Kỹ thuật lập trình hướng đối tượng với C++

1

Chương 5 - Kỹ thuật thừa kế

5.1. Khái niệm

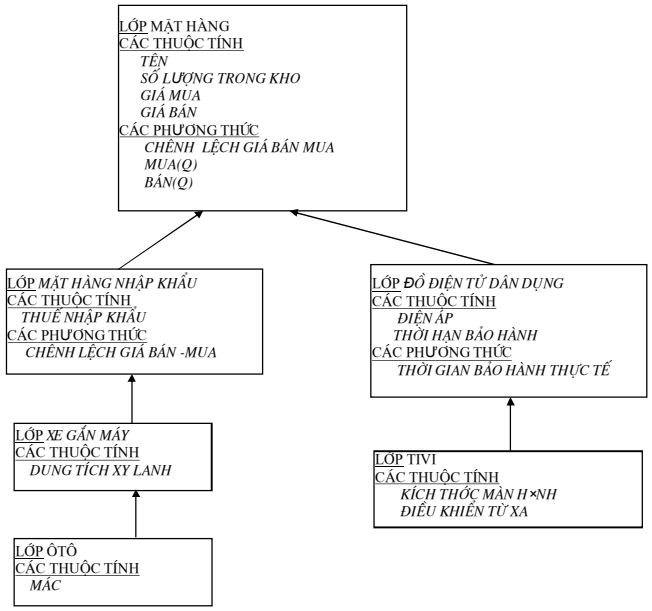
Kế thừa là khả năng cho phép xây dựng một lớp mới (**lớp dẫn xuất**) thừa hưởng các thành phần từ một hay nhiều lớp đã có (**lớp cơ sở**). Trong lớp con (**lớp dẫn xuất**) ta có thể **bổ sung thêm** các thành phần hoặc **định nghĩa lại** các thành phần.

Ví dụ 1: Lớp hình Chữ nhật có chiều rộng, chiều dài, diện tích, chu vi. Lớp hình Vuông kế thừa lớp hình Chữ nhật nhưng có chiều dài và chiều rộng bằng nhau.

Ví dụ 2: Lớp PhanSo1 {ts, ms, nhập, in, tối giản}

Lớp PhanSo2 kế thừa lớp PhanSo1 thêm các phương thức cộng, trừ, nhân, chia: Lớp PhanSo2 {ts, ms, nhập, in, tối giản, cộng, trừ, nhân, chia }

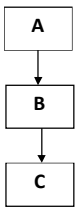
2

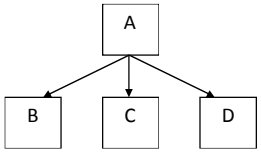


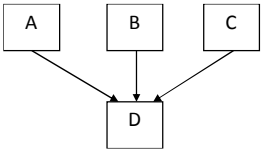
VÍ DỤ VỀ SƠ ĐỒ THỪA KẾ

3

Kế thừa tạo ra mô hình phân cấp:

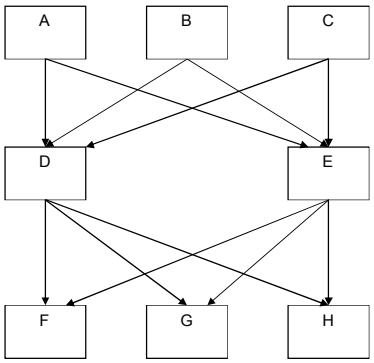






Kế thừa đơn: chỉ có một lớp cơ sở

Đa kế thừa: có nhiều hơn một lớp cơ sở



4

5.2. Xây dựng lớp dẫn xuất:

Cú pháp:

```
class  <tên lớp con> :      [kiểu dẫn xuất] <tên lớp cha1> ,
                            [kiểu dẫn xuất] <tên lớp cha 2> , …
{
    // Các thành phần của lớp con
};
```

Trong đó:

Kiểu dẫn xuất có thể là: public, private (ngầm định), protected.

