Лабораторна робота
№7

Тема: «Розробка програм з користувацькими класами. Робота з класами **та об‘єктами.»**

Виконав: : Шалівський Віталій
Група: ALK-43

Київ 2025р.

1.Поясніть принцип ООП — успадкування. Наведіть приклади.

Успадкування (inheritance) — це один із головних принципів об’єктно-орієнтованого програмування (ООП).
Воно дозволяє **створювати нові класи (похідні)** на основі **вже існуючих класів (базових)**.
Похідний клас **успадковує всі властивості та методи базового**, але може **доповнювати або перевизначати** їх під власні потреби.

Завдяки успадкуванню можна:

уникати дублювання коду;

розширювати можливості вже написаних класів;

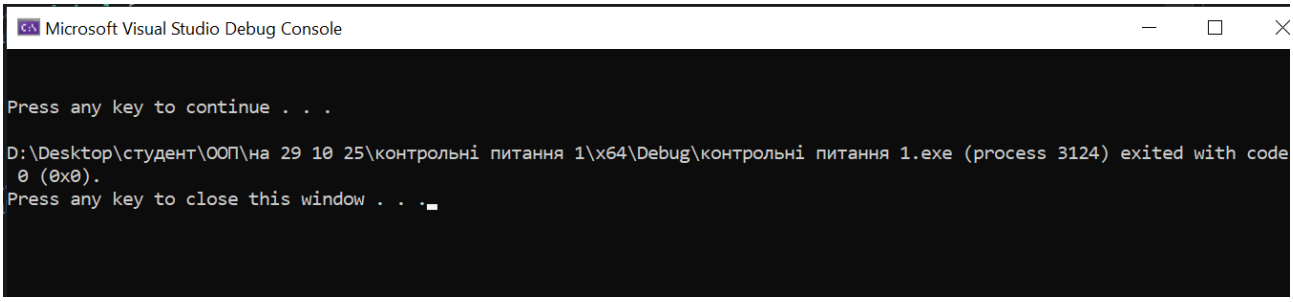
створювати ієрархію об’єктів (наприклад, “Тварина → Собака → Вівчарка”).

Код програми:

```cpp
#include <iostream>
#include <windows.h>
using namespace std;

class Animal {
public:
    void eat() {
        cout << "Тварина їсть" << endl;
    }
};

class Dog : public Animal {
public:
    void bark() {
        cout << "Собака гавкає" << endl;
    }
};
```

```cpp
int main() {
    SetConsoleOutputCP(65001); // Включаємо українське кодування

    Dog dog;
    dog.eat();
    dog.bark();

    system("pause");
    return 0;
}
```

Результат виконання програми:



2.Які ви знаєте специфікатори доступу?

У мові програмування **C++** існує **три основні специфікатори доступу**:

public — відкритий доступ

- Члени класу доступні **з будь-якої частини програми**.
- Використовується для методів і змінних, які мають бути доступні зовні.

private — закритий доступ

- Члени класу доступні **тільки всередині самого класу**.
- Використовується для **захисту даних від прямої зміни**.

protected — захищений доступ

- Члени класу доступні **всередині цього класу та його нащадків (успадкованих класів)**.
- Зовні програми — недоступні.

3.Що відбувається з відкритими та закритими членами базового

класу, якщо базовий клас успадковується як відкритий похідним?

Коли базовий клас успадковується **як public**, то:

Член базового класу	У похідному класі стає	Пояснення
public	public	Відкриті члени залишаються відкритими — ними можна користуватися зовні.
protected	protected	Захищені члени залишаються доступними лише у похідному класі та його нащадках.
private	недоступний	Закриті члени **не успадковуються** — вони залишаються

		прихованими для похідного класу.

4.Що відбувається з закритими та відкритими членами базового

класу, якщо базовий клас успадковується як закритий похідним?

Коли базовий клас успадковується **як private**, то:

Член базового класу	У похідному класі стає	Пояснення
public	private	Відкриті члени стають закритими — більше недоступні зовні.
protected	private	Захищені члени теж стають закритими — доступ тільки всередині похідного класу.
private	недоступний	Закриті члени базового класу залишаються недоступними навіть для похідного.

