнтоване програмування» і використання їх на практиці програмування мовою С++.

У цій курсовій роботі було зроблено задачу розробки проекту консольної бази даних по предметній області «Розробники комп’ютерних ігор». У ній передбачалася робота з функціями та файлами, виділенням пам’яті, алгоритмом пошуку унікальних значень і т. п.

Програму було написано у середовищі Visual Studio – С++. Перевірка роботи програмі і її тестування показали, що завдання реалізовано коректно.

Розроблена система використання баз даних і їх управління, завдяки чому, можна повноцінно реалізовувати бази даних і можливі маніпуляції з ними, завдяки основних розроблених функцій, які надають можливість роботи з базами даних і навіть можливість використання власних методів.

Під час написання курсової роботи освоювались основні бібліотеки мови С++ і їх засоби реалізації, а також були поглиблені знання про роботу з консоллю і базами.

Вивчивши та проаналізувавши мову С++, зроблено висновок, що ця мова є досить гнучким і доступним інструментом для роботи з базами даних і також використання у проєктах, що реалізовані на різних мовах програмування.

Отже, можна зробити висновок, що С++ - це мова програмування високого рівня, яка попри виконання інших завдань, підходить і для використання бази даних.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мачулянський О.В. Моделювання засобами С++: навч. посіб. / О.В. Мачулянський, Д.Д. Татарчук. – Київ: КПІ , 2009. - 14 с.

2. Юрчишин В.М. Програмування: навч. посіб. / В. М. Юрчишин, Б. В. Клим, В. Б. Кропивницька. - Івано-Франківськ: ІФНТУНГ, 2012. - 188 с.

3. Інтернет-джерело «Вікіпедія».

4. Cplusplus.com

5. Cyberforum.ru

6. Stackoverflow.com

7. studfile.net

Додатки

Додаток А

Код програми мовою C++

DataBase_sport_IF.cpp

```
#include "Data.h"
#include <iostream>
#include "windows.h"

using namespace std;


void Functions::Menu(Stadion* stadion, int stadion_count, Gym* gym, int
gym_count, Court* court, int court_count)
{
	int _stateMenu;
	cout
		<< "\t\t\t	|-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=|" << endl
		<< "\t\t\t	|    База даних організацій    |" << endl
		<< "\t\t\t	|------------------------------|" << endl
		<< "\t\t\t	|Виберіть дію:                 |" << endl
		<< "\t\t\t	|(0) Вихід із програми         |" << endl
		<< "\t\t\t	|(1) Спортивна інфраструктура  |" << endl
		<< "\t\t\t	|(2) Дані про змагання         |" << endl
		<< "\t\t\t	|(3) Дані про спортсменів      |" << endl
		<< "\t\t\t	|(4) Дані про тренерів         |" << endl
		<< "\t\t\t	|(5) Дані про призерів змагань |" << endl
		<< "\t\t\t	|(6) Ввід даних                |" << endl
		<< "\t\t\t	|-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=|" << endl
		<< "\t\t\t: ";

	cout << "Введіть номер: " << endl;
	cin >> _stateMenu;

	while (_stateMenu != 0) {


		switch (_stateMenu)
		{
		case 1:
			system("cls");
			Functions::Buildings_Sport(stadion, stadion_count, gym,
gym_count, court, court_count);
			break;
		case 2:
			system("cls");
```

```
                Functions::Competitions(stadion, stadion_count, gym,
    gym_count, court, court_count);
                break;
            case 3:
                system("cls");
                Functions::Sportsmans(stadion, stadion_count, gym,
    gym_count, court, court_count);
                break;
            case 4:
                system("cls");
                Functions::Trainers(stadion, stadion_count, gym,
    gym_count, court, court_count);
                break;
            case 5:
                system("cls");
                Functions::Achievement(stadion, stadion_count, gym,
    gym_count, court, court_count);
                break;
            case 0:
                system("cls");
                cout << "Завершення роботи" << endl << endl;
                break;
            default:
                system("cls");
                Functions::Menu(stadion, stadion_count, gym, gym_count,
    court, court_count);
                cout << "Помилка! Спробуйте ще раз" << endl << endl;
                break;
            }
        }
};
```

Файл Data.h

```
class Trainer
{
public:
	char trainer_name[100];
};
class Kind_of_Sport
{
public:
	char sport_name[100];
	int trainer_count;
	Trainer* trainer;
};
class Sportsman
{
public:
	char sportsman_name[100];
	int place;
	int sport_count;
	Kind_of_Sport* sport;
};
class Competitions
{
public:
	char competitions_name[100];
	int sportsman_quantity;
	int date_start;
	int date_end;
	Sportsman* sportsman;
};
class Sponsor
{
public:
	char sponsor_name[30];
	int competitions_quantity;
	Competitions* competitions;
};
class Buildings
{
public:
```

```
        char buildings_name[30];
        int area;
        int sponsor_count;
        Sponsor* sponsor;
};
class Stadion : public Buildings
{
public:
        int capacity;
};
class Gym : public Buildings
{
public:
        int count_simulators;
};
class Court : public Buildings
{
public:
        char type[30];
};

class Functions
{
public:
        static void Menu(Stadion* stadion, int stadion_count, Gym* gym,
int gym_count, Court* court, int court_count);
        static void Competitions(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count);
        static void Sportsmans(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count);
        static void Trainers(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count);
        static void Achievement(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count);
        static void Buildings_Sport(Stadion* stadion, int stadion_count,
Gym* gym, int gym_count, Court* court, int court_count);
};

class Data {
private:
        Trainer trainer;
        Kind_of_Sport kind_of_sport;
        Sportsman sportsman;
        Competitions competitions;
        Buildings buildings;
```

```
        Sponsor sponsor;
        Stadion stadion;
        Gym  gym;
        Court court;
public:
        Data();
        Data(Trainer trainer_, Kind_of_Sport kind_of_sport_, Sportsman
sportsman_, Competitions competitions_, Buildings buildings_, Sponsor
sponsor_,
                Stadion stadion_, Gym gym_, Court court_);
        ~Data();


        void DataEntry(Trainer trainer_, Kind_of_Sport kind_of_sport_,
Sportsman sportsman_, Competitions competitions_, Buildings buildings_,
Sponsor sponsor_,
                Stadion stadion_, Gym gym_, Court court_);

        Trainer GetTrainer() { return trainer; };
        Kind_of_Sport GetKind_of_Sport() { return kind_of_sport; };
        Sportsman GetSportsman() { return sportsman; };
        Competitions GetCompetitions() { return competitions; };
        Buildings GetBuildings() { return buildings; };
        Sponsor GetSponsor() { return sponsor; };
        Stadion GetStadion() { return stadion; };
        Gym  GetGym() { return gym; };
        Court GetCourt() { return court; };

        Data& operator = (Data d_o);

};
```

