5.3. Tính đa hình

Tính đa hình là khả năng để cho một thông báo có thể thay đổi cách thực hiện của nó theo **lớp cụ thể của đối tượng nhận thông báo. Nghĩa là**

- Khi một lớp dẫn xuất được tạo ra, nó có thể thay đổi cách thực hiện các phương thức nào đó mà nó thừa hưởng từ lớp cơ sở. **Một thông báo khi được gởi đến một đối tượng của lớp cơ sở, sẽ dùng phương thức đã định nghĩa cho nó trong lớp cơ sở.**

- Nếu một lớp dẫn xuất **định nghĩa lại** một phương thức thừa hưởng từ lớp cơ sở thì **một thông báo có cùng tên với phương thức này, khi được gởi tới một đối tượng của lớp dẫn xuất sẽ gọi phương thức đã định nghĩa cho lớp dẫn xuất.**

- Để thực hiện được tính đa hình, C++ dùng cơ chế **liên kết động** (dynamic binding) thay cho cơ chế **liên kết tĩnh** (static binding).
- Cơ chế liên kết tĩnh thực hiện khi chương trình biên dịch và được dùng trong các ngôn ngữ cổ điển như C, Pascal, …
- C++ sử khái niệm *phương thức ảo* để biểu diễn cho cơ chế liên kết động.

1. Liên kết tĩnh

Xem ví dụ sau: Giả sử có 2 lớp A, B và C được xây dựng như sau:

```
class A
{
  public:
          void xuat()
           {
             cout <<"\n Lop A";
           }
       };
   class B : public A
   {
        public:
          void xuat()
           {
              cout <<"\n Lop B";
            }
       };
```

Cả 2 lớp này đều có hàm thành phần là xuat(). Lớp B có lớp cơ sở là A và B kế thừa các phương thức của A. Do đó một đối tượng của B sẽ có 2 hàm xuat(). Xem các câu lệnh sau:

```
B  ob; //  ob là đối tượng của lớp con B
ob.xuat();       // Gọi tới hàm thành phần xuat() của lớp B
ob.A::xuat() ;  // Gọi tới hàm thành phần xuat() của lớp A
```

Các lời gọi hàm thành phần trong ví dụ trên đều xuất phát từ đối tượng ob và mọi lời gọi đều *xác định rõ* hàm cần gọi.

Ta xét tiếp tình huống các lời gọi không phải từ một biến đối tượng mà từ một con trỏ đối tượng. Xét các câu lệnh:

```
A   *p;   // p  là các con trỏ kiểu A
A  a;                  // a là đối tượng kiểu A
B  b;                  // b là đối tượng kiểu B
```

Bởi vì con trỏ của lớp cơ sở có thể dùng để chứa địa chỉ các đối tượng của lớp dẫn xuất, nên cả 3 phép gán sau đều hợp lệ:

```
p = &a;  //p trỏ vào đối tượng lớp A
```

Ta xét lời gọi hàm thành phần từ các con trỏ p: p->xuat(); //goi ham lop A

```
p = &b;  //p trỏ vào đối tượng lớp B
p->xuat()  //vẫn gọi hàm xuat() của lớp A
```

Cả 2 câu lệnh trên đều gọi tới hàm thành phần xuat() của lớp A, bởi vì con trỏ p có kiểu lớp A. Sở dĩ như vậy là vì một lời gọi (xuất phát từ một đối tượng hay con trỏ) tới hàm thành phần **luôn luôn liên kết với một hàm thành phần cố định** và sự liên kết này xác định trong quá trình biên dịch chương trình. Ta bảo đây là **sự liên kết tĩnh**.

Có thể tóm lược cách thức gọi các phương thức theo cơ chế **liên kết tĩnh** như sau:

1. Nếu lời gọi xuất phát từ một đối tượng của lớp nào đó, thì phương thức của lớp đó sẽ được gọi.
2. Nếu lời gọi xuất phát từ một con trỏ kiểu lớp, thì phương thức của lớp đó sẽ được gọi bất kể con trỏ chứa địa chỉ của đối tượng nào.

Vấn đề đặt ra: Ta muốn một lời gọi phương thức từ con trỏ sẽ phụ thuộc vào lớp mà con trỏ đang trỏ đến đối tượng của lớp đó.
Ví dụ: khi ta cho p trỏ đến đối tượng của lớp B và thực hiện lời gọi p->xuat(); thì phải gọi đến hàm xuất của lớp B.
Điều này sẽ thực hiện được gọi là sự liên kết động và C++ dung phương thức ảo.

