Bài 4: Kế thừa

Nội dung

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

Ôn lại

- Quá tải hàm
- Quá tải phương thức
- Quá tải toán tử
- Hàm bạn (friend function)

3

Quan hệ bao gồm

- Các đối tượng có thể có quan hệ với nhau.
- Quan hệ "**bao gồm**" (**has a)**: A **"has a"** B nếu trong đối tượng A có một thành phần là đối tượng B (*quan hệ bao gồm - composition)*
  - Ví dụ: Đối tượng Ngôi nhà có thành phần là đối tượng mái nhà, Tường và Cửa ra vào. Đối tượng Hình chữ nhật có thành phần là Điểm trên trái và dưới phải. Đối tượng Sinh viên có một thành phần là đối tượng Ngày tháng (ngày sinh)...

4

Quan hệ kế thừa

- Quan hệ **"is a"**: A **"is a"** B nếu đối tượng A có những đặc tính của đối tượng B (*quan hệ kế thừa - inheritance)*
  - Ví dụ: Con voi có bản chất là một Động vật. Hình cầu là một sự mở rộng của Hình tròn. Sinh viên đại cương và Sinh viên chuyên ngành đều có những đặc điểm của Sinh viên...
- **Chú ý:** Trong lập trình đôi khi ta có thể cài đặt quan hệ giữa các đối tượng một cách "linh hoạt". Chẳng hạn có thể coi Hình tròn là đối tượng chứa một Điểm (tâm hình tròn) hoặc là một sự kế thừa từ đối tượng Điểm (!?).

5

Ví dụ về quan hệ bao gồm

- Xây dựng lớp Hinhchunhat từ lớp Diem
- #include <iostream.h>

```
// Lop Diem
class Diem
{
private:
        int x, y;
public:
        Diem(); // Cau tu mac dinh
        Diem(int xx, int yy);  // Cau tu khoi tao toa do
        void DatToado(int xx, int yy);  // Dat toa do
        int LayX();  // Tra ve toa do x
        int LayY();  // Tra ve toa do y
};
```

6

Ví dụ về quan hệ bao gồm

```
// Cai dat phuong thuc
Diem::Diem()
{
    DatToado(0, 0);
}
Diem::Diem(int xx, int yy)
{
    DatToado(xx, yy);
}
void Diem::DatToado(int xx, int yy)
{
    x = xx;
    y = yy;
}
int Diem::LayX() { return x; }
int Diem::LayY() { return y; }
```

7

Ví dụ về quan hệ bao gồm

```
// Lop Hinhchunhat
class Hinhchunhat
{
private:
    Diem tt, dp;
public:
    Hinhchunhat(); // Cau tu mac dinh
    Hinhchunhat(int xtt, int ytt, int xdp, int ydp); // Khoi tao toa do
    void HienThongtin(); // Hien thi thong tin
};
```

8

Ví dụ về quan hệ bao gồm

```
Hinhchunhat::Hinhchunhat() : tt(0,0), dp(0,0)
{ }

Hinhchunhat::Hinhchunhat(int xtt, int ytt, int xdp, int ydp) :
        tt(xtt, ytt), dp(xdp, ydp)
{ }

void Hinhchunhat::HienThongtin()
{
    cout<<"Toi la hinh chu nhat"<<endl;
    cout<<"Toa do diem tren trai la:["
          <<tt.LayX()<<","<<tt.LayY()<<"]"<<endl;
    cout<<"Toa do diem duoi phai la: ["
          <<dp.LayX()<<","<<dp.LayY()<<"]"<<endl;
}
```

9

Ví dụ về quan hệ bao gồm

```
// main
int main()
{
    Hinhchunhat  hcna;
    Hinhchunhat  hcnb(1,1,20,10);
    hcna.HienThongtin();
    hcnb.HienThongtin();
   return 0;
}
```

10

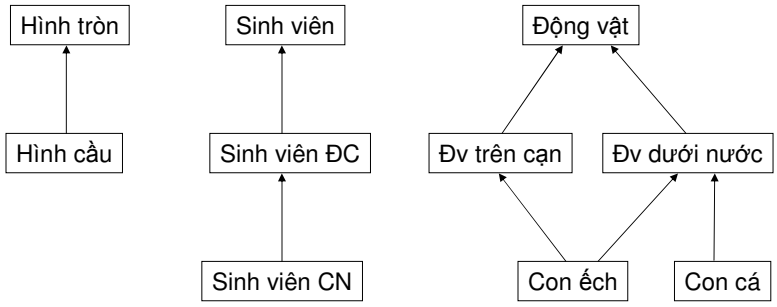
Quan hệ bao gồm

- Chú ý:
  - Cấu tử của Diem sẽ được gọi trước cấu tử của Hinhchunhat
  - Huỷ tử của Hinhchunhat sẽ được gọi trước huỷ tử của Diem.
- Nhận xét:
  - Quan hệ composition là rất "tự nhiên".
  - Vì class Diem cũng là một kiểu dữ liệu (do người dùng tự định nghĩa) cho nên việc một đối tượng thuộc kiểu Diem có trong Hinhchunhat cũng giống như một dữ liệu kiểu int có trong một lớp nào đó.