5.Поясніть, навіщо потрібна категорія захищеності protected?

Категорія protected використовується для створення членів класу, які:

- **недоступні зовні класу** (як private);
- **але доступні в похідних класах**, які успадковують цей клас.

Це проміжний рівень доступу між private і public.

6.Якщо один клас успадковується іншим, яким порядок виклику

конструкторів та деструкторів? Наведіть приклади.

У мові **C++** порядок виконання наступний:

Порядок виклику конструкторів:

1. Спочатку викликається **конструктор базового класу**.
2. Потім — **конструктор похідного класу**.

Порядок виклику деструкторів:

1. Спочатку викликається **деструктор похідного класу**.
2. Потім — **деструктор базового класу**.

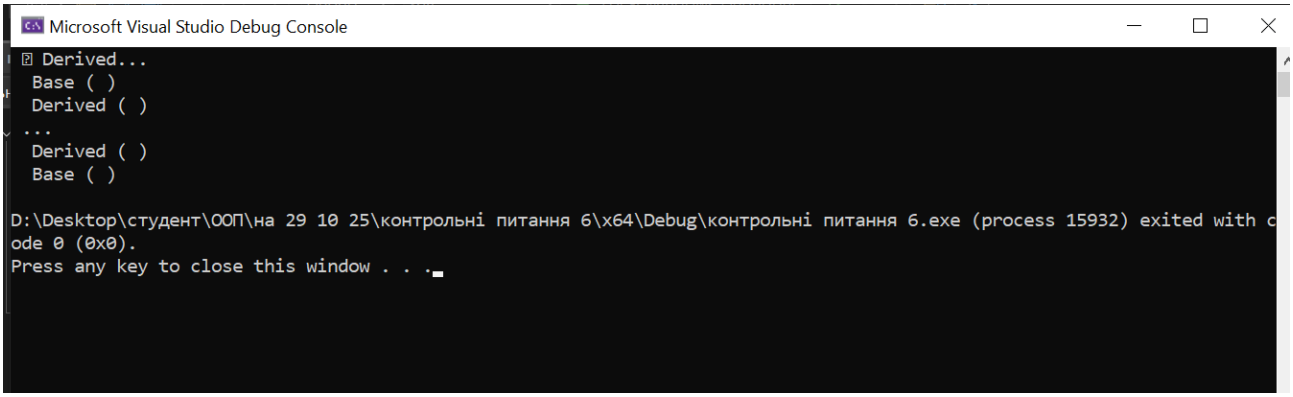
Тобто конструктори викликаються **зверху вниз**, а деструктори — **знизу вгору**.

Код програми:

```cpp
#include <iostream>
#include <windows.h>
using namespace std;

// Базовий клас
class Base {
public:
    Base() {
        cout << "Викликано конструктор Base (Базовий клас)" << endl;
    }

    ~Base() {
        cout << "Викликано деструктор Base (Базовий клас)" << endl;
    }
};

// Похідний клас
class Derived : public Base {
public:
    Derived() {
        cout << "Викликано конструктор Derived (Похідний клас)" << endl;
    }

    ~Derived() {
        cout << "Викликано деструктор Derived (Похідний клас)" << endl;
    }
};
```

```cpp
int main() {
    // 🔧 Установка української мови в консолі
    SetConsoleOutputCP(65001); // UTF-8
    SetConsoleCP(65001);

    cout << "Створення об'єкта Derived..." << endl;
    Derived obj;
    cout << "Завершення програми..." << endl;

    return 0;
}
```

Результат виконаня програми:



7.Що таке множинне успадкування?

Множинне успадкування — це коли похідний клас успадковує властивості та методи від кількох базових класів одночасно.

Тобто **один клас має більше ніж одного "батька"**.