Файл Functions.cpp

```
#include <iostream>
#include <string>
#include "Data.h"
#include "windows.h"

using namespace std;


void main() {
	system("color 3f");
	SetConsoleCP(1251);
	SetConsoleOutputCP(1251);


	int stadion_count, gym_count, court_count;
	Stadion* stadion;
	Gym* gym;
	Court* court;


	do {
		cout << "Введіть кількість стадіонів(> 0): ";
		cin >> stadion_count;
		cout << endl;
		system("cls");
	} while (stadion_count <= 0);
	do {
		cout << "Введіть кількість спортзалів(> 0): ";
		cin >> gym_count;
		cout << endl;
		system("cls");
	} while (gym_count <= 0);
	do {
		cout << "Введіть кількість кортів для тенісу(> 0): ";
		cin >> court_count;
		cout << endl;
		system("cls");
	} while (court_count <= 0);

	stadion = new Stadion[stadion_count];
	gym = new Gym[gym_count];
	court = new Court[court_count];

	for (int i = 0; i < stadion_count; i++)
	{
		cout << "Введіть назву стадіона № " << i + 1 << ": ";
		cin.ignore();
```

```
            cin.getline(stadion[i].buildings_name, 30);
            cout << endl;
            system("cls");
            do {
                  cout << "Введіть площу (> 0): ";
                  cin >> stadion[i].area;
                  cout << endl;
                  system("cls");
            } while (stadion[i].area <= 0);

            do {
                  cout << "Введіть вмістимість стадіона(> 0): ";
                  cin >> stadion[i].capacity;
                  cout << endl;
                  system("cls");
            } while (stadion[i].capacity <= 0);

            do {
                  cout << "Кількість спонсорів стадіона(> 0): ";
                  cin >> stadion[i].sponsor_count;
                  cout << endl;
                  system("cls");
            } while (stadion[i].sponsor_count <= 0);

            stadion[i].sponsor = new Sponsor[stadion[i].sponsor_count];
            for (int j = 0; j < stadion[i].sponsor_count; j++)
            {
                  cout << "Введіть ім'я спонсора № " << j + 1 << ": ";
                  cin.ignore();
                  cin.getline(stadion[i].sponsor[j].sponsor_name, 30);
                  cout << endl;
                  system("cls");
                  do {
                        cout << "Кількість змагань, закріплених за
спонсором(> 0): ";
                        cin >>
stadion[i].sponsor[j].competitions_quantity;
                        cout << endl;
                        system("cls");
                  } while (stadion[i].sponsor[j].competitions_quantity <=
0);

                  stadion[i].sponsor[j].competitions = new
Competitions[stadion[i].sponsor[j].competitions_quantity];


                  for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                  {
                        cout << "Назва змагання №" << m + 1 << ": ";
                        cin.ignore();
```

```
      cin.getline(stadion[i].sponsor[j].competitions[m].competitions_nam
e, 30);
                        cout << endl;
                        system("cls");
                        cout << "Дата початку змагання: ";
                        cin >>
stadion[i].sponsor[j].competitions[m].date_start;
                        cout << endl;
                        system("cls");

do {
                              cout << "Дата закінчення змагання: ";
                              cin >>
stadion[i].sponsor[j].competitions[m].date_end;
                              cout << endl;
                              system("cls");
                        } while
(stadion[i].sponsor[j].competitions[m].date_start >
stadion[i].sponsor[j].competitions[m].date_end);

                        do {
                              cout << "Кількість спортсменів, які брали
участь у змаганнях: ";
                              cin >>
stadion[i].sponsor[j].competitions[m].sportsman_quantity;
                              cout << endl;
                              system("cls");
                        } while
(stadion[i].sponsor[j].competitions[m].sportsman_quantity <= 0);

                        stadion[i].sponsor[j].competitions[m].sportsman =
new Sportsman[stadion[i].sponsor[j].competitions[m].sportsman_quantity];


                        for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                        {
                              cout << "Введіть ім'я спортсмена № " << n + 1
<< ": ";
                              cin.ignore();

      cin.getline(stadion[i].sponsor[j].competitions[m].sportsman[n].spo
rtsman_name, 30);
                              cout << endl;
                              system("cls");
                              do {
                                    cout << "Місце, яке заняв даний
спортсмен: ";
                                    cin >>
stadion[i].sponsor[j].competitions[m].sportsman[n].place;
```

```
                                    cout << endl;
                                    system("cls");
                               } while
(stadion[i].sponsor[j].competitions[m].sportsman[n].place <= 0);

                               do {
                                    cout << "Кількість видів спорту, якими
займається спортсмен: ";
                                    cin >>
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count;
                                    cout << endl;
                                    system("cls");
                               } while
(stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count <= 0);


     stadion[i].sponsor[j].competitions[m].sportsman[n].sport = new
Kind_of_Sport[stadion[i].sponsor[j].competitions[m].sportsman[n].sport_c
ount];


                               for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                               {
                                    cout << "Назва виду спорту № " << a + 1
<< ", яким займається спортсмен: ";
                                    cin.ignore();

     cin.getline(stadion[i].sponsor[j].competitions[m].sportsman[n].spo
rt[a].sport_name, 30);
                                    system("cls");
                                    cout << endl;

                                    do {
                                         cout << "Кількість тренерів, у
яких займається спортсмен даним видом спорту: ";
                                         cin >>
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_coun
t;
                                         cout << endl;
                                         system("cls");
                                    } while
(stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_cou
nt <= 0);


     stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].traine
r = new
Trainer[stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trai
ner_count];
```

```
                                    for (int b = 0; b <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_coun
t; b++)
                                    {
                                          cout << "Введіть ім'я тренера № "
<< b + 1 << ": ";
                                          cin.ignore();

      cin.getline(stadion[i].sponsor[j].competitions[m].sportsman[n].spo
rt[a].trainer[b].trainer_name, 30);
                                          cout << endl;
                                          system("cls");
                                    }
                              }
                        }
                  }
            }
      }
      cout << endl;
      ////////////////////////////////////////////////////////////////////
////////////////////////////////
      for (int i = 0; i < gym_count; i++)
      {
            cout << "Введіть назву спортзала № " << i + 1 << ": ";
            cin.ignore();
            cin.getline(gym[i].buildings_name, 30);
            cout << endl;
            system("cls");
            do {
                  cout << "Площа спортзалу(> 0): ";
                  cin >> gym[i].area;
                  cout << endl;
                  system("cls");
            } while (gym[i].area <= 0);

            do {
                  cout << "Кількість тренажерів у залі(> 0): ";
                  cin >> gym[i].count_simulators;
                  cout << endl;
                  system("cls");
            } while (gym[i].count_simulators <= 0);

            do {
                  cout << "Кількість спонсорів спортзала(> 0): ";
                  cin >> gym[i].sponsor_count;
                  cout << endl;
                  system("cls");
            } while (gym[i].sponsor_count <= 0);



gym[i].sponsor = new Sponsor[gym[i].sponsor_count];
```

```
            for (int j = 0; j < gym[i].sponsor_count; j++)
            {
                  cout << "Введіть спонсора № " << j + 1 << ": ";
                  cin.ignore();
                  cin.getline(gym[i].sponsor[j].sponsor_name, 30);
                  cout << endl;
                  system("cls");
                  do {
                        cout << "Кількість змагань, за які відповідає
спонсор(> 0): ";
                        cin >> gym[i].sponsor[j].competitions_quantity;
                        cout << endl;
                        system("cls");
                  } while (gym[i].sponsor[j].competitions_quantity <= 0);

                  gym[i].sponsor[j].competitions = new
Competitions[gym[i].sponsor[j].competitions_quantity];
                  for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                  {
                        cout << "Назва змагання № " << m + 1 << ": ";
                        cin.ignore();

      cin.getline(gym[i].sponsor[j].competitions[m].competitions_name,
30);
                        cout << endl;
                        system("cls");
                        cout << "Дата початку змагання: ";
                        cin >>
gym[i].sponsor[j].competitions[m].date_start;
                        cout << endl;
                        system("cls");
                        do {
                              cout << "Дата закінчення змагання: ";
                              cin >>
gym[i].sponsor[j].competitions[m].date_end;
                              cout << endl;
                              system("cls");
                        } while
(gym[i].sponsor[j].competitions[m].date_start >
gym[i].sponsor[j].competitions[m].date_end);

                        do {
                              cout << "Кількість спортсменів, які брали
участь у змаганнях: ";
                              cin >>
gym[i].sponsor[j].competitions[m].sportsman_quantity;
                              cout << endl;
                              system("cls");
                        } while
(gym[i].sponsor[j].competitions[m].sportsman_quantity <= 0);
```

```
                        gym[i].sponsor[j].competitions[m].sportsman = new
Sportsman[gym[i].sponsor[j].competitions[m].sportsman_quantity];
                        for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                        {
                              cout << "Введіть ім'я спортсмена № " << n + 1
<< ": ";
                              cin.ignore();

      cin.getline(gym[i].sponsor[j].competitions[m].sportsman[n].sportsm
an_name, 30);
                              cout << endl;
                              system("cls");
                              do {
                                    cout << "Введіть місце, яке заняв
спортсмен: ";
                                    cin >>
gym[i].sponsor[j].competitions[m].sportsman[n].place;
                                    cout << endl;
                                    system("cls");
                              } while
(gym[i].sponsor[j].competitions[m].sportsman[n].place <= 0);

                              do {
                                    cout << "Кількість видів спорту, якими
займається спортсмен: ";
                                    cin >>
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count;
                                    cout << endl;
                                    system("cls");
                              } while
(gym[i].sponsor[j].competitions[m].sportsman[n].sport_count <= 0);


      gym[i].sponsor[j].competitions[m].sportsman[n].sport = new
Kind_of_Sport[gym[i].sponsor[j].competitions[m].sportsman[n].sport_count
];
                              for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                              {
                                    cout << "Назва виду спорту № " << a + 1
<< ", яким займається даний спортсмен: ";
                                    cin.ignore();

      cin.getline(gym[i].sponsor[j].competitions[m].sportsman[n].sport[a
].sport_name, 30);
                                    cout << endl;
                                    system("cls");
                                    do {
```

```
                                        cout << "Кількість тренерів, у
яких займається спортсмени даним видом спорту: ";
                                        cin >>
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
                                        cout << endl;
                                        system("cls");
                                  } while
(gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count
<= 0);


      gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer =
new
Trainer[gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_
count];
                                  for (int b = 0; b <
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                  {
                                        cout << "Ім'я тренера № " << b + 1
<< ": ";
                                        cin.ignore();

      cin.getline(gym[i].sponsor[j].competitions[m].sportsman[n].sport[a
].trainer[b].trainer_name, 30);
                                        cout << endl;
                                        system("cls");
                                  }
                            }
                      }
                }
          }
      }
      cout << endl;
      ////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////
      for (int i = 0; i < court_count; i++)
      {
            cout << "Введіть назву корта для тенісу № " << i + 1 << ": ";
            cin.ignore();
            cin.getline(court[i].buildings_name, 30);
            cout << endl;
            system("cls");
            do {
                  cout << "Площа корта(> 0): ";
                  cin >> court[i].area;
                  cout << endl;
                  system("cls");
            } while (court[i].area <= 0);

            cout << "Тип покриття корта: ";
```

```
            cin.ignore();
            cin.getline(court[i].type, 30);
            cout << endl;
            system("cls");
            do {
                  cout << "Кількість спонсорів корта(> 0): ";
                  cin >> court[i].sponsor_count;
                  cout << endl;
                  system("cls");
            } while (court[i].sponsor_count <= 0);

            court[i].sponsor = new Sponsor[court[i].sponsor_count];
            for (int j = 0; j < court[i].sponsor_count; j++)
            {
                  cout << "Введіть ім'я спонсора № " << j + 1 << ": ";
                  cin.ignore();
                  cin.getline(court[i].sponsor[j].sponsor_name, 30);
                  cout << endl;
                  system("cls");
                  do {
                        cout << "Кількість змагань, закріплених за
спонсором(> 0): ";
                        cin >> court[i].sponsor[j].competitions_quantity;
                        cout << endl;
                        system("cls");
                  } while (court[i].sponsor[j].competitions_quantity <=
0);

                  court[i].sponsor[j].competitions = new
Competitions[court[i].sponsor[j].competitions_quantity];
                  for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                  {
                        cout << "Назва змагання № " << m + 1 << ": ";
                        cin.ignore();

      cin.getline(court[i].sponsor[j].competitions[m].competitions_name,
30);
                        cout << endl;
                        system("cls");
                        cout << "Дата початку змагання: ";
                        cin >>
court[i].sponsor[j].competitions[m].date_start;
                        cout << endl;
                        system("cls");
                        do {
                              cout << "Дата закінчення змагання: ";
                              cin >>
court[i].sponsor[j].competitions[m].date_end;
                              cout << endl;
                              system("cls");
```

```
                    } while
(court[i].sponsor[j].competitions[m].date_start >
court[i].sponsor[j].competitions[m].date_end);

                    do {
                          cout << "Кількість спортсменів, які брали
участь у змаганні: ";
                          cin >>
court[i].sponsor[j].competitions[m].sportsman_quantity;
                          cout << endl;
                          system("cls");
                    } while
(court[i].sponsor[j].competitions[m].sportsman_quantity <= 0);

                    court[i].sponsor[j].competitions[m].sportsman =
new Sportsman[court[i].sponsor[j].competitions[m].sportsman_quantity];
                    for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                    {
                          cout << "Ім'я спортсмена №" << n + 1 << ": ";
                          cin.ignore();

     cin.getline(court[i].sponsor[j].competitions[m].sportsman[n].sport
sman_name, 30);
                          cout << endl;
                          system("cls");
                          do {
                                cout << "Місце, яке заняв спортсмен у
даному змаганні: ";
                                cin >>
court[i].sponsor[j].competitions[m].sportsman[n].place;
                                cout << endl;
                                system("cls");
                          } while
(court[i].sponsor[j].competitions[m].sportsman[n].place <= 0);

                          do {
                                cout << "Кількість видів спорту, якими
займається спортсмен: ";
                                cin >>
court[i].sponsor[j].competitions[m].sportsman[n].sport_count;
                                cout << endl;
                                system("cls");
                          } while
(court[i].sponsor[j].competitions[m].sportsman[n].sport_count <= 0);


     court[i].sponsor[j].competitions[m].sportsman[n].sport = new
Kind_of_Sport[court[i].sponsor[j].competitions[m].sportsman[n].sport_cou
nt];
```

```
                              for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                              {
                                    cout << "Назва виду спорта № " << a + 1
<< ", Яким займається спортсмен: ";
                                    cin.ignore();

      cin.getline(court[i].sponsor[j].competitions[m].sportsman[n].sport
[a].sport_name, 30);
                                    cout << endl;
                                    system("cls");
                                    do {
                                          cout << "Кількість тренерів, у
яких займаються спортсмени даним видом спорту: ";
                                          cin >>
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
                                          cout << endl;
                                          system("cls");
                                    } while
(court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count
<= 0);

court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer = new
Trainer[court[i].sponsor[j].competitions[m].sportsman[n].sport[a].traine
r_count];
                                    for (int b = 0; b <
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                    {
                                          cout << "Введіть ім'я тренера № "
<< b + 1 << ": ";
                                          cin.ignore();

      cin.getline(court[i].sponsor[j].competitions[m].sportsman[n].sport
[a].trainer[b].trainer_name, 30);
                                          cout << endl;
                                          system("cls");
                                    }
                              }
                        }
                  }
            }
      }
      cout << endl;
      system("cls");

      Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}
```