Kế thừa (Inheritance)

- Kế thừa là một trong những đặc điểm quan trọng của OOP, nó cho phép xây dựng những lớp mới dựa trên những lớp đã có sẵn.
  - Lớp mới được gọi là lớp dẫn xuất (derived class)
  - Lớp đã có được gọi là lớp cơ sở (base class)
- Có thể bổ sung các thành phần dữ liệu và các phương thức mới
- Từ một lớp cơ sở ta có thể xây dựng nhiều lớp dẫn xuất

Kế thừa (Inheritance)



13

Kế thừa (Inheritance)

- Một lớp dẫn xuất có thể sử dụng nhiều lớp cơ sở khác nhau
- Một lớp có thể là lớp dẫn xuất của lớp này vừa là lớp cơ sở đối với lớp khác
- Từ cơ cấu kế thừa này, hình thành cấu trúc hình cây của các lớp
  - Các lớp cơ sở còn được gọi là lớp cha
  - Các lớp dẫn xuất được gọi là lớp con

14

Kế thừa (Inheritance)

- **Kế thừa đơn**: một lớp mới được dẫn xuất từ duy nhất một lớp cơ sở
- **Đa kế thừa:** Lớp mới được xây dựng từ sự kế thừa 2 hoặc nhiều lớp cơ sở

15

Kế thừa đơn

- Ví dụ: Xây dựng lớp Hinhcau kế thừa từ lớp Hinhtron
- *// lớp Hinhtron*
- class Hinhtron
- {
- private:
- int bankinh;
- public:
- Hinhtron();
- Hinhtron(int bk);
- void DatBankinh(int bk);
- int LayBankinh();
- int LayDuongkinh();
- float LayDientich();
- };
- *Giả sử các phương thức đã được định nghĩa*

16

Kế thừa đơn

```
// lớp Hinhcau
class Hinhcau : public Hinhtron
{
public:
    Hinhcau();
    Hinhcau(int bk);
    float LayThetich();
};
```

Lớp Hình cầu kế thừa dữ liệu (bankinh) và các phương thức từ lớp Hình tròn.

Lớp Hình cầu định nghĩa thêm các phương thức của riêng nó.

17

Kế thừa đơn

```
Hinhcau::Hinhcau() : Hinhtron(0)
{
}

Hinhcau::Hinhcau(int bk) : Hinhtron(bk)
{
}

float Hinhcau::LayThetich()
{
    int r = LayBankinh();
    return (float(4)/3)*3.14*r*r*r;
}
```

Gọi cấu tử của lớp cha

Gọi phương thức của lớp cha

18

Kế thừa đơn

```
// test main 1
int main()
{
        Hinhcau hc;
        hc.DatBankinh(3);
        cout<<"Duong kinh hinh cau = "<<hc.LayDuongkinh();
        cout<<"The tich hinh cau = "<<hc.LayThetich();
        Return 0;
}
```

Gọi phương thức của lớp cha

Gọi phương thức đã định nghĩa thêm

Kế thừa đơn

```
// test main 2
int main()
{
        Hinhcau hc(3);
        Hinhtron ht = hc;
        cout<<"Ban kinh hinh tron = "<<ht.LayBankinh();
        Return 0;
}
```

Gán đối tượng con cho đối tượng cha -> hợp lệ

- Chú ý về gán đối tượng:
  - Gán đối tượng con cho đối tượng cha chỉ đơn thuần là việc sao chép dữ liệu, đối tượng cha không thể sử dụng các phương thức của đối tượng con
  - Gán đối tượng cha cho đối tượng con là không hợp lệ

Kế thừa đơn

- Lớp con không kế thừa cấu tử và huỷ tử của lớp cha
- Thứ tự gọi cấu tử và huỷ tử:
  - Cấu tử của lớp cha gọi trước rồi đến cấu tử lớp con
  - Huỷ tử của lớp con gọi trước rồi đến huỷ tử lớp cha

21

Nạp chồng (overriding)

- Làm thế nào để tính diện tích (bề mặt) của hình cầu ? Ta gọi *hc.LayDientich()* ?
  - Trả lời: *hc.LayDientich* sẽ trả lại giá trị diện tích của hình tròn -> kết quả không đúng !
  - Giải pháp 1: Định nghĩa thêm một phương thức LayDientichHC trong lớp Hinhcau
  - Giải pháp 2: Định nghĩa nạp chồng (override) phương thức LayDientich trong lớp Hinhcau
- Nạp chồng là việc định nghĩa lại một phương thức của lớp cha trong lớp con.

