LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương 7: Đa hình (p1)

Trình bày: ThS. Lê Thanh Trọng

NỘI DUNG

1. Giới thiệu
2. Vùng chọn kiểu

NỘI DUNG

1. Giới thiệu
2. Vùng chọn kiểu

Giới thiệu

- 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
- Ví dụ, Phương thức tính diện tích của lớp hình, hình tam giác, tứ giác,…

Giới thiệu

- Đa hình: Là hiện tượng các đối tượng thuộc các lớp khác nhau có khả năng hiểu cùng một thông điệp theo các cách khác nhau
- Ví dụ: Nhận được cùng một thông điệp “nhảy”, một con kangaroo và một con cóc nhảy theo hai kiểu khác nhau: chúng cùng có hành vi “nhảy” nhưng các hành vi này có nội dung khác nhau

Bài toán

- Giả sử, cần quản lý danh sách các đối tượng có kiểu có thể khác nhau →Cần giải quyết 2 vấn đề:
  - Cách lưu trữ
  - Thao tác xử lý
- Xét trường hợp cụ thể, các đối tượng có thể là Người, Sinh viên hoặc Công nhân

Bài toán

- Về mặt lưu trữ:
  - Có thể dùng mảng
  - Danh sách liên kết
  - …

Có hai cách để giải quyết vấn đề:
- Vùng chọn kiểu
- Phương thức ảo

- Về thao tác: Phải thõa yêu cầu đa hình, thao tác có hoạt động khác nhau ứng với các loại đối tượng khác nhau

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";}
    void Xuat() const {
            cout << "Nguoi, ho ten: " << HoTen << " sinh “
            cout << NamSinh; }
};
```

Ví dụ

```
class SinhVien : public Nguoi{
protected:
    char *MaSo;
public:
    SinhVien(char *n, char *ms, int ns) : Nguoi(n,ns) {
            MaSo = strdup(ms);
    }
    ~SinhVien() { delete [ ] MaSo;}
    void Xuat() const {
            cout<<"Sinh vien "<<HoTen<<", ma so "<<MaSo;
    }
};
```

Ví dụ

```
class NuSinh : public SinhVien
{
  public:
    NuSinh( char *ht, char *ms, int ns) : SinhVien(ht,ms,ns) {
    }
    void Xuat() const
    {
          cout << HoTen
          cout << " ma so " << MaSo << " an 2 to pho";
    }
};
```

Ví dụ

```
class CongNhan : public Nguoi{
protected:
    double MucLuong;
public:
    CongNhan( char *n, double ml, int ns) : Nguoi(n,ns), MucLuong(ml){ }
    void Xuat() const {
            cout << "Cong nhan, ten " << HoTen
            cout << " muc luong: " << MucLuong;
    }
};
```

Ví dụ

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

Ví dụ

```
const int N = 4;
void main(){
    Nguoi *a[N];
    a[0] = new SinhVien(“Vien Van Sinh”, “200001234”, 1982);
    a[1] = new NuSinh(“Le Thi Ha Dong”, “200001235”, 1984);
    a[2] = new CongNhan(“Tran Nhan Cong”, 1000000, 1984);
    a[3] = new Nguoi(“Nguyen Thanh Nhan”, 1960);
    XuatDs(4,a);
}
```

```
Nguoi, ho ten: Vien Van Sinh sinh 1982
Nguoi, ho ten: Le Thi Ha Dong sinh 1984
Nguoi, ho ten: Tran Nhan Cong sinh 1984
Nguoi, ho ten: Nguyen Thanh Nhan sinh 1960
```

NỘI DUNG

1. Giới thiệu
2. Vùng chọn kiểu

Dùng vùng chọn kiểu

- Để bảo đảm xuất liệu tương ứng với đối tượng, phải có cách nhận diện đối tượng
  - Ta 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
- Các đối tượng thuộc lớp người có cùng giá trị cho vùng chọn kiểu, các đối tượng thuộc lớp sinh viên có giá trị của vùng chọn kiểu khác của lớp người

Dùng vùng chọn kiểu – Ví dụ

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

Dùng vùng chọn kiểu – Ví dụ

```
class SinhVien : public Nguoi{
protected:
    char *MaSo;
public:
    SinhVien(char *n, char *ms, int ns) :        Nguoi(n,ns) {
            MaSo = strdup(ms); pl = SV;
    }
    ~SinhVien() {delete [ ] MaSo;}
    void Xuat() const {
            cout<<"Sinh vien "<<HoTen<<", ma so " << MaSo;
    }
};
```

Dùng vùng chọn kiểu – Ví dụ

```
class CongNhan : public Nguoi{
protected:
    double MucLuong;
public:
    CongNhan( char *n, double ml, int ns) : Nguoi(n,ns), MucLuong(ml){
          pl = CN;
    }
    void Xuat() const{
          cout << "Cong nhan, ten " << HoTen
         cout << " muc luong: " << MucLuong;
    }
};
```

Dùng vùng chọn kiểu – Ví dụ

```
void XuatDs(int n, Nguoi *an[]) {
    for (int i = 0; i < n; i++){
        switch(an[i]->pl){
            case Nguoi::SV:
                ((SinhVien *)an[i])→Xuat();  break;
            case Nguoi::CN:
                ((CongNhan *)an[i])→Xuat(); break;
            default:
                an[i]->Xuat(); break;
        }
        cout << "\n";
    }
}
```

Dùng vùng chọn kiểu – Ví dụ

```
const int N = 4;
void main(){
    Nguoi *a[N];
    a[0] = new SinhVien(“Vien Van Sinh”, “200001234”, 1982);
    a[1] = new NuSinh(“Le Thi Ha Dong”, “200001235”, 1984);
    a[2] = new CongNhan(“Tran Nhan Cong”, 1000000, 1984);
    a[3] = new Nguoi(“Nguyen Thanh Nhan”, 1960);
    XuatDs(4,a);
}
```

```
Sinh vien Vien Van Sinh, ma so 200001234
Sinh vien Le Thi Ha Dong, ma so 200001235
Cong nhan, ten Tran Nhan Cong muc luong:1000000
Nguoi, ho ten: Nguyen Thanh Nhan sinh 1960
```

Dùng vùng chọn kiểu

- Cách tiếp cận trên giải quyết được vấn đề: Lưu trữ các đối tượng khác kiểu nhau và thao tác khác nhau tương ứng từng đối tượng. Tuy nhiên, tồn tại một số khuyết điểm:
  - Mã lệnh dài dòng, thủ công (nhiều switch case)
  - Dễ sai sót, khó sửa
  - Khó nâng cấp, bảo trì
- Các nhược điểm trên có thể khắc phục được nhờ phương thức ảo