

Лабораторне заняття 9 (2-й семестр)

Поліморфізм в C++. Віртуальні функції

Поліморфізм в програмуванні проявляється, наприклад, в перевантаженні функцій, операторів, операцій. В контексті успадкування поліморфізм можна розглядати, як можливість об'єктів різних класів, що пов'язані відносинами наслідування, реагувати по різному під час виклику одного й того-самого методу

Поліморфізм — можливість об'єктів різних класів, що пов'язані відносинами наслідування, реагувати по різному під час виклику одного й того ж методу. Наприклад, базовий клас **Quad** (Чотирикутник) та похідний від нього клас **Rectangle** (Прямокутник), обидва можуть містити методи розрахунку площі або периметру, але, при цьому, такі методи будуть розрізнятись між собою за реалізацією.

Реалізація поліморфізму в C++ здійснюється засобами *віртуальних функцій* — функцій, що об'являються в базовому класі з використанням ключового слова *virtual* і перевизначаються в одному або декількох похідних класах. Таким чином, кожний похідний клас може мати власну версію віртуальної функції. Важливим моментом забезпечення ідеї поліморфізму є те, що звернення до віртуальної функції відбувається через покажчик (або посилання) на базовий клас, в такому випадку компілятор C++ автоматично визначає, яку саме версію віртуальної функції потрібно викликати, по типу об'єкту, що адресується цим покажчиком, такий вибір відбувається під час виконання програми. Ключове слово *virtual* також може вказуватись і перед назвами методів в похідних класах, але це не є обов'язковим.

По відношенню до всіх віртуальних методів компілятор застосовує стратегію *пізнього* або *динамічного* зв'язування. Це означає, що на етапі компіляції він не визначає, який з методів повинен бути викликаний, а передає відповідальність програмі, яка приймає рішення на етапі виконання, коли вже точно відомо, який тип об'єкта, на який вказує наш покажчик.

Поліморфний клас — клас, який включає поліморфну функцію.

Приклад 1. Розглянемо приклад використання віртуальної функції:

```
class A{
public:
    virtual void who() { // оголошення віртуальної функції.
        cout << "Базовий клас.\n ";
    }
};

class A1 : public A {
public:
    void who() { // перевизначення функції who() для класу A1,
                // ключове слово virtual не є обов'язковим.
        cout << "Перший похідний клас.\n ";
    }
};

class A2 : public A {
```

```

public:
    void who() { // ще одне перевизначення функції who() для класу A2.
        cout << "Другий похідний клас.\n ";
    }
};

int main() {
    A base_object;
    A *p;
    A1 a1_object;
    A2 a2_object;

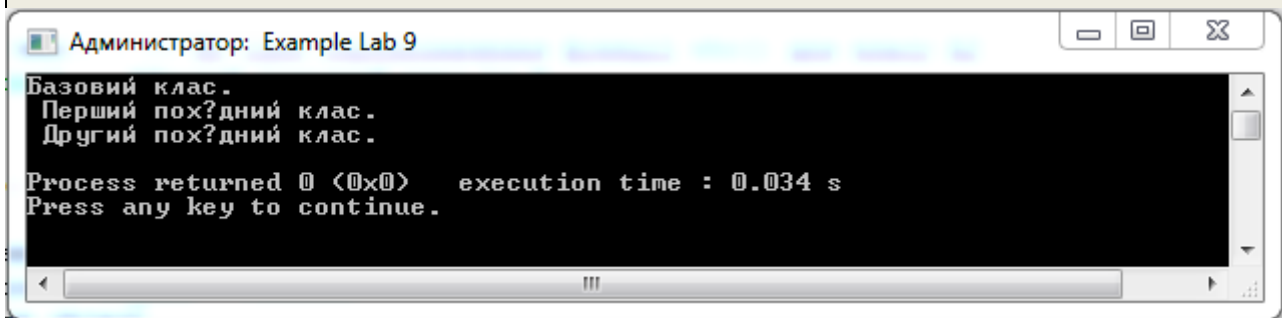
    p = &base_object; // встановлюємо покажчик на об'єкт базового класу.
    p->who(); // викликається метод who() класу A.

    p = &a1_object; // встановлюємо покажчик на об'єкт класу A1.
    p->who(); // викликається метод who() класу A1.

    p = &a2_object; // встановлюємо покажчик на об'єкт класу A2.
    p->who(); // викликається метод who() класу A2.

    return 0;
}

```



На відміну від перевантаження функції в похідному класі, кількість та тип параметрів віртуальних функцій в базовому та похідному класах повинні точно співпадати і мати однакові прототипи, якщо прототипу таких функцій будуть відрізнятися, то функція в похідному класі буде вважатись компілятором просто перевантаженою, а не віртуальною. Крім того, віртуальна функція має бути членом класу, для якого вона визначається, а не його “другом”, але віртуальна функція може бути дружньою для іншого класу. Деструктори в C++ можуть бути віртуальними, а конструктори — ні.