22

Nạp chồng (overriding)

```
// lớp Hinhcau
class Hinhcau : public Hinhtron
{
public:
        ...
        float LayDientich();
};

float Hinhcau::LayDientich()
{
        int r = LayBankinh(); // phuong thuc LayBankinh la cua Hinhtron
        return 4*3.14*r*r;
}
```


Nạp chồng (overriding)

```
// test main 3
int main()
{
        Hinhcau hc(2);
        cout<<"Dien tich hinh cau = "<<hc.LayDientich();
        return 0;
}
```

Vì đã nạp chồng LayDientich nên kết quả in ra sẽ là diện tích (bề mặt) của hình cầu.

Toán tử phạm vi

- Làm thế nào nếu ta muốn tính diện tích của mặt cắt qua tâm hình cầu (chính là diện tích hình tròn tương ứng) ?
- Trả lời: Sử dụng toán tử phạm vi **::**

```
// test main 4
int main()
{
        Hinhcau hc(2);
        cout<<"Dien tich hinh cau = "<<hc.LayDientich();
        cout<<"Dien tich mat cat = "<<hc.Hinhtron::LayDientich();
        return 0;
}
```

25

Từ khoá protected

- Trong phương thức LayDientich và LayThetich của lớp Hinhcau, để lấy lại giá trị bán kính ta phải dùng LayBankinh() vì bankinh được khai báo là private.
- Nếu khai báo bankinh là public thì sẽ vi phạm tính che dấu dữ liệu.
- -> Giải pháp: Dùng từ khoá protected
- Từ khoá **protected** cho phép dữ liệu/phương thức của lớp cha có thể được truy nhập từ lớp con (mà không được truy nhập từ các lớp khác).

26

Từ khoá protected

```
// lớp Hinhtron
class Hinhtron
{
protected:
        int bankinh;
public:
        ...
};

float Hinhcau::LayThetich()
{ return (float(4)/3)*3.14*bankinh*bankinh*bankinh; }

float Hinhcau::LayDientich()
{ return 4*3.14*bankinh*bankinh; }
```

bankinh sẽ có thể được truy nhập từ lớp con của Hinhtron

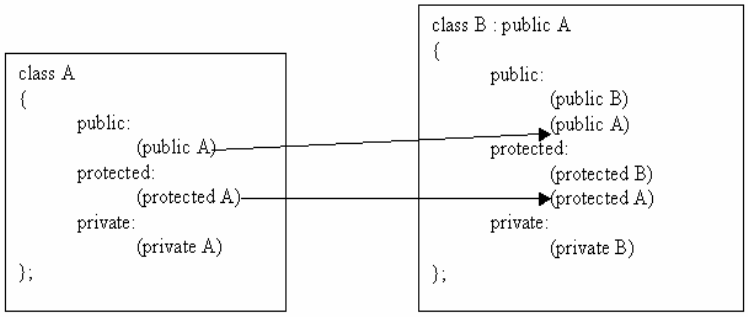
27

Các kiểu kế thừa

- Kế thừa kiểu public
- Kế thừa kiểu protected
- Kế thừa kiểu private

28

Kế thừa kiểu Public

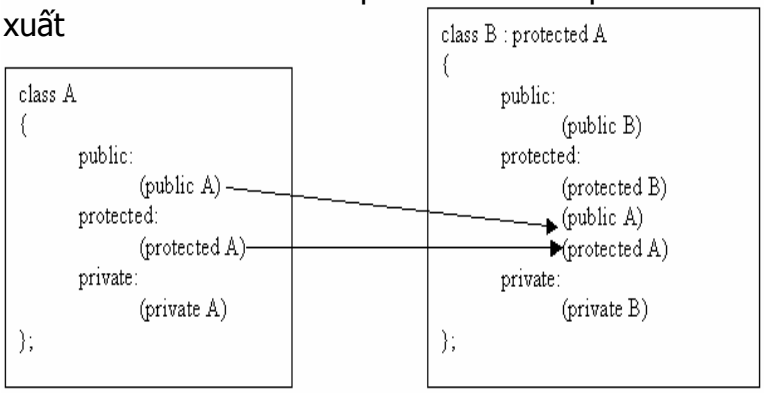


Kế thừa kiểu Public

- Kế thừa kiểu Public
  - Các thành viên public của lớp cơ sở trở thành các thành viên public của lớp dẫn xuất
  - Các thành viên protected của lớp cơ sở trở thành các thành viên protected của lớp dẫn xuất.
  - Các thành viên private của lớp cơ sở không bao giờ được truy cập trực tiếp từ một lớp dẫn xuất.
- Ví dụ (hình vẽ): lớp dẫn xuất B không thể truy cập đến các thành phần private trong lớp cơ sở A

