МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ЛІСОТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

Інститут комп’ютерних наук та інформаційних технологій

Кафедра комп’ютерних наук

КУРСОВА РОБОТА

З дисципліни «Об’єктно-орієнтоване програмування»

Виконав

Студент групи КН-21

Гайванович О. А.

Керівник роботи

Сало М. Ф.

м. Львів – 2024

1. ЗАВДАННЯ НА КУРСОВУ РОБОТУ

студенту 2-го курсу групи КН-21
Гайвановичу Остапу
Варіант 9

1. Створити клас ВЕРШИНА з координатами точки на площині.
2. Створити базовий клас ФІГУРА, який містить масив об’єктів класу ВЕРШИНА. Визначити чисто віртуальний метод переміщення фігури на площині.
3. Для обох класів визначити конструктори ініціалізації, переміщення, копіювання, деструктори та методи для зміни і читання значень полів даного класу.
4. Перевантажити операції для класу ВЕРШИНА: + для додавання координат, - для віднімання, операцію * для множення на коефіцієнт, операцію / для ділення на коефіцієнт, операцію присвоєння об'єктів = та переміщення, потокові операції введення » та виведення « об'єктів.
5. Створити похідні класи: ПРЯМОКУТНИК та КВАДРАТ. Визначити функції переміщення фігури у межах площини. Визначити конструктори та деструктори, методи або операторні функції введення-виведення.
6. У межах ієрархії класів побудувати поліморфічний кластер на основі віртуального методу переміщення фігури. Продемонструвати механізм пізнього зв'язування.
7. Розробити клас-контейнер, що містить масив об'єктів класів ПРЯМОКУТНИК та КВАДРАТ. Знайти фігуру з найбільшою площею.
8. Для роботи з масивом об'єктів побудувати та використати клас-ітератор.
9. Реалізувати пункти 1-8 завдання мовою C++.
10. Реалізувати пункти 1-8 завдання відповідними засобами мови C# з використанням WinForms.
11. Вимоги до оформлення курсової роботи дивитись у методичних вказівках до виконання курсового проектування з ООП (диск Р)
12. Забезпечити читання і запис даних для формування масиву об’єктів згідно завдання із файлу.

Вимоги:

1. Вхідні дані до роботи: літературні джерела, технічна документація щодо розробки програм, ДСТУ з оформлення документації.
2. Зміст пояснювальної записки: Вступ. Специфікація роботи. Програмна документація. Висновки. Додатки.

3. Перелік графічного матеріалу: діаграма класів додатку; схема алгоритму реалізації одного з розроблених методів (за узгодженням з керівником курсової роботи).

Дата здачі студентом завершеної роботи 10.05.2024 р.

Календарний план виконання курсової роботи

№ з/п	Назва етапу роботи	Строк виконання	Відмітка про виконання
1	З’ясування загальної постановки завдання; розробка чернетки першого розділу пояснювальної записки.		
2	Програмна реалізація, налагодження та тестування додатка; розробка чернетки пояснювальної записки (другий розділ, висновки, список використаних джерел, додатки).		
3	Остаточне налагодження та тестування додатка; розробка чернетки пояснювальної записки (вступ, другий розділ, висновки, список використаних джерел, додатки).		
4	Розробка остаточного варіанта пояснювальної записки (вступ, другий розділ, висновки, список використаних джерел, додатки).		

Дата видачі завдання: 2.11.2023 р.

Керівник роботи: Сало М. Ф. _____
(підпис)

Студент: Гайванович О. А. _____
(підпис)

ЗМІСТ

2. РЕФЕРАТ

Пояснювальна записка: 45 стор., 8 рис., 2 табл., 3 додатки, 15 джерел.

ОБ'ЄКТ РОЗРОБЛЕННЯ: ієрархія класів для представлення геометричних фігур на площині та демонстрації концепцій об'єктно-орієнтованого програмування.

МЕТА РОБОТИ: реалізувати систему класів для опису вершин, фігур, їх властивостей та операцій, а також продемонструвати використання принципів ООП, таких як успадкування, поліморфізм, перевантаження операторів, використання контейнерів та ітераторів.

МЕТОДИ ТА ЗАСОБИ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ: об'єктно-орієнтоване програмування з використанням C++ та C#, середовища розробки Microsoft Visual Studio, бібліотек STL та Nlohmann-JSON (для C++), фреймворку Microsoft .NET, графічного API Windows Forms (для C#).

РЕЗУЛЬТАТИ ТА РЕКОМЕНДАЦІЇ: розроблено ієрархію класів для представлення точок, фігур (прямокутників, квадратів) та операцій над ними. Реалізовано перевантаження операторів, використання контейнера та ітератора, візуалізацію за допомогою Windows Forms. Програма демонструє переваги ООП: інкапсуляцію, поліморфізм, модульність, повторне використання коду.