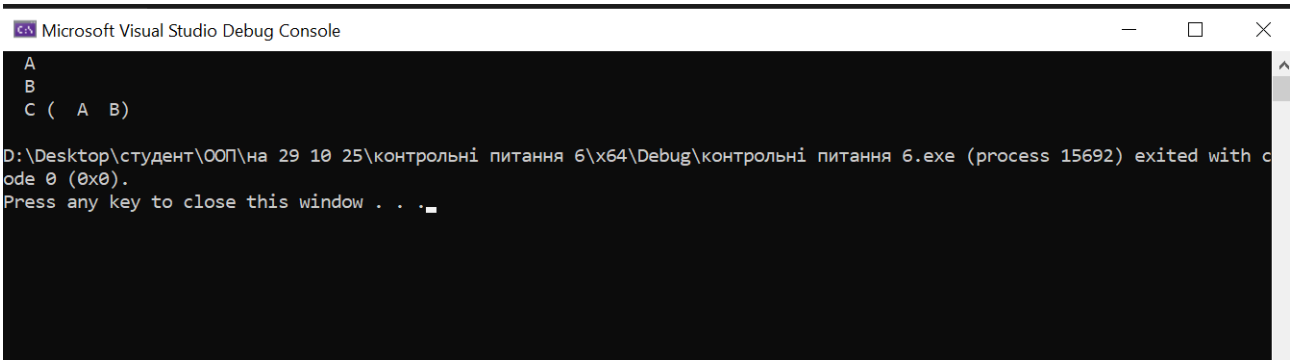
- Клас `C` успадкував усі методи класів `A` та `B`.
- Тепер об'єкт `obj` може викликати функції з **обох базових класів**.
- Таку схему використовують, коли об'єкт повинен мати **властивості кількох типів** одночасно.

Код програми:

```cpp
#include <iostream>
#include <windows.h>
using namespace std;

// Перший базовий клас
class A {
public:
    void showA() {
        cout << "Це клас A" << endl;
    }
};

// Другий базовий клас
class B {
public:
    void showB() {
        cout << "Це клас B" << endl;
    }
};

// Похідний клас, який успадковує від A і B
class C : public A, public B {
public:
    void showC() {
        cout << "Це клас C (успадковує від A і B)" << endl;
    }
};
```

```cpp
int main() {
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    C obj;
    obj.showA(); // метод із класу A
    obj.showB(); // метод із класу B
    obj.showC(); // власний метод класу C

    return 0;
}
```

Результат виконання програми:



Завдання 1

Варіант 1.

Створити базовий клас ФІГУРА, в якому визначено координати однієї з вершин геометричної фігури. Створити похідні класи: ПРЯМОКУТНИК (додатково визначаються довжини двох сторін), КОЛО (додатково визначається радіус), ПРЯМОКУТНИЙ ТРИКУТНИК (додатково визначаються довжини катетів).

