LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương 7: Đa hình (p2)

Trình bày: ThS. Lê Thanh Trọng

NỘI DUNG

1. Phương thức ảo
2. Phương thức thuần ảo

NỘI DUNG

1. **Phương thức ảo**
2. Phương thức thuần ảo

Phương thức ảo

- Phương thức ảo:
  - Là cách thể hiện tính đa hình trong ngôn ngữ C++.
  - Các phương thức ở lớp cơ sở có tính đa hình phải được định nghĩa là một phương thức ảo
- Con trỏ thuộc lớp cơ sở có thể trỏ đến lớp con:
  - Nguoi* pn=new SinhVien(“Le Vien Sinh”,TH11001,1982);

Phương thức ảo

- Mong muốn thông qua con trỏ thuộc lớp cơ sở có thể truy xuất hàm thành phần được định nghĩa lại ở lớp con

```
pn->Xuat();
```

- /Mong muon: goi Xuat cua lop Sinh vien,
- /Thuc te: goi Xuat cua lop Nguoi

Phương thức ảo

- Phương thức ảo cho phép giải quyết vấn đề trên
- Ta qui định một hàm thành phần là phương thức ảo bằng cách thêm từ khóa virtual vào trước khai báo hàm
- Trong ví dụ trên, ta thêm từ khóa **virtual** vào trước khai báo của hàm Xuat

Phương thức ảo – Ví dụ

```
class Nguoi {
protected:
    char *HoTen;
    int NamSinh;
public:
    Nguoi( char *ht,int ns):NamSinh(ns){HoTen = strdup(ht);}
    ~Nguoi() {delete [ ] HoTen;}
    void An() const { cout << HoTen << " an 3 chen com";}
    virtual void Xuat() const {
            cout << "Nguoi, ho ten: " << HoTen
            cout << " sinh " << NamSinh;
    }
};
```

Thêm lớp con mới

- Dùng phương thức ảo, ta dễ dàng nâng cấp sửa chữa
- Thêm một loại đối tượng mới rất đơn giản, không cần sửa đổi thao tác xử lý (XuatDs)
- Qui trình thêm chỉ là xây dựng lớp con kế thừa lớp cơ sở và định nghĩa lại phương thức (ảo) ở lớp mới tạo nếu cần

Thêm lớp con mới – Ví dụ

```
class CaSi : public Nguoi{
protected:
    double CatXe;
public:
    CaSi( char *ht, double cx, int ns): Nguoi(ht,ns),CatXe(cx) { }
    void Xuat() const {
        cout<<"Ca si, "<<HoTen<<" co cat xe "<< CatXe;
    }
};
```

Thêm lớp con mới

❖Hàm XuatDs không thay đổi, nhưng nó có thể hoạt động cho các loại đối tượng ca sĩ thuộc lớp mới ra đời

```
void XuatDs( int n, Nguoi *an[]){
    for ( int i = 0; i < n; i++){
        an[i]->Xuat();
        cout << "\n";
    }
}
```

Lưu ý khi sử dụng phương thức ảo

- Phương thức ảo chỉ hoạt động thông qua con trỏ.
- Muốn một hàm trở thành phương thức ảo có hai cách:
  - Khai báo với từ khoá virtual
  - Hoặc phương thức tương ứng ở lớp cơ sở đã là phương thức ảo

Lưu ý khi sử dụng phương thức ảo

- Phương thức ảo chỉ hoạt động nếu các phương thức ở lớp cơ sở và lớp con có nghi thức giao tiếp giống hệt nhau
- Nếu ở lớp con định nghĩa lại phương thức ảo thì sẽ gọi phương thức ở lớp cơ sở (gần nhất có định nghĩa)

Cơ chế thực hiện phương thức ảo

- Khi gọi một thao tác, khả năng chọn đúng phiên bản tùy theo đối tượng để thực hiện thông qua con trỏ đến lớp cơ sở được gọi là tính đa hình (polymorphisms)
- Cơ chế đa hình được thực hiện nhờ ở mỗi đối tượng có thêm một bảng phương thức ảo. Bảng này chứa địa chỉ của các phương thức ảo và nó được trình biên dịch khởi tạo một cách ngầm định khi thiết lập đối tượng

Cơ chế thực hiện phương thức ảo

- Khi thao tác được thực hiện thông qua con trỏ, hàm có địa chỉ trong bảng phương thức ảo sẽ được gọi
- Trong ví dụ trên, mỗi đối tượng thuộc lớp cơ sở Nguoi có bảng phương thức ảo có một phần tử là địa chỉ hàm Nguoi::Xuat
- Mỗi đối tượng thuộc lớp SinhVien có bảng tương tự nhưng nội dung là địa chỉ của hàm SinhVien::Xuat