Файл Data.cpp

```
#include <iostream>
#include <string>
#include "Data.h"

using namespace std;
```

```
void Functions::Buildings_Sport(Stadion* stadion, int stadion_count,
Gym* gym, int gym_count, Court* court, int court_count)
{
	cout << "Список споруд певного типу: 1" << endl;
	cout << "Список споруд і дати проведення змагань на певному
проміжку часу: 2" << endl << endl;

	string building_name;
	int number, start_of_competitions, end_of_competitions, count = 0;
	cout << "Введіть номер: ";
	cin >> number;
	cout << endl;

	switch (number)
	{
	case 1:
		cout << "Введіть тип (стадіон, спортзал или корт): ";
		cin >> building_name;
		cout << endl;

		if (building_name == "стадіон")
		{
			cout << "Назва стадіонів:" << endl;
			for (int i = 0; i < stadion_count; i++)
				cout << i + 1 << ". " << stadion[i].buildings_name
<< endl;
			cout << endl;
		}
		else if (building_name == "спортзал")
		{
			cout << "Назва спортзалів:" << endl;
			for (int i = 0; i < gym_count; i++)
				cout << i + 1 << ". " << gym[i].buildings_name <<
endl;
			cout << endl;
		}
		else if (building_name == "корт")
		{
			cout << "Назва кортів для тенісу:" << endl;
			for (int i = 0; i < court_count; i++)
				cout << i + 1 << ". " << court[i].buildings_name
<< endl;
			cout << endl;
		}
		else
			cout << "Помилка! Спробуйте ще раз" << endl << endl;
		break;
	case 2:
		cout << "Введіть дату початку змагання: ";
		cin >> start_of_competitions;
```

```
            cout << endl;

            cout << "Введіть дату закінчення змагання: ";
            cin >> end_of_competitions;
            cout << endl;

            cout << "Змагання, проведені на стадіоні:" << endl;
            for (int i = 0; i < stadion_count; i++)
            {
                  for (int j = 0; j < stadion[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                        {
                              if
(stadion[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
                              {
                                    if
(stadion[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "Стадіон № " << i + 1 <<
": " << stadion[i].buildings_name << endl;
                                          cout << "Назва змагання: " <<
stadion[i].sponsor[j].competitions[m].competitions_name << endl;
                                          cout << "Дата початку змагання: "
<< stadion[i].sponsor[j].competitions[m].date_start << endl;
                                          cout << "Дата закінчення змагання:
" << stadion[i].sponsor[j].competitions[m].date_end << endl << endl;
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено стадіонів, на яких проводились
змагання в певному періоді часу" << endl << endl;
            count = 0;

      ////////////////////////////////////////////////////////////////
////////
            cout << "Змагання у спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
```

```
                              if
(gym[i].sponsor[j].competitions[m].date_start >= start_of_competitions)
                              {
                                    if
(gym[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "Спортзал № " << i + 1 <<
": " << gym[i].buildings_name << endl;
                                          cout << "Назва змагання: " <<
gym[i].sponsor[j].competitions[m].competitions_name << endl;
                                          cout << "Дата початку змагання: "
<< gym[i].sponsor[j].competitions[m].date_start << endl;
                                          cout << "Дата закінчення змагання:
" << gym[i].sponsor[j].competitions[m].date_end << endl << endl;
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортзалів, на яких проводились
змагання в певному періоді часу" << endl << endl;
            count = 0;

            cout << "Змагання на кортах:" << endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {
                              if
(court[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
                              {
                                    if
(court[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "Корт № " << i + 1 << ": "
<< court[i].buildings_name << endl;
                                          cout << "Назва змагання: " <<
court[i].sponsor[j].competitions[m].competitions_name << endl;
                                          cout << "Дата початку змагання: "
<< court[i].sponsor[j].competitions[m].date_start << endl;
                                          cout << "Дата закінчення змагання:
" << court[i].sponsor[j].competitions[m].date_end << endl << endl;
                                          count++;
                                    }
                              }
```

```
                        }
                    }
                }
                if (count == 0)
                    cout << "Не знайдено кортів, на яких проводились
змагання в певному періоді часу" << endl << endl;
                count = 0;
                break;
            default:
                cout << "Помилка! Спробуйте ще раз" << endl << endl;
                break;
            }

            Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}

void Functions::Competitions(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count)
{
    cout << "Отримати список змагань, проведених на певному проміжку
часу: 1" << endl;
    cout << "Отримати список змагань, проведених в певному спортивному
комплексі: 2" << endl << endl;

    string buildings_name;
    int start_of_competitions, end_of_competitions, number, count = 0;
    cout << "Введіть номер: ";
    cin >> number;
    cout << endl;

    switch (number)
    {
    case 1:
        cout << "Введіть дату початку змагання: ";
        cin >> start_of_competitions;
        cout << endl;

        cout << "Введіть рік закінчення змагання: ";
        cin >> end_of_competitions;
        cout << endl;

        cout << "Змагання на стадіонах:" << endl;
        for (int i = 0; i < stadion_count; i++)
        {
            for (int j = 0; j < stadion[i].sponsor_count; j++)
            {
                for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                {
```

```
                              if
(stadion[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
                              {
                                    if
(stadion[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "- " <<
stadion[i].sponsor[j].competitions[m].competitions_name << endl;
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено змагань, проведених в даний
проміжок часу на стадіонах" << endl;
            cout << endl;
            count = 0;

      ////////////////////////////////////////////////////////////////
////////////////////////////////
            cout << "Змагання в спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              if
(gym[i].sponsor[j].competitions[m].date_start >= start_of_competitions)
                              {
                                    if
(gym[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "- " <<
gym[i].sponsor[j].competitions[m].competitions_name << endl;
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено змагань, проведених в даний
проміжок часу в спортзалах." << endl;
            cout << endl;
            count = 0;
```

```
     ////////////////////////////////////////////////////////////////
////////////////////////////////////
          cout << "Змагання на кортах для тенісу:" << endl;
          for (int i = 0; i < court_count; i++)
          {
               for (int j = 0; j < court[i].sponsor_count; j++)
               {
                    for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                    {
                         if
(court[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
                         {
                              if
(court[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                              {
                                   cout << "- " <<
court[i].sponsor[j].competitions[m].competitions_name << endl;
                                   count++;
                              }
                         }
                    }
               }
          }
          if (count == 0)
               cout << "Не знайдено змагань, проведених в даний
проміжок часу на кортах для тенісу" << endl;
          cout << endl;
          count = 0;
          break;

     case 2:
          cout << "Введіть назву споруди: ";
          cin >> buildings_name;
          cout << endl;

          cout << "Змагання на стадіонах:" << endl;
          for (int i = 0; i < stadion_count; i++)
          {

               if (buildings_name == stadion[i].buildings_name)
               {
                    for (int j = 0; j < stadion[i].sponsor_count; j++)
                    {
                         for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                              cout << "- " <<
stadion[i].sponsor[j].competitions[m].competitions_name << endl;
                    }
```

```
                        count++;
                  }
            }
            if (count == 0)
                  cout << "Не знайдено змагань, проведених  на даному
стадіоні" << endl;
            cout << endl;
            count = 0;

            cout << "Змагання у спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  if (buildings_name == gym[i].buildings_name)
                  {
                        for (int j = 0; j < gym[i].sponsor_count; j++)
                        {
                              for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                                    cout << "- " <<
gym[i].sponsor[j].competitions[m].competitions_name << endl;
                        }
                        count++;
                  }
            }
            if (count == 0)
                  cout << "Не знайдено змагань, проведених  в даному
спортзалі" << endl;
            cout << endl;
            count = 0;

            for (int i = 0; i < court_count; i++)
            {
                  cout << "Змагання на кортах для тенісу:" << endl;
                  if (buildings_name == court[i].buildings_name)
                  {
                        for (int j = 0; j < court[i].sponsor_count; j++)
                        {
                              for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                                    cout << "- " <<
court[i].sponsor[j].competitions[m].competitions_name << endl;
                        }
                        count++;
                  }
            }
            if (count == 0)
                  cout << "Не знайдено змагань, проведених  на даному
корті" << endl;
            cout << endl;
            count = 0;
            break;
```

```
	default:
		cout << "Помилка! Спробуйте ще раз" << endl << endl;
		break;
	}

	Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}

void Functions::Sportsmans(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count)
{
	cout << "Отримати список спортсменів, які займаються даним видом
спорту: 1" << endl;
	cout << "Отримати список спортсменів, які тренуються у якогось
тренера в цілому: 2" << endl;
	cout << "Отримати список спортсменів, які займаються одним і
більше видом спорту: 3" << endl << endl;

	string sport_type, trainer_name;
	int number, count = 0;
	cout << "Введіть номер: ";
	cin >> number;
	cout << endl;

	switch (number)
	{
	case 1:
		cout << "Оберріть вид спорту: ";
		cin >> sport_type;
		cout << endl;

		cout << "Спортсмени, які тренуються на стадіонах:" << endl;
		for (int i = 0; i < stadion_count; i++)
		{
			for (int j = 0; j < stadion[i].sponsor_count; j++)
			{
				for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
				{
					for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
					{
						for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
						{
							if (sport_type ==
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
							{
```

```
                                                cout << "- " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sportsman_name <<
endl;
                                                count++;
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які тренуються на
даному стадіоні вибраним видом спорту" << endl;