public: tất cả các thành phần public của lớp cha sẽ là pubic ở lớp con

private: tất cả các thành phần public của lớp cha sẽ là private ở lớp con

protected: tất cả các thành phần protected và public của lớp cha sẽ là protected ở lớp con

5

Ví dụ: xây dựng lớp **Chu_Nhat**, lớp **Vuong** kế thừa lớp **Chu_Nhat**:

```
#include <iostream.h>
#include <conio.h>
class   Chu_Nhat
{
  protected:
        float  dai, rong;
  public:
        Chu_Nhat (float  d = 0, float  r=0)
                { dai = d; rong = r;  }
        float   Dien_tich()  {return  dai*rong;}
};
```

6

```
class  Vuong : public Chu_Nhat
{
    public:
        Vuong(float   size) { dai = rong = size; };
};
//---------------------
void   main()
{
  Chu_Nhat    cn(7,4); //Khai bao chu nhat voi kich thuoc 7 va 4
  Vuong   v(8);  // Khai bao hinh vuong voi kich thuoc la 8
  cout << "Hinh chu nhat: Dien tich =  " << cn.Dien_tich() ;
  cout << "\nHinh vuong: Dien tich =  " << v.Dien_tich();
  getch();
}
```

7

5.3. Quyền truy xuất

✓ Các thành phần **private** của lớp cở sở không cho phép truy nhập trong lớp dẫn xuất.

✓ Các thành phần **public** của lớp cơ sở có thể truy nhập bất kỳ chỗ nào trong chương trình. Như vậy trong các lớp dẫn xuất có thể truy nhập được tới các thành phần này.

✓ Các thành phần khai báo là **protected** có phạm vi truy nhập rộng hơn so với các thành phần private, nhưng hẹp hơn so với các thành phần public. Các thành phần **protected** của một lớp chỉ được mở rộng phạm vi truy nhập cho các lớp dẫn xuất trực tiếp từ lớp này.

	Trong cùng lớp	Từ lớp kế thừa	Từ ngoài lớp
Private	+	–	–
Protected	+	+	–
Public	+	+	+

– : không thể truy cập
+ : có thể truy cập

8

(1) Quyền truy xuất các thành phần ở lớp cơ sở (lớp cha)
(2) Kiểu dẫn xuất

(1) / (2)	private	protected	public
private	private	private	private
protected	private	protected	protected
public	private	protected	public

Quyền truy xuất ở lớp con

9

Ví dụ :

✓ Giả sử lớp A có:
Thuộc tính public a1
Thuộc tính protected a2

✓ Lớp B dẫn xuất public từ A, thì A::a1 trở thành public trong B, A::a2 trở thành protected trong B.

✓ Nếu dùng B làm lớp cở để xây dựng lớp C. Thì trong C có thể truy nhập tới A::a1 và A::a2.

✓ Nếu sửa đổi để B dẫn xuất private từ A, thì cả A::a1 và A::a2 trơ thành private trong B, và khi đó trong C không được phép truy nhập tới các thuộc tính A::a1 và A::a2.

10

Chú ý: Có thể gán 1 đối tượng của lớp con vào một đối tượng của lớp cha.

Ví dụ: Lớp cơ sở (lớp cha) PhanSo1, lớp dẫn xuất (lớp con) PhanSo2 kế thừa lớp PhanSo1.

```
PhanSo1  a;
PhanSo2  b;
a = b;        // gán được
b = a;        // lỗi: Cannot convert 'Phanso1' to 'Phanso2'
```

Khi gán, các thành phần thừa (không có trong lớp cha) sẽ bị cắt tỉa và chuyển đổi kiểu lên an toàn.