Kế thừa kiểu Protected

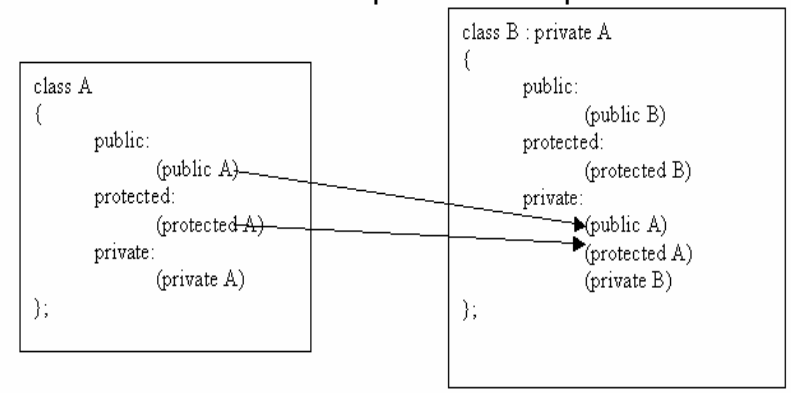
Các thành viên public và protected của lớp cơ sở trở thành các thành viên protected của lớp dẫn xuất



31

Kế thừa kiểu Private

- Các thành viên public và protected của lớp cơ sở trở thành các thành viên private của lớp dẫn xuất



32

Đa kế thừa

- Đa kế thừa có 2 loại:
  - Đa kế thừa một phần (partial multiple inheritance)
  - Đa kế thừa (multiple inheritance)
- Ví dụ:
  - Lớp *GvCohuu* kế thừa toàn phần lớp *Giaovien*
  - Lớp *GvCohuu* kế thừa một phần lớp *Canbo*

33

Đa kế thừa

- Cú pháp khai báo đa kế thừa: Ghi tên các lớp cơ sở cùng với kiểu kế thừa cách nhau bởi dấu phẩy
- Ví dụ:

```
class HinhVuong:public HCN, public Hinhthoi{
....
};
class GvCohuu:public Canbo, public Giaovien{
....
};
```

34

Đa kế thừa

- Các thành phần của một đối tượng đa kế thừa
  - Một số thành phần của các lớp cơ sở
  - Các thành phần mới
- Các vấn đề phát sinh:
  - Các lớp cơ sở có các hàm ảo hay dữ liệu trùng tên
  - Các cấu tử (hàm tạo) của lớp cơ sở khởi tạo biến cho lớp dẫn xuất
  - Các lớp cơ sở được dẫn xuất từ cùng một lớp

35

Đa kế thừa

- Cấu tử trong các đối tượng đa kế thừa:
  - Một đối tượng của một lớp dẫn xuất từ nhiều lớp cơ sở
  - Các cấu tử của các lớp cơ sở được gọi lần lượt theo đúng thứ tự khi khai báo lớp dẫn xuất
  - Cấu tử của lớp dẫn xuất được gọi

36

Đa kế thừa

- Gọi phương thức trùng tên của các lớp cơ sở: Ghi rõ tên lớp trong lời gọi
- Ví dụ:

```
Hinhvuong hv;
hv.HCN::Dtich();
//hoặc hv.Hinhthoi::Dtich();
```

37

Kế thừa ảo (Virtual Inheritance)

- Các lớp cơ sở có thể dẫn xuất từ cùng một lớp
- Để tránh lãng phí bộ nhớ ta có thể sử dụng khai báo kế thừa ảo
- Cú pháp: Thêm từ khoá Virtual trước phần khai báo kế thừa
- Ví dụ:
  - `class HCN: virtual public Tugiac`
  - `class Hinhthoi: virtual public Tugiac`

38

Ôn lại

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

39

Bài tập về nhà

- Cài đặt đầy đủ lớp Điểm và lớp Hinhchunhat. Thêm cấu tử khởi tạo Hinhchunhat từ hai điểm và cấu tử sao chép. Thêm phương thức tính giao của 2 hình chữ nhật, kiểm tra một điểm có nằm trong hình chữ nhật hay không (quá tải phương thức này).
- Tạo **lớp Hình cầu** kế thừa từ lớp Hình tròn. Nạp chồng phương thức tính diện tích.
- Tạo **lớp Hình trụ** (Cylinder) kế thừa từ lớp Hình tròn. Nạp chồng phương thức tính
- diện tích.
- Tạo **lớp Hình vuông** kế thừa từ lớp Hình chữ nhật

40