            cout << endl;
            count = 0;

      ////////////////////////////////////////////////////////////////
////////////////
            cout << "Спортсмени, які тренуються в спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          if (sport_type ==
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
                                          {
                                                cout << "- " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                                count++;
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які тренуються в
даному спортзалі вибраним видом спорту" << endl;
            cout << endl;
            count = 0;
```

```
     ////////////////////////////////////////////////////////////////////
////////////////////
          cout << "Спортсмени, які тренуються на кортах для тенісу:" <<
endl;
          for (int i = 0; i < court_count; i++)
          {
               for (int j = 0; j < court[i].sponsor_count; j++)
               {
                    for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                    {
                         for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                         {
                              for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                              {
                                   if (sport_type ==
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
                                   {
                                        cout << "- " <<
court[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                        count++;
                                   }
                              }
                         }
                    }
               }
          }
          if (count == 0)
               cout << "Не знайдено спортсменів, які тренуються на
даному корті вибраним видом спорту" << endl;
          cout << endl;
          count = 0;
          break;

     case 2:
          cout << "Введіть тренера: ";
          cin >> trainer_name;
          cout << endl;

          cout << "Спортсмени, які тренуються на стадіонах:" << endl;
          for (int i = 0; i < stadion_count; i++)
          {
               for (int j = 0; j < stadion[i].sponsor_count; j++)
               {
                    for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                    {
```

```
                              for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          for (int b = 0; b <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_coun
t; b++)
                                          {
                                                if (trainer_name ==
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].t
rainer_name)
                                                {
                                                      cout << "- " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sportsman_name <<
endl;
                                                      count++;
                                                }
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які тренуються на
даному стадіоні у тренера" << endl;
            cout << endl;
            count = 0;

            cout << "Спортсмени, які тренуються в спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          for (int b = 0; b <
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                          {
```

```
                                                if (trainer_name ==
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].train
er_name)
                                                {
                                                      cout << "- " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                                      count++;
                                                }
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які тренуються в
даному спортзалі у тренера" << endl;
            cout << endl;
            count = 0;

            cout << "Спортсмени, які тренуються на кортах для тенісу:" <<
endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          for (int b = 0; b <
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                          {
                                                if (trainer_name ==
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].tra
iner_name)
                                                {
                                                      cout << "- " <<
court[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                                      count++;
                                                }
                                          }
                                    }
                              }
                        }
                  }
```

```
                    }
               }
               if (count == 0)
                    cout << "Не знайдено спортсменів, які тренуються на
даному корті у тренера" << endl;
               cout << endl;
               count = 0;
               break;
          case 3:
               cout << "Спортсмени, які тренуються на стадіонах:" << endl;
               for (int i = 0; i < stadion_count; i++)
               {
                    for (int j = 0; j < stadion[i].sponsor_count; j++)
                    {
                         for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                         {
                              for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                   if
(stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count > 1)
                                   {
                                        cout << "- " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sportsman_name <<
endl;
                                        cout << "Види спорта: " << endl;
                                        for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                             cout << a + 1 << "; " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name
<< endl;
                                        cout << endl;
                                        count++;
                                   }
                              }
                         }
                    }
               }
               if (count == 0)
                    cout << "Не знайдено спортсменів, які займаються більше
ніж одним видом спорту на стадіоні" << endl << endl;
               count = 0;

               cout << "Спортсмени, які тренуються у спортзалах:" << endl;
               for (int i = 0; i < gym_count; i++)
               {
                    for (int j = 0; j < gym[i].sponsor_count; j++)
                    {
                         for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                         {
```

```
                              for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    if
(gym[i].sponsor[j].competitions[m].sportsman[n].sport_count > 1)
                                    {
                                          cout << "- " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                          cout << "Види спорту: " << endl;
                                          for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                                cout << a + 1 << "; " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name <<
endl;
                                          cout << endl;
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які займаються більше
ніж одним видом спорту в спортзалах" << endl << endl;
            count = 0;

            cout << "Спортсмени, які тренуються на кортах:" << endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    if
(court[i].sponsor[j].competitions[m].sportsman[n].sport_count > 1)
                                    {
                                          cout << "- " <<
court[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                          cout << "Види спорту: " << endl;
                                          for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                                cout << a + 1 << "; " <<
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name <<
endl;
                                          cout << endl;
                                          count++;
                                    }
```

```
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено спортсменів, які займаються більше
ніж одним видом спорту на кортах" << endl << endl;
            count = 0;
            break;
      default:
            cout << "Помилка! Спробуйте ще раз." << endl << endl;
            break;
      }

      Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}

void Functions::Trainers(Stadion* stadion, int stadion_count, Gym* gym,
int gym_count, Court* court, int court_count)
{
      cout << "Отримати список тренерів вказаного спортсмена: 1" <<
endl;
      cout << "Отримати список тренерів певного виду спорту:  2" << endl
<< endl;

      string sportsman_name, sport_name;
      int number, count = 0;
      cout << "Введіть номер: ";
      cin >> number;
      cout << endl;

      switch (number)
      {
      case 1:
            cout << "Введіть спортсмена: ";
            cin >> sportsman_name;
            cout << endl;

            cout << "Тренери, які тренерують спортсменів на стадіонах:"
<< endl;
            for (int i = 0; i < stadion_count; i++)
            {
                  for (int j = 0; j < stadion[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
```

```
                                    if (sportsman_name ==
stadion[i].sponsor[j].competitions[m].sportsman[n].sportsman_name)
                                    {
                                          for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                          {
                                                cout << "Вид спорту: " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name
<< endl;
                                                for (int b = 0; b <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_coun
t; b++)
                                                      cout << b + 1 << ". "
<<
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].t
rainer_name << endl;
                                                cout << endl;
                                          }
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено тренера, який проводять тренування
в даного спортсмена на стадіонах" << endl << endl;
            count = 0;

            cout << "Тренера, які тренують спортсменів в спортзалах:" <<
endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    if (sportsman_name ==
gym[i].sponsor[j].competitions[m].sportsman[n].sportsman_name)
                                    {
                                          for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                          {
                                                cout << "Вид спорту: " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name <<
endl;
```

```
                                                for (int b = 0; b <
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                                      cout << b + 1 << ". "
<<
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].train
er_name << endl;
                                                cout << endl;
                                          }
                                          count++;
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено тренера, який проводять тренування
в даного спортсмена в спортзалах" << endl << endl;
            count = 0;

            cout << "Тренера, які тренують спортсменів на кортах:" <<
endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    if (sportsman_name ==
court[i].sponsor[j].competitions[m].sportsman[n].sportsman_name)
                                    {
                                          for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                          {
                                                cout << "Вид спорту: " <<
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name <<
endl;