2. Phương thức ảo

Phương thức ảo được định nghĩa *trong lớp cơ sở* và được định nghĩa lại trong lớp dẫn xuất.

Để định nghĩa phương thức ảo thì phần khai báo phải bắt đầu bằng từ khóa ***virtual***. Khi một lớp có chứa phương thức ảo được kế thừa, lớp dẫn xuất sẽ định nghĩa lại phương thức ảo đó cho chính mình.

Trong phần định nghĩa lại phương thức ảo ở lớp dẫn xuất, không cần phải sử dụng lại từ khóa virtual.

3. Quy tắc gọi phương thức ảo

Phương thức ảo chỉ khác phương thức thông thường khi được gọi từ một **con trỏ của lớp cơ sở.**

Lời gọi tới phương thức ảo từ một **con trỏ sẽ *phụ thuộc vào đối tượng cụ thể* mà con trỏ đang trỏ tới: *con trỏ đang trỏ tới đối tượng của lớp nào thì phương thức của lớp đó sẽ được gọi*.**

```
//vi du ve lien ket dong – Phương thức ảo
#include <iostream>
using namespace std;
class A
  {
    public:
            virtual void xuat()  //ham xuat la Phuong thuc ảo
             {
               cout <<"\n Lop A";
             }
  };
class B : public A
            {
              public:
                void xuat()  //định nghĩa lại phương thức ảo
                 {
                    cout <<"\n Lop B";
                 }
    };
```

```
int main()
{
        A *p;  //con tro cua lop cơ so
        A a;
        B b;
        p =&a; //p tro den doi tuong lop cha A
        p->xuat();  //goi hàm lop A, in ra Lop A

        p=&b;  //p tro den doi tuong lop con B
        p->xuat(); //goi hàm lop B, in ra : Lop B vì xuat() la phuong thuc ao

        return 0;

}
```

4. Một số ví dụ:

Ví dụ 1: Xét ví dụ 5.13, ta dung phương thức ảo Khenthuong

```
#include <iostream>
#include <stdio.h>
using namespace std;
class Nguoi
{
    char hoten[25], ma[10];
  public :
    void nhap()
                {
                  cout<<"\n Nhap ma: ";   fflush(stdin);
                  gets(ma);
                 cout<<"\n Ho ten: ";    gets(hoten);
                }
                void xuat()
                {
                        cout<<"\n Ma: "<<ma;
                        cout<<"\n Ho ten: "<<hoten;
                }
                virtual int khenthuong()    //khenthuong la ham ao va se duoc dinh nghia lai o lop con
                { return 0;}
};
```

```
class Sinhvien : public Nguoi
{    float dtb;
  public :
     void nhap_SV()
     { cout<<"\n Nhap diem trung binh: ";
       fflush(stdin);
       cin>>dtb;
     }
     void xuat_SV()
    {
       cout<<"\n Diem trung binh: "<<dtb;
     }
     int khenthuong()  //dinh nghia lai ham ao cho lop Sinhvien
    {
          return dtb > 8;
       }
};
```

```
class Giaovien : public Nguoi
{    float sbb;
  public :
     void nhap_GV()
     {
           cout<<"\nNhap so bai bao: ";
          cin>>sbb;
        }
     void xuat_GV()
     {
        cout<<"\n So bai bao: "<<sbb;
        }
     int khenthuong()  //dinh nghia lai ham ao cho lop Giaovien
    {
       return sbb > 1;
     }
};
```

```
int main()
{
  Nguoi *p; //p la con tro cua lop cha (Nguoi)
  Sinhvien sv[100];
  Giaovien gv[20];
  //nhap du lieu cho Sinh vien
  int m,n;
  cout<<"\n Nhap so sinh vien: ";
  cin>>n;
  for(int i=0;i<n;++i)
   {
    sv[i].nhap();
    sv[i].nhap_SV();
    }
  cout<<"\n Danh sach SV duoc khen thuong :";
  for(int i=0;i<n;++i)
   {  p = &sv[i]; //p tro vao doi tuong thi i cua lop sinhvien
      if (p->khenthuong())  //goi ham khenthuong cua lop Sinhvien
       { sv[i].xuat();
         sv[i].xuat_SV();
       }
   }
```

```
//nhap du lieu cho giao vien
  cout<<"\n Nhap so giao vien : ";
  cin>>m;
  for(int i=0;i<m;++i)
   {
     gv[i].nhap();
     gv[i].nhap_GV();
   }
  cout<<"\n Danh sach GV duoc khen thuong :";
  for(int i=0;i<n;++i)
   { p = &gv[i];  //pp tro vao doi tuong thu i cua lop Giaovien
     if (p->khenthuong()) //goi ham khenthuong cua lop Giaovien
       { gv[i].xuat();
         gv[i].xuat_GV();
       }
   }
  return 0;
}
```