ЗНАЧУЩІСТЬ РОБОТИ: набуття практичних навичок об'єктно-орієнтованого програмування, реалізації принципів ООП, використання бібліотек та середовищ розробки. Отримані результати можуть бути використані для вивчення ООП та розширені для створення більш складних систем.

КЛЮЧОВІ СЛОВА: об'єктно-орієнтоване програмування, ієрархія класів, фігури, перевантаження операторів, віртуальні методи, контейнери, ітератори, c++, c#, WinForms.

ВСТУП

Об'єктно-орієнтоване програмування (ООП) — це парадигма програмування, яка базується на концепції об'єктів, що містять дані (властивості) та коди (методи). ООП забезпечує кращу модульність програм та повторне використання коду шляхом пакування даних та функціональності в об'єкти.

Основними принципами ООП є:

1. **Інкапсуляція**: об'єднання даних та методів, що оперують цими даними, в єдиний об'єкт, забезпечуючи приховування реалізації від зовнішнього світу.
2. **Успадкування**: можливість створення нових класів на основі існуючих, успадковуючи їхні властивості та методи.
3. **Поліморфізм**: здатність об'єктів різних класів обробляти той самий метод різними способами.
4. **Абстракція**: виділення суттєвих характеристик об'єкта та відкидання несуттєвих деталей.

Однією з ключових переваг ООП є підвищена модульність, повторне використання коду, спрощена розробка та супровід великих проєктів. Ці переваги роблять ООП широко застосовуваним у розробці програмного забезпечення, особливо у великих та складних системах.

C++ є однією з найпопулярніших та широко використовуваних мов програмування у світі. Вона була розроблений у 1980-х роках Б'ярном Страуструпом як розширення мови C, додаючи підтримку об'єктно-орієнтованого програмування, що стало його ключовою особливістю.

C++ повністю підтримує концепції ООП, надаючи можливість створювати класи, успадковувати їх, використовувати поліморфізм через механізм віртуальних функцій, а також реалізовувати інкапсуляцію та абстракцію. Крім того, C++ дозволяє поєднувати об'єктно-орієнтоване та процедурне програмування в одному проєкті.

C++ також має багато інших особливостей, таких як обробка винятків, перевантаження операторів, шаблони, багатопотоковість тощо, що роблять його потужним та гнучким інструментом для розробки різноманітних додатків, включаючи системне програмування, ігри, мультимедіа та багато іншого.

Мова програмування C# була розроблена в 1998—2001рр. компанією Microsoft як частина її загальної .NET-стратегії. Головним архітектором C# був Андерс Хейлсберг, один з провідних фахівців у галузі мов програмування, який здобув світове визнання завдяки створенню успішного продукту Turbo Pascal у 1980-х роках.

C# безпосередньо пов'язана з C++ та Java, і це не випадково: адже ці три мови є найпопулярнішими та найулюбленішими мовами програмування у світі. Велика кількість професійних програмістів знають C, C++ та Java. Оскільки C# побудований на міцному, зрозумілому фундаменті цих «фундаментальних» мов, перехід до нього відбувається без особливих труднощів.

Ключові особливості C#:

- Уніфікована система типів, де все розглядається як об'єкт (класи, структури, масиви, вбудовані типи).
- Використання просторів імен замість включення файлів заголовків.
- Простота синтаксису: відсутність операторів "->" та "::", використання лише оператора ".".
- Сучасність: можливість використання псевдонімів для просторів імен та класів, атрибутів для полегшення відладки коду.
- Можливість прикріплення типізованих, розширюваних метаданих до об'єктів для зв'язку бізнес-логіки з кодом.

Порівняно з іншими мовами:

- Синтаксис C# успадкований від C++ та Java, тому для розробників, що мають досвід з цими мовами, він буде знайомим.
- На відміну від C++, C# вводить такі новації, як атрибути, делегати та події, які дозволяють застосовувати принципово нові прийоми програмування.
- Порівняно з Java, C# має підтримку просторів імен, властивостей, індексаторів та перечислень контейнерів.
- Унікальними рисами C# є події, індексатори, атрибути та делегати.

3. ФОРМУЛЮВАННЯ ЗАДАЧІ

Метою даної роботи є реалізація ієрархії класів для представлення геометричних фігур на площині та демонстрація різних концепцій об'єктно-орієнтованого програмування, таких як успадкування, поліморфізм, перевантаження операторів, використання контейнерів та ітераторів.