                                                for (int b = 0; b <
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                                      cout << b + 1 << ". "
<<
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].tra
iner_name << endl;
                                                cout << endl;
                                          }
                                          count++;
```

```
                                    }
                               }
                          }
                     }
               }
               if (count == 0)
                    cout << "Не знайдено тренера, який проводять тренування
в даного спортсмена  на кортах" << endl << endl;
               count = 0;
               break;

          case 2:
               cout << "Введіть вид спорту: ";
               cin >> sport_name;
               cout << endl;

               cout << "Тренера, які проводять заняття на стадіонах:" <<
endl;
               for (int i = 0; i < stadion_count; i++)
               {
                    for (int j = 0; j < stadion[i].sponsor_count; j++)
                    {
                         for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                         {
                              for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                   for (int a = 0; a <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                   {
                                        if (sport_name ==
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
                                        {
                                             for (int b = 0; b <
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_coun
t; b++)
                                                  cout << "- " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].t
rainer_name << endl;
                                             count++;
                                        }
                                   }
                              }
                         }
                    }
               }
               if (count == 0)
                    cout << "Не знайдено тренера, який відповідає за даний
вид спорту на стадіонах" << endl;
               cout << endl;
```

```
            count = 0;

            cout << "Тренера, які тренують спортсменів в спортзалах" <<
endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
                                    for (int a = 0; a <
gym[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          if (sport_name ==
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
                                          {
                                                for (int b = 0; b <
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                                      cout << "- " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].train
er_name << endl;
                                                count++;
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено тренера, який відповідає за даний
вид спорту в спортзалах" << endl;
            cout << endl;
            count = 0;

            cout << "Тренера, які тренерують спортсменів на кортах:" <<
endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {
                              for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                              {
```

```
                                    for (int a = 0; a <
court[i].sponsor[j].competitions[m].sportsman[n].sport_count; a++)
                                    {
                                          if (sport_name ==
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].sport_name)
                                          {
                                                for (int b = 0; b <
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer_count;
b++)
                                                      cout << "- " <<
court[i].sponsor[j].competitions[m].sportsman[n].sport[a].trainer[b].tra
iner_name << endl;
                                                count++;
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Не знайдено тренера, який відповідає за даний
вид спорту на кортах" << endl;
            cout << endl;
            count = 0;
            break;
      default:
            cout << "Помилка! Спробуйте ще раз" << endl << endl;
            break;
      }

      Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}

void Functions::Achievement(Stadion* stadion, int stadion_count, Gym*
gym, int gym_count, Court* court, int court_count)
{
      cout << "Отримати список призерів даного змагання: 1" << endl;
      cout << "Отримати список організаторів змагань і число проведених
ними змагань протягом певного часу : 2" << endl << endl;

      string competition_name;
      int start_of_competitions, end_of_competitions, number, count = 0;
      cout << "Введіть номер: ";
      cin >> number;
      cout << endl;

      switch (number)
      {
      case 1:
            cout << "Введіть назву змагання: ";
```

```
            cin >> competition_name;
            cout << endl;

            cout << "Тренування на стадіонах:" << endl;
            for (int i = 0; i < stadion_count; i++)
            {
                  for (int j = 0; j < stadion[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
                        {
                              if (competition_name ==
stadion[i].sponsor[j].competitions[m].competitions_name)
                              {
                                    for (int n = 0; n <
stadion[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                                    {
                                          if
(stadion[i].sponsor[j].competitions[m].sportsman[n].place < 4)
                                          {
                                                cout << "- " <<
stadion[i].sponsor[j].competitions[m].sportsman[n].sportsman_name <<
endl;
                                                count++;
                                          }
                                    }
                              }
                        }
                  }
            }
            if (count == 0)
                  cout << "Даного змагання, проведеного на стадіонах, не
існує, або призери відсутні" << endl;
            cout << endl;
            count = 0;

            cout << "Тренування в спортзалах:" << endl;
            for (int i = 0; i < gym_count; i++)
            {
                  for (int j = 0; j < gym[i].sponsor_count; j++)
                  {
                        for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
                        {
                              if (competition_name ==
gym[i].sponsor[j].competitions[m].competitions_name)
                              {
                                    for (int n = 0; n <
gym[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                                    {
```

```
                                        if
(gym[i].sponsor[j].competitions[m].sportsman[n].place < 4)
                                        {
                                              cout << "- " <<
gym[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                              count++;
                                        }
                                  }
                            }
                      }
                }
          }
          if (count == 0)
                cout << "Даного змагання, проведеного в спортзалах, не
існує, або призери відсутні" << endl;
          cout << endl;
          count = 0;

          cout << "Тренування на кортах для тенісу:" << endl;
          for (int i = 0; i < court_count; i++)
          {
                for (int j = 0; j < court[i].sponsor_count; j++)
                {
                      for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                      {
                            if (competition_name ==
court[i].sponsor[j].competitions[m].competitions_name)
                            {
                                  for (int n = 0; n <
court[i].sponsor[j].competitions[m].sportsman_quantity; n++)
                                  {
                                        if
(court[i].sponsor[j].competitions[m].sportsman[n].place < 4)
                                        {
                                              cout << "- " <<
court[i].sponsor[j].competitions[m].sportsman[n].sportsman_name << endl;
                                              count++;
                                        }
                                  }
                            }
                      }
                }
          }
          if (count == 0)
                cout << "Даного змагання, проведеного на кортах, не
існує, або призери відсутні" << endl;
          cout << endl;
          count = 0;
          break;
```

```
	case 2:
		cout << "Дата початку змагання: ";
		cin >> start_of_competitions;
		cout << endl;

		cout << "Дата закінчення змагання: ";
		cin >> end_of_competitions;
		cout << endl;

		cout << "Спонсори стадіонів:" << endl;
		for (int i = 0; i < stadion_count; i++)
		{
			for (int j = 0; j < stadion[i].sponsor_count; j++)
			{
				cout << "Спонсор: " <<
stadion[i].sponsor[j].sponsor_name << endl;
				for (int m = 0; m <
stadion[i].sponsor[j].competitions_quantity; m++)
				{
					if
(stadion[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
					{
						if
(stadion[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
						{
							cout << "- " <<
stadion[i].sponsor[j].competitions[m].competitions_name << endl;
							count++;
						}
					}
				}
				if (count == 0)
					cout << "У даного спонсора відсутні змагання,
що проводяться в даний період часу" << endl;
				cout << endl;
			}
		}
		count = 0;

		cout << "Спонсори спортзалів:" << endl;
		for (int i = 0; i < gym_count; i++)
		{
			for (int j = 0; j < gym[i].sponsor_count; j++)
			{
				cout << "Спонсор: " <<
gym[i].sponsor[j].sponsor_name << endl;
				for (int m = 0; m <
gym[i].sponsor[j].competitions_quantity; m++)
				{
```

```
                              if
(gym[i].sponsor[j].competitions[m].date_start >= start_of_competitions)
                              {
                                    if
(gym[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "- " <<
gym[i].sponsor[j].competitions[m].competitions_name << endl;
                                          count++;
                                    }
                              }
                        }
                        if (count == 0)
                              cout << "У даного спонсора відсутні змагання,
що проводяться в даний період часу" << endl;
                        cout << endl;
                  }
            }
            count = 0;

            cout << "Спонсори кортів для тенісу:" << endl;
            for (int i = 0; i < court_count; i++)
            {
                  for (int j = 0; j < court[i].sponsor_count; j++)
                  {
                        cout << "Спонсор: " <<
court[i].sponsor[j].sponsor_name << endl;
                        for (int m = 0; m <
court[i].sponsor[j].competitions_quantity; m++)
                        {


                              if
(court[i].sponsor[j].competitions[m].date_start >=
start_of_competitions)
                              {
                                    if
(court[i].sponsor[j].competitions[m].date_end <= end_of_competitions)
                                    {
                                          cout << "- " <<
court[i].sponsor[j].competitions[m].competitions_name << endl;
                                          count++;
                                    }
                              }
                        }
                        if (count == 0)
                              cout << "У даного спонсора відсутні змагання,
що проводяться в даний період часу" << endl;
                        cout << endl;
                  }
            }
```

```
            count = 0;
            break;
      default:
            cout << "Помилка! Спробуйте ще раз" << endl << endl;
            break;
      }

      Functions::Menu(stadion, stadion_count, gym, gym_count, court,
court_count);
}
```

