



Bài 5: Đa hình



Nội dung

1. Ví dụ về Animal, Dog và Cat
2. Phương thức ảo
3. Đa hình
4. Huỷ tử ảo
5. Lớp cơ sở trừu tượng
6. Ví dụ về Shape, Circle và Rect
7. Review
8. Bài tập

Ôn lại

1. Quan hệ bao gồm và kế thừa
2. Kế thừa
 - Kế thừa đơn
 - Đa kế thừa
 - Kế thừa ảo
3. Các kiểu kế thừa: public, protected, private

3

Animal, Dog và Cat

- Giả sử có các lớp Animal, Dog và Cat như sau:
- ```
class Animal
```
- ```
{
```
- ```
protected:
```
- ```
    char name[30];
```
- ```
public:
```
- ```
    Animal(char *name);
```
- ```
 void Speak();
```
- ```
};
```
- ```
Animal::Animal(char *name)
```
- ```
{ strcpy(this->name, name); }
```
- ```
void Animal::Speak()
```
- ```
{ cout<<"Hello, I am an animal"<<endl; }
```

4

Animal, Dog và Cat

```
▪ // lớp Dog
▪ class Dog : public Animal
▪ {
▪ public:
▪     Dog(char *name) : Animal(name) {}
▪     void Speak(); // nạp chồng Speak của lớp cha
▪ };

▪ void Dog::Speak()
▪ {
▪     cout<<"My name is "<<name<<" , go go !"<<endl;
▪ }
```

5

Animal, Dog và Cat

```
▪ // lớp Cat
▪ class Cat : public Animal
▪ {
▪ public:
▪     Cat(char *name) : Animal(name) {}
▪     void Speak(); // nạp chồng Speak của lớp cha
▪ };

▪ void Cat::Speak()
▪ {
▪     cout<<"My name is "<<name<<" , meoo !"<<endl;
▪ }
```

6

Animal, Dog và Cat

```
1. // hàm main
2. int main()
3. {
4.     Animal* ani; // con trỏ tới các đối tượng Animal
5.     Dog dog("Tony"); // đối tượng thuộc lớp Dog
6.     Cat cat("Fluffy"); // đối tượng thuộc lớp Cat

7.     ani = &dog;
8.     ani->Speak();


9.     ani = &cat;
10.    ani->Speak();
11.    return 0;
12. }
```

7

Animal, Dog và Cat

- Kết quả màn hình:
 - Hello, I am an animal
 - Hello, I am an animal
- Giải thích:
 - Khi biên dịch, chương trình sẽ gán lời gọi Speak với đối tượng của lớp Animal (dòng 8 và 10). Nó sẽ gọi Speak của lớp Animal khi câu lệnh được thực hiện.
 - Quá trình này gọi là kết nối tĩnh (static binding) - phương thức gọi được xác định tại thời điểm dịch (compile time).


8



Phương thức ảo (virtual method)

- Để cho kết quả đúng như mong muốn, ta cần khai báo Speak trong Animal là phương thức ảo.
 - ***virtual*** void Speak();
- Phương thức ảo là phương thức của lớp cơ sở và được định nghĩa lại trong lớp dẫn xuất. Khi một con trỏ của lớp cơ sở gọi phương thức ảo, chương trình sẽ chọn phương thức cần thiết (dựa trên đối tượng gọi) để thực hiện. Quá trình này gọi là kết nối động (dynamic binding) - phương thức gọi được xác định vào lúc chạy (execution time).

9



Phương thức ảo (virtual method)

- Kết quả sau khi khai báo Speak() là hàm ảo:
 - My name is Tony, go go !
 - My name is Fluffy, meoo !
- Chú ý: Phương thức ảo phải được gọi thông qua con trỏ hoặc tham chiếu.