Наслідування віртуальних функцій. Якщо функція об’явлена як віртуальна, то вона залишається такою незалежно від кількості рівнів похідних класів, в яких вона використана. Наприклад, якщо б клас **A2** було б наслідувало від **A1**, а не від класу **A**, то функція `who()` все одно б залишалась віртуальною і механізм поліморфізму працював би коректно:

```

class A2 : public A1 {
public:
    void who() {

```

```

        cout << "Другий похідний клас.\n";
    }
};
int main() {
    ...
    p = &a2_object;
    p->who();
    return 0;
}

```

Якщо похідний клас не перевизначає віртуальну функцію, то використовується варіант функції, що визначений в базовому класі:

```

class A2 : public A { // Функцію who() не визначено
};
...
int main() {
    ...
    p = &a2_object;
    p->who();
    return 0;
}

```

Суто віртуальні функції та абстрактні класи. В багатьох випадках базовий клас може задавати тільки каркас для поняття і в ньому не може бути реалізовано деякі функції, наприклад, узагальнений клас **Shape** (Форма) ніяким чином не може реалізувати метод `area()`. В таких випадках в базовому класі об'являються функції без чіткої реалізації, які повинні бути обов'язково реалізовані в похідних класах. Такі функції мають назву **суто віртуальні**, будь-який похідний клас, який успадковується від базового, що містить суто віртуальну функцію має примусово реалізувати (визначити) таку функцію. Формат об'явлення суто віртуальної функції в базовому класі:

virtual *тип ім'я_функції*(*список_параметрів*) = 0;

Абстрактний клас — це клас, який містить хоча б одну суто віртуальну функцію. У абстрактного класу не може бути об'єктів, спроба створення об'єктів такого класу призведе до помилки компіляції, винятком є об'явлення покажчику на тип базового класу, який потім використовується для посилання на об'єкти похідних класів (див. приклади вище).

Приклад 2:

```

#include <iostream>

using namespace std;

class A{ // оголошення абстрактного класу A.
public:
    virtual void who() = 0; // метод who() є суто віртуальним.
}

```

```
};

class A1 : public A {
public:
    void who() { // визначення функції who() для класу A1.
        cout << "Перший похідний клас.\n ";
    }
};

class A2 : public A { // функцію who() не визначено для класу A2.
};

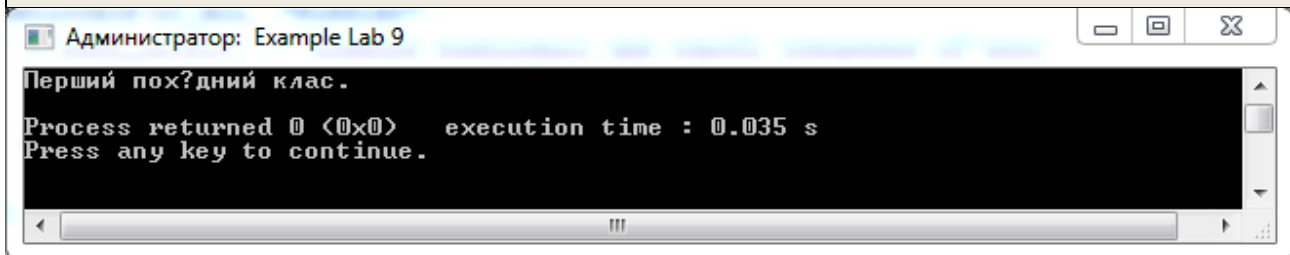
int main() {
    // A base_object; // !помилка компіляції при спробі створення об'єкта
    //абстрактного класу.

    A *p; // покажчик на об'єкт абстрактного класу є допустимим.

    A1 a1_object; // об'єкт буде створено, оскільки метод who() реалізовано.

    // A2 a2_object; // помилка компіляції при спробі створення об'єкта
    //похідного класу без реалізованої віртуальної функції.
    p = &a1_object; // єдиний коректний варіант виклику методу who() .
    // можливий тільки для класу A1.

    p->who();
    return 0;
}
```



