Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра програмного забезпечення



**Звіт**

Про виконання лабораторної роботи №7

**На тему:**

“Множинне наслідування. Принцип поліморфізму. Віртуальні функції та абстрактні класи”

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ-18

Юшкевич А.І.

**Прийняв:**

асис. каф. ПЗ

Далявський В.С.

« … » … 2023 р.

∑ = \_\_\_\_\_\_\_\_\_\_\_\_

Львів – 2023

**Тема**: Множинне наслідування. Принцип поліморфізму. Віртуальні функції та абстрактні класи.

**Мета**: Отримати практичні навички використання множинного наслідування, освоїти способи вирішення проблеми неоднозначності при множинному наслідуванні. Навчитись створювати абстрактні класи, колекції об’єктів базового типу, що включають об’єкти похідних типів. Навчитись використовувати чисті віртуальні функції.

Теоретичні відомості

**Множинне наслідування**

Множинне наслідування (multiple inheritance) - це можливість використовувати кілька базових класів для класів нащадків. Це означає, що клас може успадковувати властивості і методи з декількох батьківських класів одночасно.

Уявімо ситуацію, що ми створюємо програму, в якій ми працюємо з класами тварин. Один з цих класів Bird (птах), а другий Horse (кінь). Клас Bird містить функцію Fly(), а клас Horse містить функцію Gallop() – біг галопом.

А тепер сталось так, що нам потрібно створити новий міфічний персонаж – крилатого Пегаса (Pegasus), який буде гібридом між птахом та конем. В таких випадках використовують множинне наслідування

**Поліморфізм**

Поліморфізм – одна з трьох основних парадигм ООП. Якщо говорити коротко, поліморфізм – це здатність об'єкта використовувати методи похідного класу, про існування якого базовий клас не знає.

Поліморфізм - це властивість об'єктів різних типів відображати специфічну для них поведінку, при цьому використовується загальний інтерфейс, що дозволяє взаємодіяти з цими об'єктами у єдиному контексті. Це означає, що можна працювати із об'єктами різних типів використовуючи базовий тип, що дозволяє підвищувати гнучкість і повторне використання коду.

У контексті C++ поліморфізм досягається за допомогою віртуальних функцій і наслідування, що дозволяє об'єктам похідних класів перевизначати методи базового класу, які є віртуальними. Коли ці методи викликаються на об'єкті базового класу, викликається відповідний метод на об'єкті похідного класу, що дозволяє досягти поліморфізму.

Віртуальні функції повинні мати ключове слово virtual в оголошенні, яке означає, що функція є віртуальною. Клас, що містить віртуальну функцію, називається поліморфним класом. У похідних класах віртуальна функція може бути перевизначена. Починаючи з C++11 перевизначені функції варто позначати ключовим словом override. Це робить код більш зрозумілим та допомагає виявити помилки хибного перевизначення.

Завдання

1. Внести зміни в ієрархію класів із лабораторної роботи №6, зробивши класи поліморфними.
2. Базовий абстрактний клас має мати віртуальний деструктор та хоча б одне визначення для чисто віртуальної функції.
3. Створити масиви об’єктів базового типу, в них помістити об’єкти похідного типу. Продемонструвати виклик функцій з об’єктів – елементів масиву. Використати оператор dynamic\_cast.
4. Використати множинне наслідування (дозволяється додатково створити формальний абстрактний клас), продемонструвати вирішення проблеми з неоднозначністю доступу до членів базових класів за допомогою віртуального наслідування, за допомогою явного звертання до членів класу та за допомогою заміщення функцій в похідному класі.
5. Для роботи із динамічною пам’яттю використовувати std::unique\_ptr.
6. Продемонструвати роботу класів таким чином (на вибір):
   1. За допомогою тестів Google Test, які перевірятимуть роботу ієрархії класів.
   2. За допомогою інтерактивної програми із демонстрацією її роботи.
7. Сформувати звіт до лабораторної роботи. Відобразити в ньому діаграму наслідування класів.