10

Một số ví dụ

```
int main() // main 1
{
    Animal *ani;
    Animal x("Noname");
    Dog dog("Buddy");
    Cat cat("Chip");

    x.Speak();
    dog.Speak();
    cat.Speak();
    x = dog;
    x.Speak();
    ani = &x;
    ani->Speak();
    ani = &cat;
    ani->Speak();
    return 0;
}
```

11

Một số ví dụ

```
void testPointer(Animal *ani)
{ cout<<"Goi thong qua con tro: "<<endl; ani->Speak(); }

void testReference(Animal &ani)
{ cout<<"Goi thong qua tham chieu: "<<endl; ani.Speak(); }

void testValue(Animal ani)
{ cout<<"Goi thong qua doi so gia tri: "<<endl; ani.Speak(); }

int main() // main 2
{
    Animal *ani;
    Dog dog("Buddy");
    ani = &dog;
    testPointer(ani);
    testReference(*ani);
    testValue(*ani);
    return 0;
}
```

12

Một số ví dụ

- Giả sử muốn có nhiều động vật
 - `int main()`
 - {
 - `Animal *ani[10];` // mảng con trỏ đến các động vật
 - `Dog dog("Rover"); ani[0] = &dog;`
 - `Cat cat1("Spot"); ani[1] = &cat1;`
 - `Cat cat2("Chip"); ani[2] = &cat2;`
 - `Cat cat3("Buddy"); ani[3] = &cat3;`
 - `ListAnimal(ani, 4);`
 - }
 - `void ListAnimal(Animal *ani[], int n)`
 - {
 - `for(int i = 0; i < n; i++)`
 - `ani[i]->Speak();`
 - }

13

Một số ví dụ

- Kết quả:
 - My name is Rover, go go !
 - My name is Spot, meoo !
 - My name is Chip, meoo !
 - My name is Buddy, meoo !
- Nhận xét: Nếu không có hàm ảo, ta phải:
 - Khai báo 2 mảng để lưu 2 loại Dog và Cat tương ứng.
 - Khi muốn liệt kê animal phải duyệt từng mảng.
 - Khi muốn thêm một loài vật nuôi mới thì phải tạo thêm mảng để lưu giữ loài vật này.

14

Lớp MiniPig

- Với đa hình, ta có thể thêm các lớp mới mà không ảnh hưởng tới các module có sẵn:
 - *// lớp MiniPig*
 - `class MiniPig : public Animal`
 - `{`
 - `public:`
 - `MiniPig(char *name) : Animal(name) {}`
 - `void Speak();`
 - `};`
 - `void MiniPig::Speak()`
 - `{`
 - `cout<<"My name is "<<name<<" , un in !"<<endl;`
 - `}`

15

Lớp MiniPig

- Sử dụng:
 - `int main()`
 - `{`
 - `Animal *ani[10];`
 - `Dog dog("Rover"); ani[0] = &dog;`
 - `Cat cat1("Spot"); ani[1] = &cat1;`
 - `Cat cat2("Chip"); ani[2] = &cat2;`
 - `Cat cat3("Buddy"); ani[3] = &cat3;`
 - `MiniPig pig1("Inky"); ani[4] = &pig1;`
 - `MiniPig pig2("Princess"); ani[5] = &pig2;`
 - `ListAnimal(ani, 6);`
 - `return 0;`
 - `}`

16



Đặc điểm của OOP

- Tính đa hình (polymorphism)
 - Tính đa hình giúp dễ mở rộng chương trình. Chương trình có thể được viết để xử lý các đối tượng tổng quát rồi sau đó đưa vào các đối tượng cụ thể. Đa hình cho phép nhiều cách xử lý khác nhau với cùng một phương thức, tùy vào mỗi đối tượng cụ thể.
 - Tính đa hình trong C++ thể hiện qua các hàm ảo (virtual). Khi một con trỏ của lớp cơ sở gọi hàm ảo, chương trình sẽ chọn hàm được gọi dựa vào đối tượng đang trỏ tới tại thời điểm chạy (execution time). Quá trình này gọi là kết nối động (dynamic binding).

17



Một số chú ý đối với hàm ảo

- Chú ý:
 - Hàm ảo trong lớp dẫn xuất phải giống hàm của lớp cơ sở.
 - Đặt từ khoá virtual với hàm ảo trong lớp cơ sở và **nên đặt virtual trong cả lớp dẫn xuất.**
 - Nếu lớp dẫn xuất không định nghĩa lại hàm ảo của lớp cơ sở, nó sẽ sử dụng hàm của lớp cơ sở.
 - Không thể khai báo cấu tử là hàm ảo.
 - Có thể khai báo hủy tử là hàm ảo.