Вхідні дані:

- Координати по осях X та Y для створення об’єктів класу Вершина
- Координати нижньої лівої вершини та сторони (для Квадрату) чи двох сторін (для Прямокутника) для створення об'єктів класів Прямокутник, Квадрат
- Параметри для переміщення фігур (наприклад, зміщення по осях X та Y)

Вихідні дані:

- Відображення координат вершин створених фігур
- Результати операцій над фігурами (додавання/віднімання координат, множення/ділення на коефіцієнт, переміщення фігури)
- Візуалізація фігур на площині
- Фігура з найбільшою площею серед створених

Вимоги до інструментальних засобів:

- Середовище розробки для C++
- Середовище розробки для C# з підтримкою фреймворку Microsoft .NET та графічного API Windows Forms

Наведені вимоги визначають необхідну функціональність, вхідні/вихідні дані, форми їх представлення, а також інструментальні засоби, потрібні для успішної реалізації поставленого завдання курсової роботи.

4. МЕТОДИ ТА ЗАСОБИ РОЗВ’ЯЗУВАННЯ ЗАДАЧІ

Для реалізації поставленого завдання було використано об'єктно-орієнтований підхід до програмування. Він передбачає розбиття системи на об'єкти, які інкапсулюють дані та функціональність, і взаємодію між цими об'єктами для виконання певних операцій. Зокрема, були застосовані такі основні концепції ООП:

1. Абстракція та інкапсуляція для визначення класів (Вершина, Фігура, Прямокутник, Квадрат) з відповідними даними та методами.
2. Успадкування для створення ієрархії класів, де похідні класи успадковують властивості та поведінку базових класів.
3. Поліморфізм для реалізації віртуальних методів та пізнього зв'язування при роботі з об'єктами через базовий клас.
4. Перевантаження операторів для покращення читабельності коду та підвищення зручності роботи з об'єктами класу Вершина.
5. Шаблони для створення класів-контейнерів та ітераторів, що працюють з різними типами об'єктів.

Засоби розробки:

1. Середовище розробки: Microsoft Visual Studio зі встановленими відповідними компонентами для роботи з C++ з-під консолі та C# з .NET Windows Forms
2. Компілятори C++ (з підтримкою стандарту C++20) та C#
3. Бібліотека стандартних шаблонів STL C++
4. Для C++ — стороння бібліотека JSON автора Nlohmann — для серіалізації / десеріалізації фігур
5. Засоби для відладки та тестування програм

Вибір цих методів та засобів обумовлений їх широкою підтримкою та поширеністю у сфері об'єктно-орієнтованого програмування. Застосування ООП дозволяє створювати модульні, масштабовані та легкі для підтримки системи. Використання сучасних IDE, як-от Visual Studio, полегшує процес написання, компіляції та налагодження коду. Бібліотеки стандартних контейнерів та графічні бібліотеки забезпечують необхідну функціональність без потреби у створенні всіх компонентів «з нуля». А формат JSON дуже поширений для серіалізації об’єктів і легко читається людиною.

Переваги використаних підходів: висока гнучкість, модульність та можливість повторного використання коду. Крім того, ООП відповідає сучасним практикам розробки і широко підтримується на різних платформах.

5. ПРОЕКТУВАННЯ СТРУКТУРИ ПРОГРАМИ

Структура програми була спроектована з використанням концепцій ООП, таких як інкапсуляція, успадкування та поліморфізм. Структура класів була спроектована відповідно до вимог завдання та принципів ООП.

Застосування об'єктно-орієнтованого підходу дозволило створити модульну та легко розширювану структуру програми, з можливістю додавання нових типів фігур шляхом успадкування від базового класу. Інкапсуляція забезпечила приховування внутрішньої реалізації об'єктів від зовнішнього коду, а поліморфізм — можливість роботи з об'єктами різних класів через базовий клас.

Детальну структуру класів, їх відношення успадкування, атрибути та методи можна проілюструвати за допомогою діаграм UML класів та послідовностей (див. додатки).

5.1. Структура програми на C++

Клас *Vertex* (Вершина) представляє собою вершину з координатами точки на площині, по осях X та Y. Є параметризований конструктор (за замовчування значення координат рівні 0), конструктори копіювання та переміщення. Є гетери та сетери координат, перевантажені оператори присвоювання, порівняння, потокового введення та виведення, а також базові арифметичні оператори та оператори. Для зручності визначення типу даних координат було прийнято рішення ввести псевдонім типу *_Vertex_t*, який використовуватиметься у всій програмі; за замовчуванням цьому псевдоніму присвоєний тип *float*.

Клас *Shape* (Фігура) — абстрактний клас, що представляє геометричну фігуру з масивом вершин та віртуальними методами. Присутні конструктори ініціалізації масиву вершин трьома шляхами. Реалізований метод перевірки введених вершин на факт послідовного розташування за годинниковою стрілкою. Реалізований віртуальний метод переміщення фігури, оголошено кілька чистих віртуальних методів.