Phương thức hủy bỏ ảo

- Trong ví dụ quản lý danh sách các đối tượng thuộc các lớp Nguoi, SinhVien, CongNhan,… Thao tác dọn dẹp đối tượng là cần thiết

```
const int N = 4;
void main(){
    Nguoi *a[N];
    a[0] = new SinhVien("Vien Van Sinh", "20001234“,1982);
    a[1] = new NuSinh("Le Thi Ha Dong", "20001235“,1984);
    a[2] = new CongNhan("Tran Nan Cong", 1000000, 1984);
    a[3] = new Nguoi("Nguyen Thanh Nhan", 1960);
    XuatDs(4,a);
    for ( int i = 0; i < 4; i++)
        delete a[i];
}
```

Phương thức hủy bỏ ảo

- Thông qua con trỏ thuộc lớp cơ sở Nguoi, chỉ có phương thức hủy bỏ của lớp Nguoi được gọi
- Để bảo đảm việc dọn dẹp là đầy đủ, ta phải dùng phương thức hủy bỏ ảo

```
class Nguoi{
    protected:
        char *HoTen; int NamSinh;
    public:
        Nguoi(char *ht, int ns):NamSinh(ns) {
            HoTen = strdup(ht);
        }
        virtual ~Nguoi() {
            delete [ ] HoTen;
        }
        virtual void Xuat(ostream &os) const {//…}
};
```

NỘI DUNG

1. Phương thức ảo
2. **Phương thức thuần ảo**

Phương thức thuần ảo và lớp cơ sở trừu tượng

- Lớp cơ sở trừu tượng là lớp cơ sở không có đối tượng nào thuộc chính nó
- Xét các lớp Circle, Rectangle, Square kế thừa từ lớp Shape
- Các hàm trong lớp Shape có nội dung nhưng nội dung không có ý nghĩa, đồng thời luôn có thể tạo được đối tượng thuộc lớp Shape, điều này không đúng với tư tưởng của phương pháp luận hướng đối tượng

Phương thức thuần ảo và lớp cơ sở trừu tượng

- Có thể thay thế cho nội dung không có ý nghĩa bằng phương thức ảo thuần tuý. Phương thức ảo thuần tuý là phương thức ảo không có nội dung
- Khi lớp có phương thức ảo thuần tuý, lớp trở thành lớp cơ sở trừu tượng, không thể tạo đối tượng thuộc lớp cơ sở trừu tượng
- Có thể định nghĩa phương thức ảo thuần tuý, nhưng chỉ có các đối tượng thuộc lớp con có thể gọi nó

Phương thức thuần ảo và lớp cơ sở trừu tượng

- Trong ví dụ trên, các hàm thành phần trong lớp Shape là phương thức ảo thuần tuý. Nó bảo đảm không thể tạo được đối tượng thuộc lớp Shape

- Ví dụ trên cũng định nghĩa nội dung cho phương thức ảo thuần tuý, nhưng chỉ có các đối tượng thuộc lớp con có thể gọi

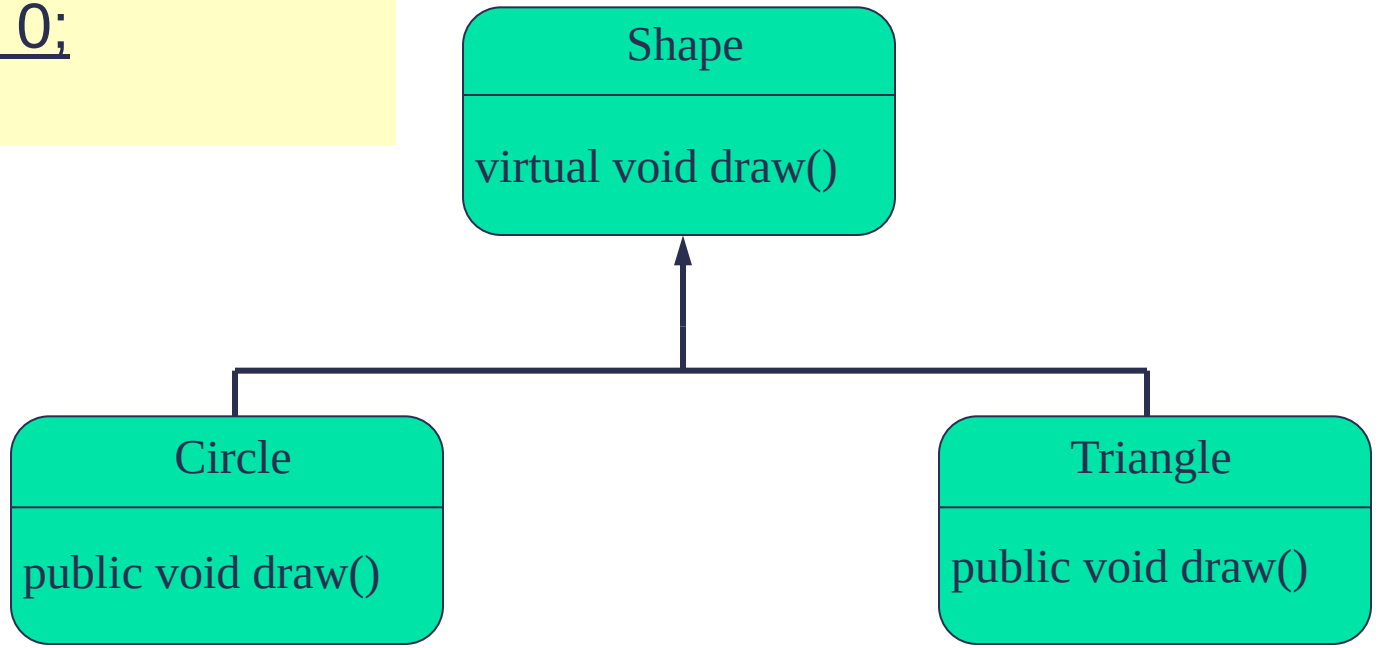
Phương thức thuần ảo và lớp cơ sở trừu tượng