11

5.4. Định nghĩa lại quyền truy xuất

Để định nghĩa lại chỉ cần liệt kê thành phần đó sau từ khoá quyền truy xuất tương ứng

<quyền truy xuất>: <tên lớp cha>::<tên thành phần>;

Ví dụ:

```
class A
{   private:  f1, f2 ;
    protected:  f3, f4 ;
    public:  f5, f6 ;
};
class  B :  A
{    public:
            A::f6 ;
};
```

Kết quả: f1, f2, f3, f4, f5 là private, còn f6 là public

12

Chú ý:

- Khi định nghĩa lại quyền truy xuất với 1 thành phần thì mọi thành phần cùng tên cũng bị tác động

- Chỉ có thể định lại quyền truy xuất theo đúng quyền của thành phần đó trong lớp cha

- Nếu trong lớp cơ sở có nhiều thành phần cùng tên nhưng khác quyền truy xuất thì không thể định nghĩa lại

- Nếu lớp con có một thành phần cùng tên thì thành phần của lớp con sẽ che phủ thành phần lớp cha, muốn truy xuất phải viết tường minh

13

5.5. Hàm tạo và hàm huỷ

a. Hàm tạo

- Hàm tạo của lớp cha không được kế thừa

- Mỗi đối tượng của lớp con có thể coi là một đối tượng của lớp cha. Do đó: khi gọi hàm tạo của lớp con sẽ kéo theo gọi hàm khởi tạo của lớp cha

Thứ tự gọi:

Hàm tạo lớp cha → Hàm tạo lớp con

- Nếu xây dựng hàm tạo của lớp con: Phải gọi hàm tạo của lớp cha tường minh

Cú pháp:

<hàm tạo dẫn xuất>([th/số]):<hàm tạo cơ sở> ([th/số])

{ }

14

VD:

```
class  sophuc
{   protected:
        double   thuc, ao;
    public:
        sophuc()
            { thuc = 0; ao = 0; }
  .....
} ;
class  sophuc1 : public  sophuc
{   public:
        sophuc1() : sophuc() { };
.....
} ;
```

15

Chú ý:

- Hàm tạo lớp cơ sở thực hiện trước
- Nếu lớp dẫn xuất có nhiều lớp cơ sở thì trình tự thực hiện tuân theo trình tự kế thừa.

b. Hàm huỷ

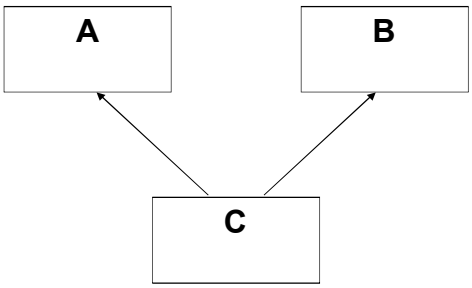
- Hàm huỷ của lớp cơ sở không được kế thừa
- Các hàm huỷ được thi hành theo trình tự ngược lại so với hàm khởi tạo
- Hàm huỷ của lớp dẫn xuất thi hành trước hàm huỷ của lớp cơ sở

5.6. Đa kế thừa

- Đa kế thừa là khả năng xây dựng lớp dẫn xuất kế thừa từ nhiều hơn một lớp cơ sở
- Đa kế thừa có thể là tính năng rất mạnh nhưng đôi khi gây ra một số vấn đề.

16

VD:



Khai báo lớp C nh sau:

```
class  C: public   A, public   B
{
……….
};
```

Bên trong lớp C có thể khai báo các thành phần dữ liệu và các phương thức của lớp C.

17

<u>Thứ tự gọi các hàm tạo nh sau:</u>

Các hàm tạo của các lớp cơ sở theo thứ tự khai báo của các lớp cơ sở trong lớp dẫn xuất đợc gọi trớc và sau cùng là hàm tạo của lớp dẫn xuất mới đợc gọi.

Nh ví dụ trên, hàm tạo của lớp A đợc gọi trớc, sau đó là hàm tạo của lớp B và cuối cùng là hàm tạo của lớp C đợc gọi.

<u>Thứ tự gọi các hàm hủy nh sau:</u>

Các hàm hủy đợc gọi theo thứ tự ngợc lại với cách gọi các hàm tạo.

Trong ví dụ trên, hàm hủy của lớp C sẽ đợc gọi đầu tiên rồi đến hàm hủy của lớp B đợc gọi và cuối cùng là hàm hủy của lớp A đợc gọi.

<u>Các thuộc tính thừa kế:</u>

Đa thừa kế cũng có tính chất thừa kế nh kiểu thừa kế đơn.