Класи *Square* та *Rectangle* — похідні від *Shape*. Містяться конструктори створення фігур за верхньою лівою вершиною та стороною чи сторонами (для квадрату чи прямокутника відповідно). Є можливість створити фігуру з усіх вершин — тоді відбуватиметься валідація тих вершин. Реалізовані усі чисті віртуальні методи батьківського класу. Серед методів є метод для обчислення площі фігури.

Простір імен *menuMethods* включає методи, що можуть використовуватись як дії меню. Окрім цього, тут також є внутрішні простіри імен, *cin_aix* та *os_aux*, де містяться допоміжні функції для роботи з потоками, що спрощують роботу при розробці нових дій пунктів меню.

Клас *Menu* описує консольне меню. Структуру меню потрібно визначити самостійно, використовуючи методи для дій з простору імен *menuMethods*, Є можливість створювати підменю або ж додати пункт меню (структура *MenuItem*).

Крім того, був реалізований шаблонний клас-контейнер *simpleVector* для зберігання фігур у контейнері для ітерації по об'єктах контейнера. Для назв методів використовується інтерфейс бібліотеки стандартних шаблонів STL.

В точці входу в програму, головному методі *main(),* створюється об’єкт класу *Menu* та його структура, з подальшим показом меню на консоль та його запуском. Також тут «ловляться» всі можливі виключення (якщо вони раніше не були опрацьовані, наприклад, в методах з простору імен *menuMethods*): в такому випадку виводиться повідомлення про таке виключення і робота програми завершується.

5.2. Структура програми на C#

Клас *Vertex* описує вершину. Реалізовує інтерфейс *IComparable<Vertex>*. Перевизначено кілька методів класу *Object*. Як і у програмі на C++, є параметризований конструктор зі значеннями за замовчуванням, конструктор копіювання, визначені оператори порівняння. Для координат замість полів використано властивості, яких немає в C++.

Абстрактний клас *Shape* представляє геометричну фігуру. Інтерфейс *ICloneable* має бути реалізований у похідних класах. Окрім властивостей імені та масиву вершин є ще одна, що визначає колір фігури на полотні. Перевизначено кілька методів класу *Object*. Присутній конструктори ініціалізації масиву вершин. Реалізований метод перевірки введених вершин на факт послідовного розташування за годинниковою стрілкою. Реалізований віртуальний метод переміщення фігури, оголошено кілька чистих віртуальних методів.

Класи *Square* та *Rectangle* — похідні від *Shape*. Містяться конструктори створення фігур за верхньою лівою вершиною та стороною чи сторонами (для квадрату чи прямокутника відповідно). Є можливість створити фігуру з усіх вершин — тоді відбуватиметься валідація тих вершин. Реалізовані усі чисті віртуальні методи батьківського класу та інтерфейси. Серед методів є метод для обчислення площі фігури.

Клас *SimpleDictionary* є узагальненим і являє собою просту реалізацію словника, де «під капотом ховається» масив з типом даних *KeyValuePair* (структура ключ-значення). Реалізовує інтерфейс *IDictionary*: містить методи для додавання, вилучення, очищення елементів, перевірки на існування елементу тощо. Також створений клас-енумератор *SimpleDictionaryEnumerator*, як альтернатива класу-ітератору в C++.

Також в проєкті містяться наступні класи форм Windows Forms:

- *fm_main*: основна форма програми, містить список фігур, перелік дій та полотно, що відображає фігури в обмеженій частині площини.
- *fm_info*: форма показу інформації про фігуру.
- *fm_new_shape*: форма створення нової фігури з усіма необхідними параметрами.
- *fm_move*: форма переміщення фігури, де можна задати нові координати.

Детальніше про Windows Forms, а також про інтерфейс користувача, описано в розділі «РОЗРОБЛЕННЯ ПРОГРАМИ ТА ЇЇ ОПИС».

6. ПРОЕКТУВАННЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА

7. РОЗРОБЛЕННЯ ГРАФІЧНИХ СХЕМ КЛАСІВ ТА АЛГОРИТМІВ З ВИКОРИСТАННЯМ UML

8. РОЗРОБЛЕННЯ ПРОГРАМИ ТА ЇЇ ОПИС

9. ІНСТРУКЦІЯ КОРИСТУВАЧА

10. КОНТРОЛЬНИЙ ПРИКЛАД ТА АНАЛІЗ РЕЗУЛЬТАТІВ КОМП'ЮТЕРНОЇ РЕАЛІЗАЦІЇ ПРОГРАМИ

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