Приклад 3: Ще один приклад використання абстрактних класів та поліморфізму:

```
#include <iostream>
#include <string>
```

```
using namespace std;

class Animal { // Базовый клас
protected:
    int size;
public:
    void setSize(int s){
        size = s;
    };
    virtual string soundsLike() = 0;
};

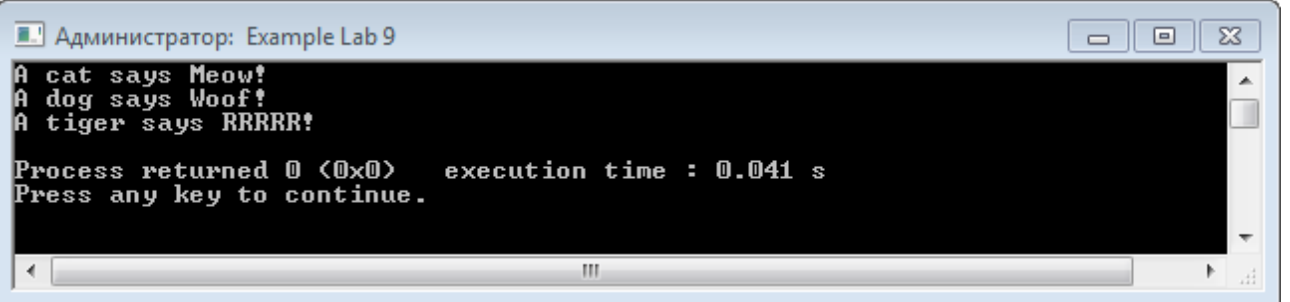
class Cat : public Animal {
public:
    string soundsLike(){
        return "Meow!";
    }
};

class Dog : public Animal {
public:
    string soundsLike(){
        return "Woof!";
    }
};

class Tiger : public Cat {
public:
    string soundsLike(){
        if(size < 50)
            return Cat::soundsLike();
        else
            return "RRRRR!";
    }
};

int main()
{
    Animal *a;
    Cat cat;
    Dog dog;
```

```
Tiger tiger;
tiger.setSize(60);
a = &cat;
cout << "A cat says " << a->soundsLike() << endl;
a = &dog;
cout << "A dog says " << a->soundsLike() << endl;
a = &tiger;
cout << "A tiger says " << a->soundsLike() << endl;
return 0;
}
```



Лабораторне заняття 9.

Хід виконання завдання:

1. Створити абстрактний клас для заданого поняття з вказаною віртуальною функцією.
2. Створити два класи-нащадки з реалізацією віртуальної функції.
3. Написати програму, яка демонструє поліморфізм створених класів.

Варіанти завдань.

1. Базовий абстрактний клас **Polygon** (Плоский багатокутник) містить суто віртуальну функцію *area()* (площа фігури). Похідні класи **Triangle** (Трикутник) та **Rectangle** (Прямокутник) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться відповідні площі.
2. Базовий абстрактний клас **Solid** (Тверде тіло) містить суто віртуальну функцію *area()* (повна площа поверхні тіла). Похідні класи **Sphere** (Сфера) та **Cube** (Куб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться площі їх поверхні.
3. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *toFloat()* (перетворення в дійсне число). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати їх перетворення в дійсне число.
4. Базовий абстрактний клас **Vector** (Радіус-вектор) задається координатами точки та містить суто віртуальну функцію *add()* (додавання скаляру до усіх координат). Похідні класи **Vector2D** (Вектор на площині) та **Vector3D** (Вектор у просторі) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і, через відповідні геттери, виводяться їх координати перед та

після операції додавання.

5. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *toStr()* (перетворення в строку). Похідні класи **Mixed** (Мішаний дріб) та **Complex** (Комплексне число) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться їх строкові представлення.
6. Базовий абстрактний клас **Polygon** (Плоский багатокутник) містить суто віртуальну функцію *area()* (площа фігури). Похідні класи **Trapezoid** (Трапеція) та **Rectangle** (Прямокутник) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться відповідні площі.
7. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *add()* (складення з іншим числом). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати операції додавання аналогічного числа: *Rational+Rational* та *Decimal+Decimal*.
8. Базовий абстрактний клас **Solid** (Тверде тіло) містить суто віртуальну функцію *volume()* (об'єм тіла). Похідні класи **Cone** (Конус) та **Cube** (Куб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться їх об'єми.
9. Базовий абстрактний клас **Polygon** (Плоский багатокутник) містить суто віртуальну функцію *perimeter()* (периметр фігури). Похідні класи **Trapezoid** (Трапеція) та **Square** (Квадрат) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться відповідні периметри.
10. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *multiply()* (множення на інше число). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати операції множення на аналогічне число: *Rational*Rational* та *Decimal*Decimal*.
11. Базовий абстрактний клас **Polygon** (Плоский багатокутник) містить суто віртуальну функцію *perimeter()* (периметр фігури). Похідні класи **Triangle** (Трикутник) та **Trapezoid** (Трапеція) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться відповідні периметри.
12. Базовий абстрактний клас **Line** (Лінія) задається списком координат її точок та містить суто віртуальну функцію *length()* (довжина лінії). Похідні класи **Segment** (Відрізок) та **Polyline** (Ламана) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і, через відповідні геттери, виводяться списки їх координат вершин і відповідні довжини.
13. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *toInt()* (перетворення в ціле число). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати їх перетворення в ціле число.
14. Базовий абстрактний клас **Solid** (Тверде тіло) містить суто віртуальну функцію *volume()* (об'єм тіла). Похідні класи **Cylinder** (Циліндр) та **Sphere** (Сфера) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються

об'єкти похідних класів і виводяться їх об'єми.

15. Базовий абстрактний клас **Vector** (Радіус-вектор) задається координатами точки та містить суто віртуальну функцію *module()* (модуль або довжина вектору). Похідні класи **Vector2D** (Вектор на площині) та **Vector3D** (Вектор у просторі) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і, через відповідні геттери, виводяться списки їх координат і відповідні довжини.
16. Базовий абстрактний клас **Vector** (Радіус-вектор) задається координатами точки та містить суто віртуальну функцію *scale()* (масштабування або множення на скаляр усіх координат вектору). Похідні класи **Vector2D** (Вектор на площині) та **Vector3D** (Вектор у просторі) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і, через відповідні геттери, виводяться їх координати перед та після операції масштабування.
17. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *toStr()* (перетворення в строку). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати їх строкові представлення.
18. Базовий абстрактний клас **Solid** (Тверде тіло) містить суто віртуальну функцію *area()* (повна площа поверхні тіла). Похідні класи **Cylinder** (Циліндр) та **Cone** (Конус) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться площі їх поверхні.
19. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *add()* (складення з іншим числом). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати операції додавання аналогічного числа: *Rational+Rational* та *Decimal+Decimal*.
20. Базовий абстрактний клас **Polygon** (Плоский багатокутник) містить суто віртуальну функцію *perimeter()* (периметр фігури). Похідні класи **Trapezoid** (Трапеція) та **Square** (Квадрат) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться відповідні периметри.
21. Базовий абстрактний клас **Solid** (Тверде тіло) містить суто віртуальну функцію *area()* (повна площа поверхні тіла). Похідні класи **Sphere** (Сфера) та **Cube** (Куб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться площі їх поверхні.
22. Базовий абстрактний клас **Vector** (Радіус-вектор) задається координатами точки та містить суто віртуальну функцію *module()* (модуль або довжина вектору). Похідні класи **Vector2D** (Вектор на площині) та **Vector3D** (Вектор у просторі) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і, через відповідні геттери, виводяться списки їх координат і відповідні довжини.
23. Базовий абстрактний клас **Number** (Число) містить суто віртуальну функцію *toFloat()* (перетворення в дійсне число). Похідні класи **Rational** (Раціональне число) та **Decimal** (Десятковий дріб) мають реалізувати вказану віртуальну функцію. Написати програму, в якій створюються об'єкти похідних класів і виводяться результати їх перетворення в дійсне число.

Контрольні запитання

- 1) Поясніть поняття поліморфізму в контексті ООП.
- 2) Для чого застосовуються віртуальні функції?
- 3) Що таке раннє та пізнє зв'язування?
- 4) Як відбувається процес пізнього динамічного зв'язування для віртуальних функцій?
- 5) Що таке пере визначення віртуальної функції? Їх особливості.
- 6) Поясніть застосування суто віртуальних функцій і абстрактних класів в ООП.