Додаток Б

Результати контрольного прикладу

Рисунок Б.1.

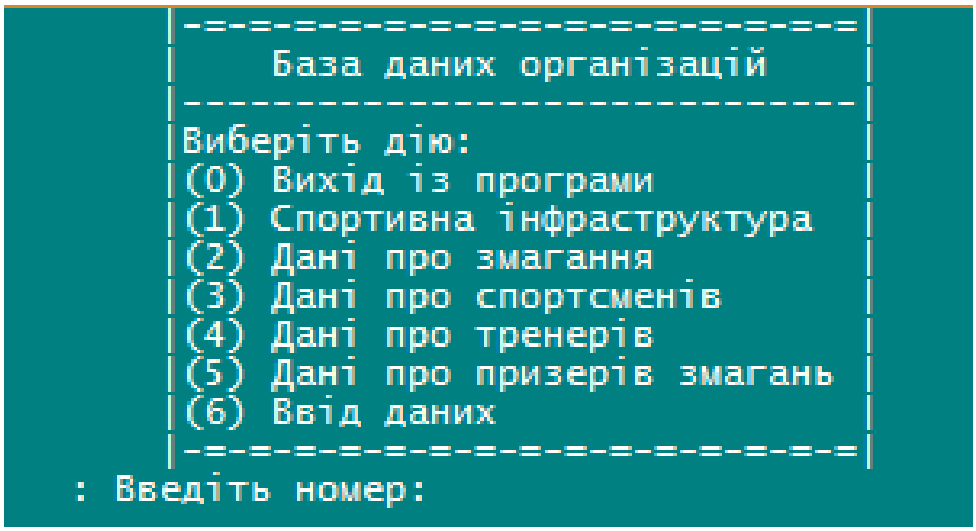


Рисунок Б.2.

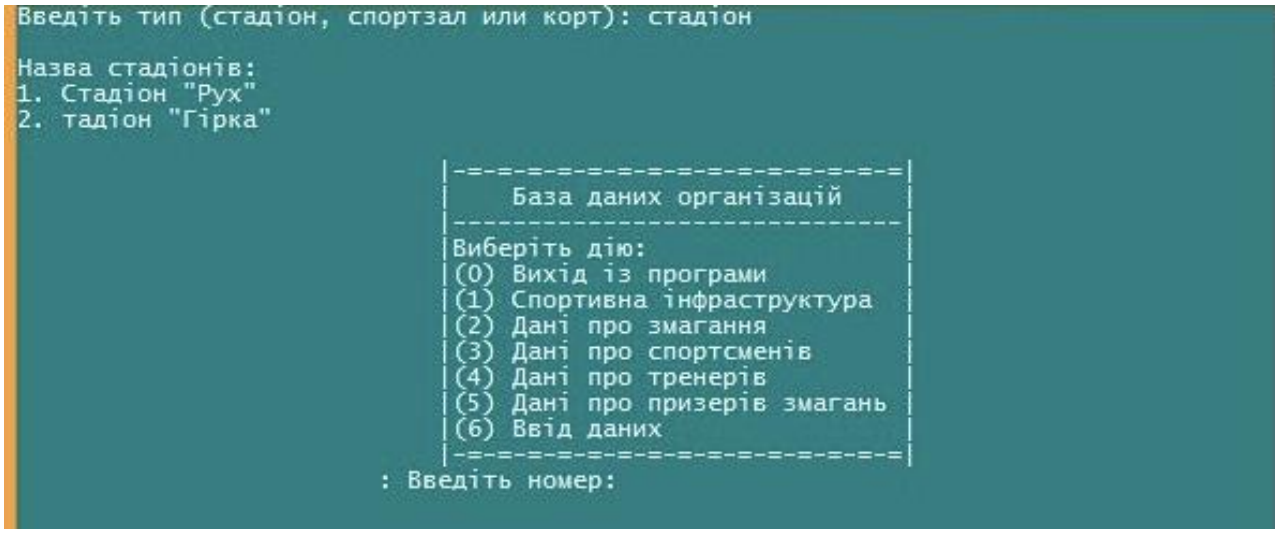


Рисунок Б.3.