- Phương thức ảo thuần tuý có ý nghĩa cho việc tổ chức sơ đồ phân cấp các lớp, nó đóng vai trò **"chừa sẵn chỗ trống"** cho các lớp con điền vào với phiên bản phù hợp

- Bản thân các lớp con của lớp cơ sở trừu tượng cũng có thể là lớp cơ sở trừu tượng

Ví dụ

```
class Shape    //Lớp trừu tượng
{
    public :
    //Hàm thuần ảo
    virtual void draw() = 0;
}
```



Ví dụ

```
class Circle : public Shape {          //Lớp trừu tượng
    public:
    void print(){
      cout << “I am a circle” << endl;
    }
class Rectangle : public Shape {   //Lớp không còn là trừu tượng
    public :
    void draw(){                              // Override Shape::draw()
      cout << “Drawing Rectangle” << endl;
    }
```

```
Shape *s;
Rectangle r;  // OK
Circle c;        // Error : variable of an abstract class
```

Tóm tắt về bài học Đa hình

- Tính đa hình xuất hiện **khi có sự kế thừa** giữa các lớp, có những phương thức tổng quát cho mọi lớp dẫn xuất nên có mặt ở lớp cơ sở nhưng **nội dung của nó chỉ được xác định ở các lớp dẫn xuất** cụ thể
- Đa hình là khả năng các đối tượng thuộc các lớp khác nhau có khả năng hiểu (hiện thực) cùng một thông điệp (phương thức) theo các cách khác nhau

Tóm tắt về bài học Đa hình

- Có hai cách để hiện thực đa hình (ở C++)
  - Vùng chọn kiểu
  - Phương thức ảo
- **Vùng chọn kiểu**
  - Thêm một vùng dữ liệu vào lớp cơ sở để nhận diện
  - Vùng này có giá trị phụ thuộc vào loại của đối tượng và được gọi là vùng chọn kiểu
  - Nhược điểm: Mã lệnh dài dòng, thủ công (nhiều switch case), dễ sai sót, khó sửa, kKhó nâng cấp, bảo trì

Tóm tắt về bài học Đa hình

❖Phương thức ảo:

- ❑Là cách thể hiện tính đa hình trong ngôn ngữ C++, chỉ hoạt động thông qua con trỏ
- ❑Các phương thức ở lớp cơ sở có tính đa hình phải được định nghĩa là một phương thức ảo
- ❑Cú pháp: Khai báo với từ khoá virtual trước hàm (lớp cha)

```
virtual void Xuat() const {}
```

❖Để bảo đảm việc dọn dẹp là đầy đủ, ta phải dùng phương thức hủy bỏ ở lớp cha là virtual

Tóm tắt về bài học Đa hình

❖Phương thức thuần ảo

- ❑Chỉ có khai báo, không có thân hàm
- ❑Có ý nghĩa cho việc tổ chức sơ đồ phân cấp các lớp, nó đóng vai trò “chừa sẵn chỗ trống” cho các lớp con điền vào với phiên bản phù hợp
- ❑Ví dụ: virtual void Draw() = 0;

❖Lớp cơ sở trừu tượng

- ❑Lớp cơ sở trừu tượng là lớp cơ sở không có đối tượng nào thuộc chính nó
- ❑Khi lớp có chứa phương thức thuần ảo, lớp trở thành lớp cơ sở trừu tượng, không thể tạo đối tượng thuộc lớp cơ sở trừu tượng