18

Hủy tử ảo

- Ví dụ:
 - `Animal *ani = new Cat("Buddy");`
 - `delete ani;` // hủy tử của lớp *Animal* sẽ được gọi
- Nếu khai báo hủy tử của *Animal* và *Cat* là virtual thì hủy tử của *Cat* được gọi:
 - `Animal *ani = new Cat("Buddy");`
 - `delete ani;` // hủy tử của *Cat* được gọi (sau đó đến // hủy tử của *Animal*)
- Các lớp sử dụng hàm ảo nên khai báo hủy tử ảo để việc hủy đối tượng được chính xác, đặc biệt trong các lớp sử dụng bộ nhớ động.

19

Lớp cơ sở trừu tượng

- Lớp cơ sở trừu tượng (abstract base class) được dùng để định nghĩa các tính chất tổng quát, chung cho các lớp khác.
 - Lớp cstk không có thể hiện (instance).
 - Trong định nghĩa của lớp cstk phải có ít nhất một hàm ảo thuần túy.
- Hàm ảo thuần túy (pure virtual function)
 - Là hàm ảo không có cài đặt.
 - Được khai báo khởi tạo = 0;

20

Lớp cơ sở trừu tượng

- Ví dụ:
 - Lớp Hình (Shape) có thể là lớp csstt của lớp Hình tròn (Circle), Hình chữ nhật (Rectangle)... Hàm ảo thuần túy là Tính diện tích, Tính chu vi...
 - Lớp Nhân sự (Employee) có thể là lớp csstt của lớp Công nhân (Worker), Người quản lý (Manager)... Hàm ảo thuần túy là Tính lương, Hiển thị thông tin...
- Chú ý:
 - Hàm ảo thuần túy của một lớp csstt phải được định nghĩa lại trong lớp dẫn xuất của nó, nếu không thì lớp dẫn xuất sẽ kế thừa lại hàm ảo thuần túy đó và trở thành một lớp csstt khác.

21

Shape, Circle và Rectangle

```
▪ class Shape
▪ {
▪ private:
▪     char name[20];
▪ public:
▪     Shape(char *name); // cấu tử
▪     virtual float GetArea() = 0; // hàm ảo thuần túy
▪     void Display();
▪ };

▪ Shape::Shape(char *name)
▪ { strcpy(this->name, name); }

▪ void Shape::Display()
▪ {
▪     cout<<"Hello, I am a "<<name<<endl;
▪     cout<<"My area is "<<GetArea()<<endl; // gọi hàm ảo
▪ }
```

22

Shape, Circle và Rectangle

```
▪ class Circle: public Shape
▪ {
▪   private:
▪     int r;
▪   public:
▪     Circle(int r); // cấu tử
▪     virtual float GetArea(); // định nghĩa lại hàm ảo của lớp cơ sở
▪ };

▪ Circle::Circle(int r) : Shape("circle")
▪ {
▪   this->r = r;
▪ }

▪ float Circle::GetArea()
▪ {
▪   return 3.14*r*r;
▪ }
```

23

Shape, Circle và Rectangle

```
▪ class Rect: public Shape
▪ {
▪   private:
▪     int a, b;
▪   public:
▪     Rect(int a, int b); // cấu tử
▪     virtual float GetArea(); // định nghĩa lại hàm ảo của lớp cơ sở
▪ };

▪ Rect::Rect(int a, int b) : Shape("rectangle")
▪ {
▪   this->a = a;
▪   this->b = b;
▪ }

▪ float Rect::GetArea()
▪ {
▪   return a*b;
▪ }
```

24

Shape, Circle và Rectangle

```
int main()
{
    Shape *s;    // con trỏ tới các đối tượng Shape
    Rect r(3,5); // đối tượng thuộc Rect
    Circle c(2); // đối tượng thuộc Circle

    s = &r;
    s->Display(); // gọi phương thức của lớp Shape

    s = &c;
    s->Display(); // gọi phương thức của lớp Shape

    // khai báo dưới đây sẽ lỗi vì Shape là một lớp csst
    // Shape sh("shape");
    return 0;
}
```

25

Review

1. Hàm ảo là gì ?
2. Tính đa hình là gì ?
3. Huỷ tử ảo là gì ? Tại sao cần huỷ tử ảo ?
4. Lớp cơ sở trừu tượng là gì ? Cho ví dụ ?
5. Hàm ảo thuần tuý là gì ?

26



Bài tập về nhà

1. Xây dựng hoàn chỉnh lớp Animal, lớp Dog và lớp Cat như bài 4 (Kế thừa).
2. Xây dựng hoàn chỉnh lớp Shape, lớp Circle và lớp Rect dựa theo nội dung trong bài.