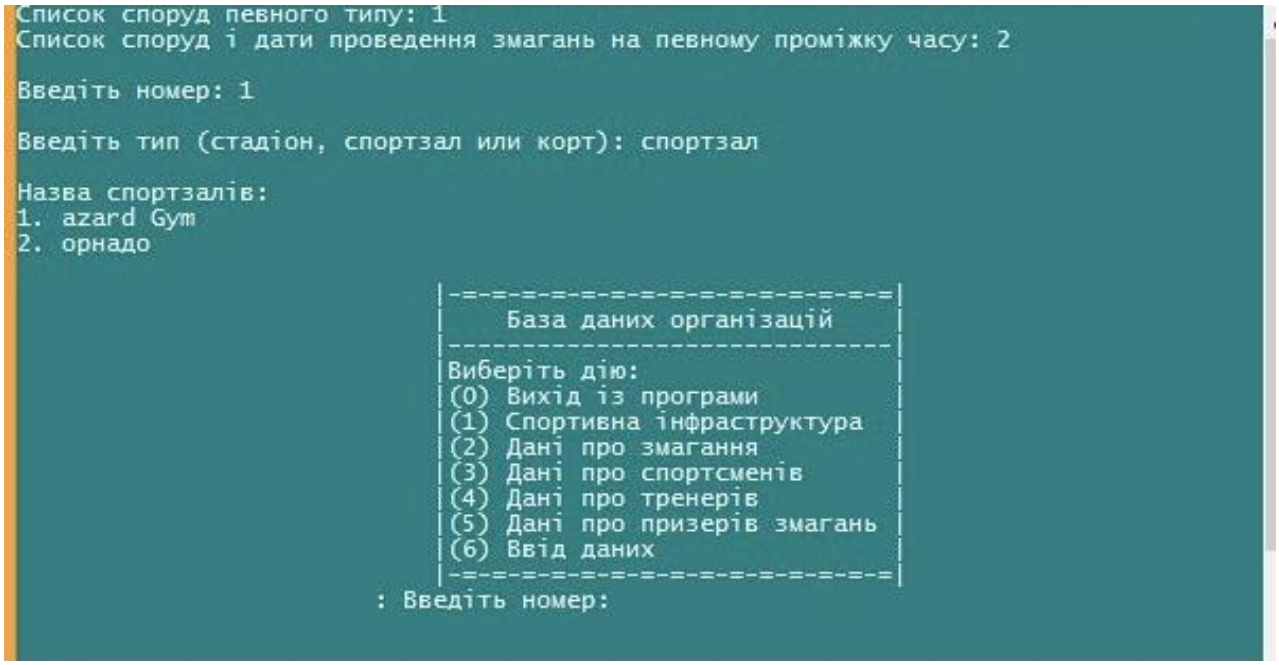


Рисунок Б.4.

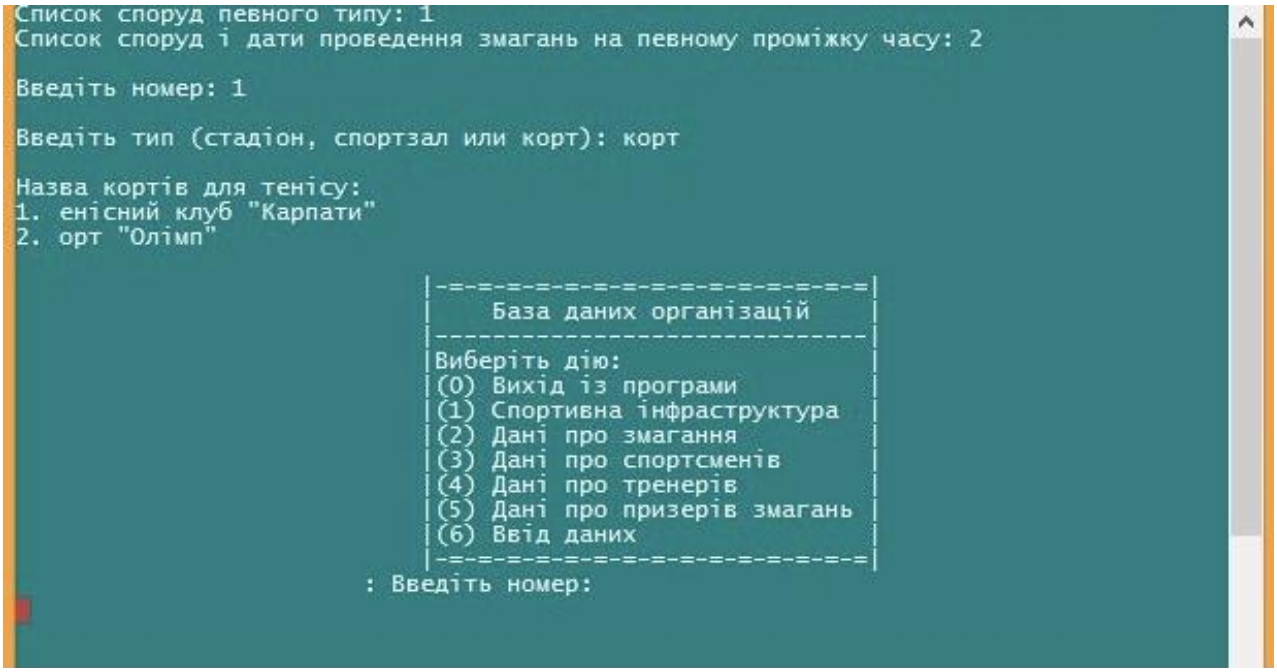


Рисунок Б.5.

```
Список споруд певного типу: 1
Список споруд і дати проведення змагань на певному проміжку часу: 2

Введіть номер: 2

Введіть дату початку змагання: 2016

Введіть дату закінчення змагання: 2020

Змагання, проведені на стадіоні:
Стадіон № 1: Стадіон "Рух"
Назва змагання: НФК УРАГАН_ПРИКАРПАТТЯ
Дата початку змагання: 2019
Дата закінчення змагання: 2019

Стадіон № 2: тадіон "Гірка"
Назва змагання: Біг призерів О.І.
Дата початку змагання: 2020
Дата закінчення змагання: 2020

Змагання у спортзалах:
Спортзал № 2: орнадо
Назва змагання: Бодібілдинг в Торнадо
Дата початку змагання: 2019
Дата закінчення змагання: 2019

Змагання на кортах:
Корт № 1: енісний клуб "Карпати"
Назва змагання: Змагання серед молоді теніс
Дата початку змагання: 2019
Дата закінчення змагання: 2019
```

Рисунок Б.6.

```
Отримати список змагань, проведених на певному проміжку часу: 1
Отримати список змагань, проведених в певному спортивному комплексі: 2

Введіть номер: 1

Введіть дату початку змагання: 2016

Введіть рік закінчення змагання: 2021

Змагання на стадіонах:
- НФК УРАГАН_ПРИКАРПАТТЯ
- Біг призерів О.І.

Змагання в спортзалах:
- Єдиноборства в Hazard
- Бодібілдинг в Торнадо

Змагання на кортах для тенісу:
- Змагання серед молоді теніс
- Зустріч чемпіонів

                            |-=-=-=-=-=-=-=-=-=-=-=-=-=-=-|
                            |    База даних організацій   |
                            |-----------------------------|
                            |Виберіть дію:                |
                            |(0) Вихід із програми        |
                            |(1) Спортивна інфраструктура |
                            |(2) Дані про змагання        |
                            |(3) Дані про спортсменів     |
                            |(4) Дані про тренерів        |
                            |(5) Дані про призерів змагань |
                            |(6) Ввід даних               |
                            |-=-=-=-=-=-=-=-=-=-=-=-=-=-=-|
                        : Введіть номер:
```

Рисунок Б.7.