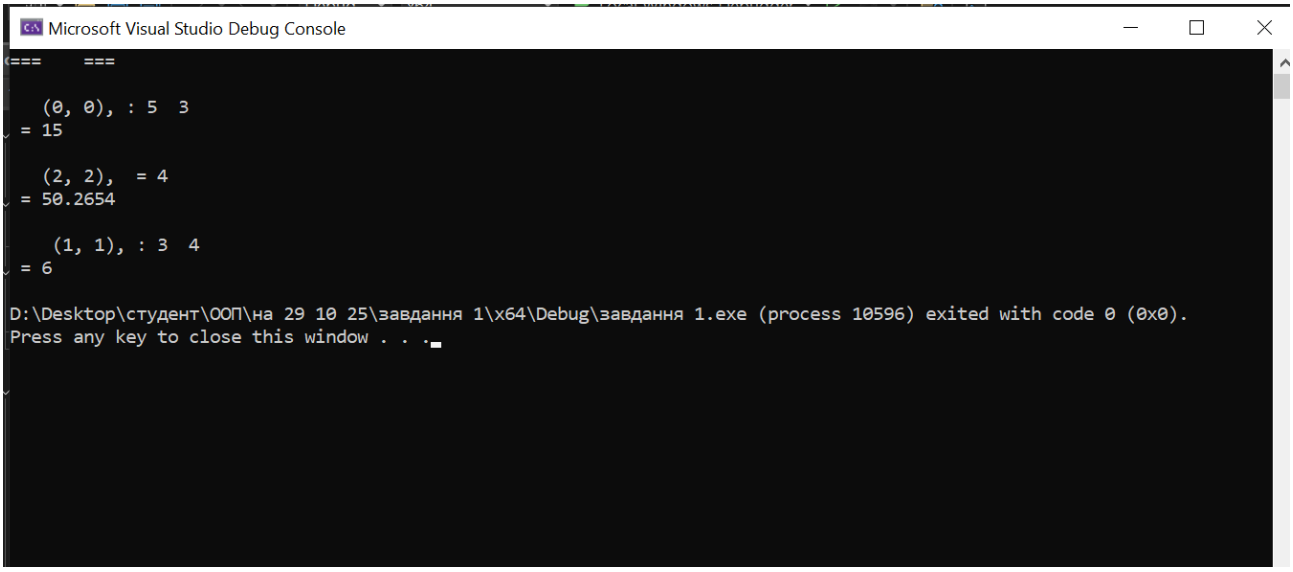
Код програми:

```cpp
#include <iostream>
#include <windows.h>
#include <cmath>
using namespace std;

// Базовий клас
class Figura {
protected:
    double x, y; // координати вершини
public:
    Figura(double x1 = 0, double y1 = 0) {
        x = x1;
        y = y1;
    }

    virtual void ShowInfo() {
        cout << "Фігура з координатами вершини (" << x << ", " << y << ")" << endl;
    }
};
```

```cpp
// Похідний клас: Прямокутник
class Pryamokutnyk : public Figura {
protected:
    double a, b; // сторони
public:
    Pryamokutnyk(double x1, double y1, double a1, double b1)
        : Figura(x1, y1), a(a1), b(b1) {
    }

    void ShowInfo() override {
        cout << "Прямокутник з вершини (" << x << ", " << y << "), сторони: " << a << " і " << b << endl;
        cout << "Площа = " << a * b << endl;
    }
};

// Похідний клас: Коло
class Kolo : public Figura {
protected:
    double r; // радіус
public:
    Kolo(double x1, double y1, double r1) : Figura(x1, y1), r(r1) {}

    void ShowInfo() override {
        cout << "Коло з центром (" << x << ", " << y << "), радіус = " << r << endl;
        cout << "Площа = " << 3.14159 * r * r << endl;
    }
};
```

```cpp
// Похідний клас: Прямокутний трикутник
class PryamTrikutnyk : public Figura {
protected:
    double a, b; // катети
public:
    PryamTrikutnyk(double x1, double y1, double a1, double b1)
        : Figura(x1, y1), a(a1), b(b1) {
    }

    void ShowInfo() override {
        cout << "Прямокутний трикутник з вершини (" << x << ", " << y << "), катети: " << a << " і " << b << endl;
        cout << "Площа = " << (a * b) / 2 << endl;
    }
};

// --- Головна функція ---
int main() {
    SetConsoleOutputCP(65001);
    SetConsoleCP(65001);

    cout << "=== Демонстрація успадкування класів ===\n" << endl;

    Pryamokutnyk pr(0, 0, 5, 3);
    Kolo k(2, 2, 4);
    PryamTrikutnyk t(1, 1, 3, 4);

    pr.ShowInfo();
    cout << endl;

    k.ShowInfo();
    cout << endl;

    t.ShowInfo();

    return 0;
}
```

Результат виконання програми


Microsoft Visual Studio Debug Console
=== ===
(0, 0), : 5 3
= 15
(2, 2), = 4
= 50.2654
(1, 1), : 3 4
= 6
D:\Desktop\студент\ООП\на 29 10 25\завдання 1\x64\Debug\завдання 1.exe (process 10596) exited with code 0 (0x0).
Press any key to close this window . . .