18

Cách gọi các hàm thành phần của các lớp cơ sở:

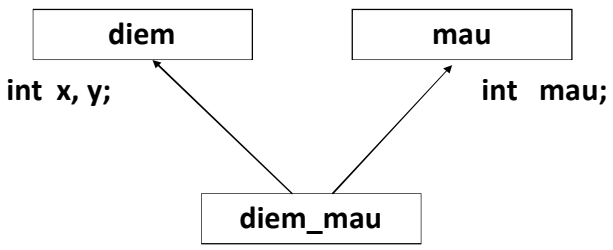
Ví dụ: Lớp C thừa kế từ lớp A và từ lớp B.

Nếu lớp A có hàm hienthi( ), lớp B có hàm hienthi( ), thì ở lớp C ta sử dụng hàm hienthi ( ) của lớp nào thì phải chỉ rõ phạm vi hàm đó :

```
void  C:: hienthi( )
{
    .....
    A::hienthi( ); // sử dụng hàm thành phần của lớp cơ sở A
    .....
    B::hienthi( ); // sử dụng hàm thành phần của lớp cơ sở B
    .....
}
```

19

VD: Xây dựng lớp diem_mau kế thừa lớp diem và lớp mau :



20

```
#include <conio.h>
#include <iostream.h>
class  diem
{   private:
        int   x, y;
    public:
        diem()
                { x=0;  y=0; }
        diem(int  xd, int  yd )
                { x=xd;  y=yd; }
        void  hienthi()
                { cout<<"\nDiem ("<<x<<", "<<y<<")"; }
};
```

21

```
class  mau
{       int  m;
    public:
        mau()
                { m = 0; }
        mau (int  md)
                { m = md; }
        void  hienthi()
                { cout<<"\nMau :"<< m; }
};
```

22

```
class  diem_mau : public  diem, public  mau
{   public :
        diem_mau() : diem(), mau() { } ;
        diem_mau ( int xd,  int yd, int  md ):
                diem(xd, yd), mau(md)
                { }
        void  hienthi()
                {
                        diem  :: hienthi();
                        mau  :: hienthi();
                }
};
```

23

```
void   main()
{
   diem_mau  A(3,4,5);
   cout<<"\nGoi phuong thuc hienthi() cua lop diem_mau:";
   A.hienthi();
   cout<<"\nGoi phuong thuc hienthi() cua lop diem:";
   A.diem::hienthi();
   cout<<"\nGoi phuong thuc hienthi() cua lop mau:";
   A.mau::hienthi();
   getch();
}
```

24

Bài tập chương 5

1/ Xây dựng lớp sophuc gồm phần thực, phần ảo, phương thức: nhập, in.

Xây dựng lớp sophuc1 kế thừa lớp sophuc, bổ sung các phép +, -

Hàm main: Nhập 2 số phức Y, Z. Tính và in Y + Z, Y – Z.

2/Cài đặt lớp PS1 gồm có:

Dữ liệu: tử số, mẫu số

Phương thức: nhập phân số (mẫu khác 0), in phân số, tối giản.

Cài đặt lớp PS2 kế thừa PS1 và bổ sung phép +, -, *, các phân số

Chương trình chính: nhập 2 phân số, thông báo các kết quả tính toán và so sánh.

25

3/ Cài đặt lớp người NGUOI gồm có:

- **Dữ liệu: họ tên, mã số, lương**
- **Phương thức: nhập, in**

Cài đặt lớp người trong biên chế BC kế thừa lớp NGUOI và bổ sung:

Dữ liệu: hệ số lương, phụ cấp

Phương thức: định nghĩa lại phương thức nhập và tính lương.

Cài đặt lớp người làm hợp đồng HD kế thừa lớp NGUOI và bổ sung:

Dữ liệu: tiền công lao động, số ngày làm việc trong tháng, hệ số vượt giờ.

Phương thức: định nghĩa lại phương thức nhập và tính lương.

Chương trình chính: nhập mảng các n người (n < 100), in ra danh sách này.

26