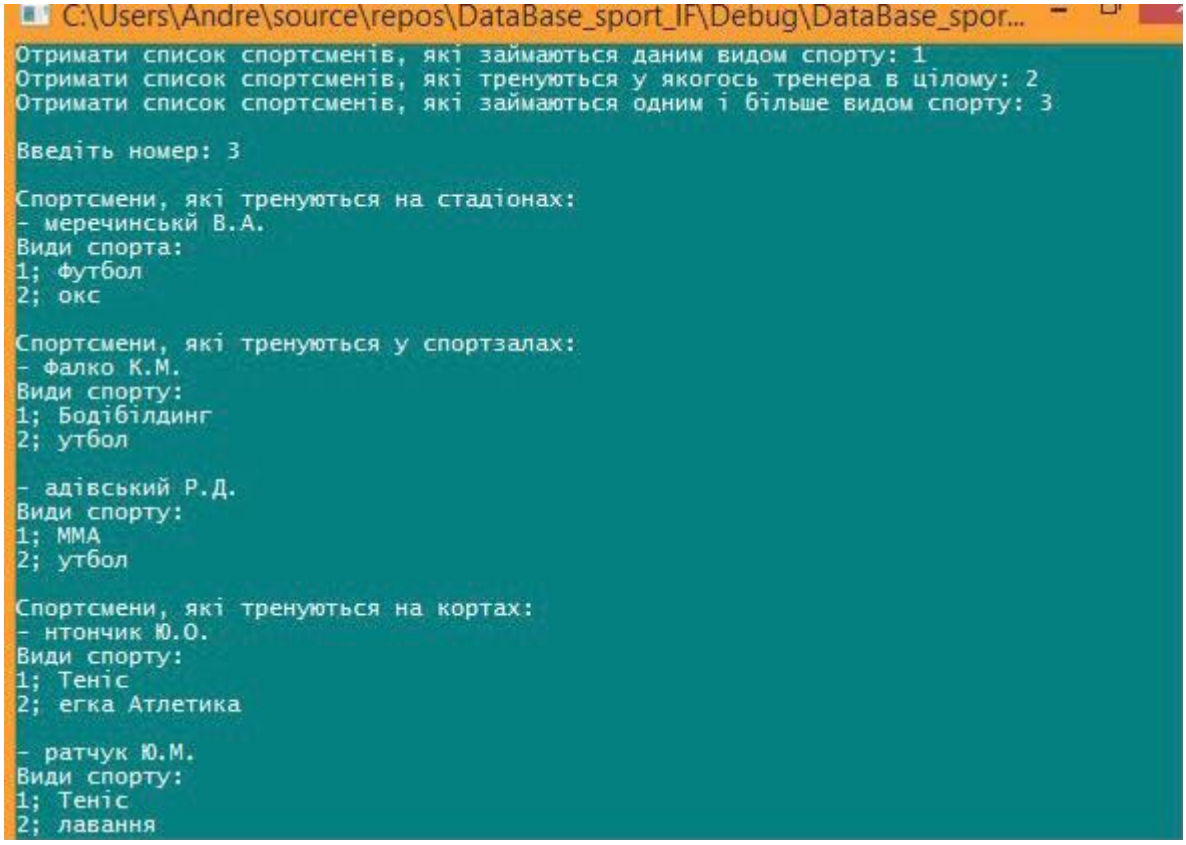

C:\Users\Andre\source\repos\DataBase_sport_IF\Debug\DataBase_spor...
Отримати список спортсменів, які займаються даним видом спорту: 1
Отримати список спортсменів, які тренуються у якогось тренера в цілому: 2
Отримати список спортсменів, які займаються одним і більше видом спорту: 3
Введіть номер: 3
Спортсмени, які тренуються на стадіонах:
- меречинський В.А.
Види спорта:
1; футбол
2; окс
Спортсмени, які тренуються у спортзалах:
- фалко К.М.
Види спорту:
1; Бодібілдинг
2; утбол
- адівський Р.Д.
Види спорту:
1; ММА
2; утбол
Спортсмени, які тренуються на кортах:
- нтончик Ю.О.
Види спорту:
1; Теніс
2; егка Атлетика
- ратчук Ю.М.
Види спорту:
1; Теніс
2; лавання

Рисунок Б.8.

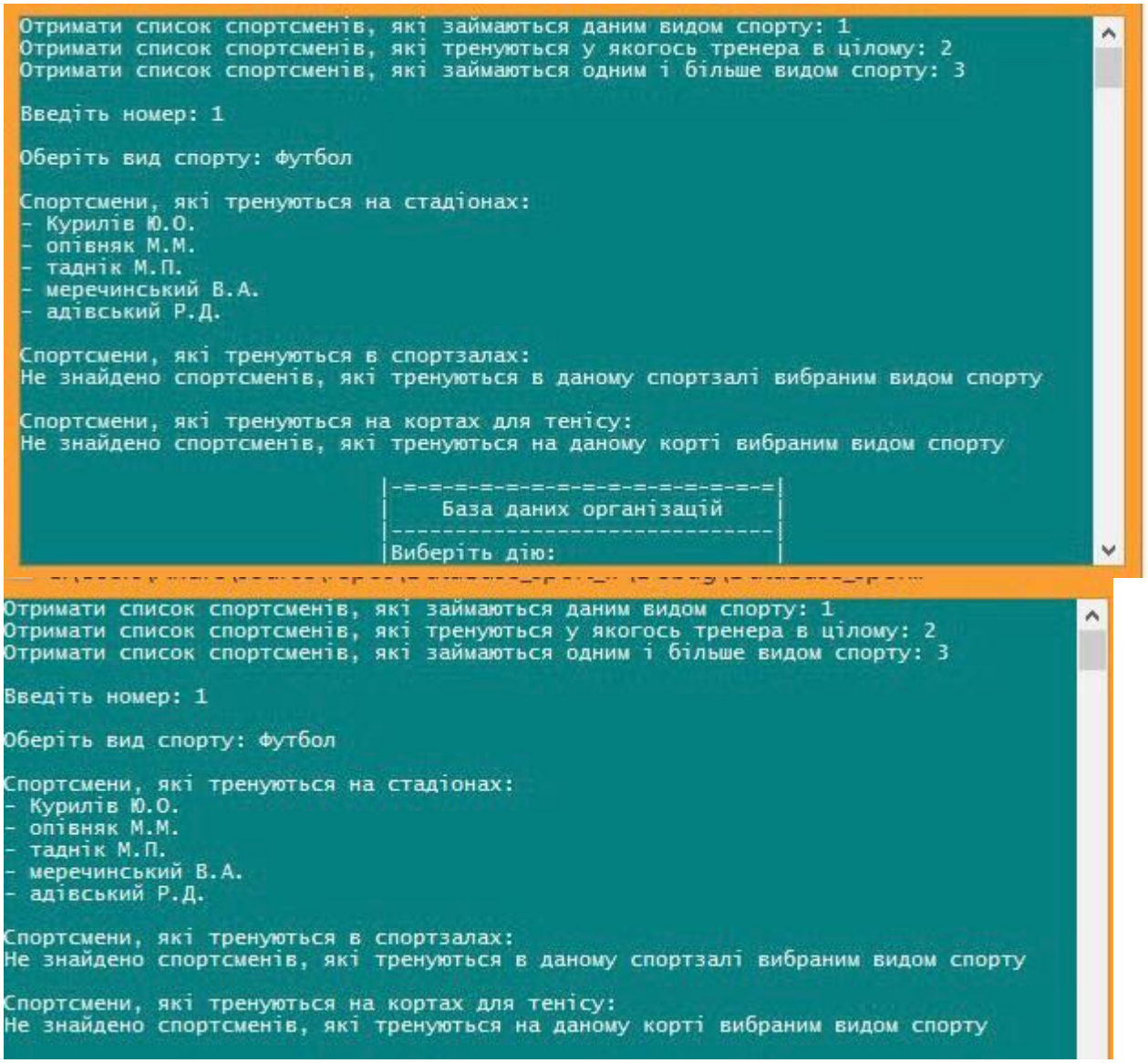


Рисунок Б.9.