Варіант №13:

|  |  |
| --- | --- |
| 13 | Клас CVehicle має містити чисто віртуальні функції:    virtual double GetDistancePerOneLiter(double dUsefulMassInKg) const = 0; virtual double GetServicePricePerThousandKm() const = 0;    Класи CCar, CTruck та CMotorcycle мають реалізувати ці функції відповідно до своїх правил.    Написати функцію, яка отримуватиме масив вказівників на базовий клас – CVehicle – та, викликаючи GetDistancePerOneLiter() та GetServicePricePerThousandKm(), обчислюватиме затрати на перевезення заданої маси на задану відстань. Функція має враховувати максимальну корисну масу, яку може перевозити транспортний засіб, і відповідно брати необхідну кількість траспортних засобів певного типу. |

# Код програми

# 

# Результат виконання

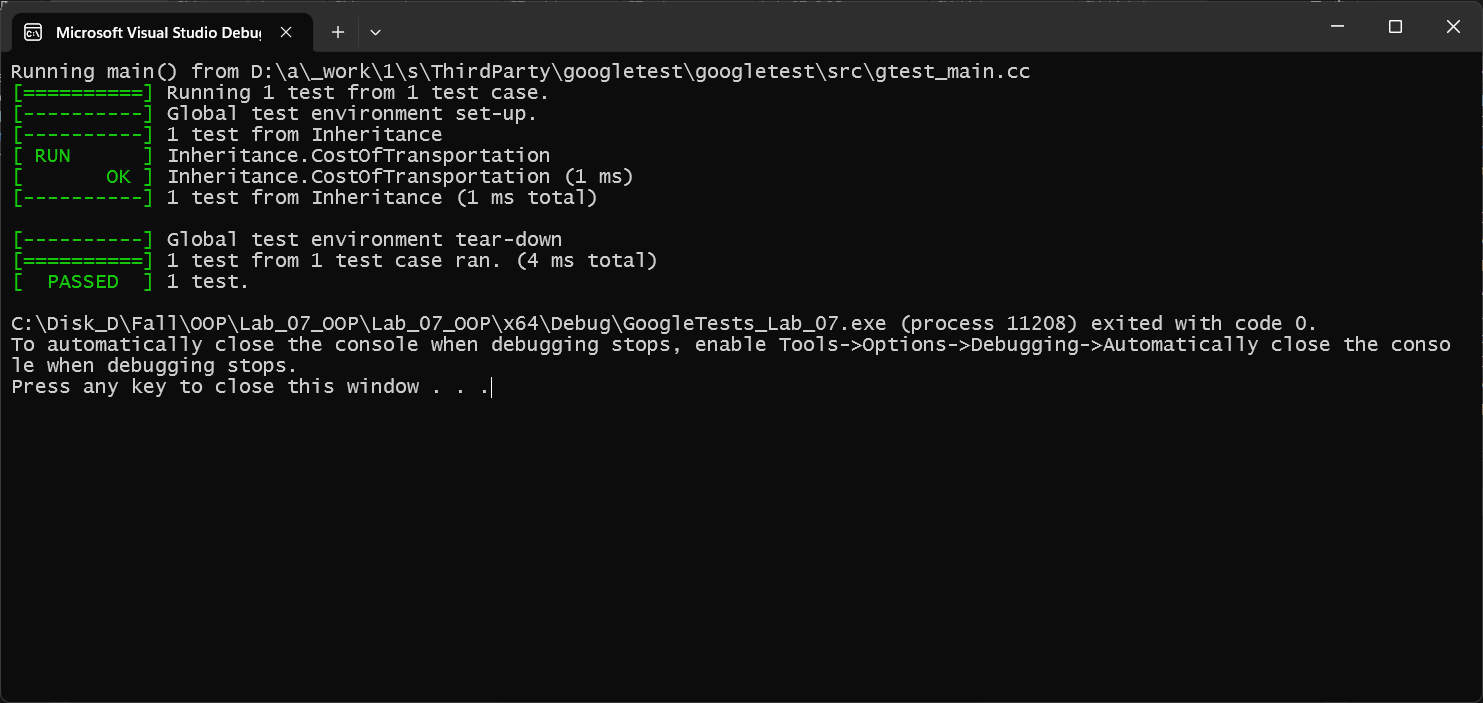


Рис. 1. Виконання тестів для функцій класів CVehicle, CCar, CTruck, CMotorcycle.

# Висновок

Під час виконання лабораторної роботи я отримав практичні навички використання множинного наслідування, освоїв способи вирішення проблеми неоднозначності при множинному наслідуванні. Навчився створювати абстрактні класи, колекції об’єктів базового типу, що включають об’єкти похідних типів. Навчився використовувати чисті віртуальні функції.