Ví dụ 2:

Danh mục trong thư viện bao gồm hai loại: *bài báo*, *sách*. Mỗi danh mục có ***nhan đề, tác giả, số lần được tham khảo*** và với mỗi loại danh mục ta có thêm các thông tin khác nhau:

- Bài báo có tên tạp chí đăng bài báo, số tạp chí.
- Sách thì có nhà xuất bản và version (lần in).

a. Hãy vẽ cấu trúc phân cấp các lớp và xây dựng các lớp đó (bao gồm các thuộc tính và các phương thức cần thiết để có thể làm được câu b).

b. Viết chương trình nhập vào danh sách có n (n<=100) phần tử gồm 2 loại danh mục trên, sau đó:

- In ra màn hình danh mục có số lượng tham khảo lớn nhất.
- In ra màn hình các tác giả của các danh mục được thưởng biết rằng điều kiện để được thưởng với bài báo cần trên 10 tham khảo, sách trên 20 tham khảo.

Chú ý: Dùng phương thức ảo

```
//vi du 2 ve phuong thuc ao (BT 5.11)
#include <iostream>
using namespace std;
class danhmuc
{
 private:
        char nhande[30];  //nhan de sach hoac tap chi
        char tacgia[30];
 protected:
        int tk;  //so lan tham khao
 public:
        void nhap();
        void xuat();
        int get_tk();
        char * get_tg();
        virtual int khenthuong()  //ham ao
        {
                return 0;
        }
};
```

```
void danhmuc::nhap()
{
        cout<<"Nhap nhan de: "; cin.ignore(1);   cin.get(nhande,30);
        cout<<"Nhap tac gia: "; cin.ignore(1); cin.get(tacgia,50);
        cout<<"Nhap so lan tham khao: "; cin>>tk;
}
void danhmuc::xuat()
{
        cout<<"Nhan de: "<<nhande<<" \n";
        cout<<"Tac gia: "<<tacgia<<"\n";
        cout<<"So lan tham khao: "<<tk;
}
int danhmuc::get_tk()
{       return tk; }

char * danhmuc::get_tg()
{       return tacgia; }
```

```
class sach: public danhmuc
{
        char nxb[50];
        int  version;
 public:
        void nhap_sach();
        void xuat_sach();
        int khenthuong();
};
void sach::nhap_sach()
{
        nhap();
        cout<<"Nha xuat ban: "; cin.ignore(1);cin.get(nxb,50);
        cout<<"Lan in thu: ";
        cin>>version;
}
void sach::xuat_sach()
{
        xuat();
        cout<<"Nha xuat ban: "<<nxb;
        cout<<"Lan in thu: "<<version;
}
int sach::khenthuong()
{       return tk > 20; }
```

```
class tapchi: public danhmuc
{
        char ten_tc[50];
        int  so_tc;
public:
        void nhap_tc();
        void xuat_tc();
        int khenthuong();
};
void tapchi::nhap_tc()
{
        nhap();
        cout<<" Ten tap chi: "; cin.ignore(1); cin.get(ten_tc,50);
        cout<<" So tap chi: "; cin>>so_tc;
}
void tapchi::xuat_tc()
{
        xuat();
        cout<<"Ten tap chi: "<<ten_tc;
        cout<<"So tap chi: "<<so_tc;
}
int tapchi::khenthuong()
{   return tk>10; }
```

```
int main()
{
	danhmuc *DS[100];
	int i,n,tl;
	cout<<"Nhap so luong n (bao gom sach va tap chi = ";
	cin>>n;
	for (i=0; i<n; i++)
	{
		cout<<"Nhap loai danh muc: 1- sach, 2-tap chi: ";
		cin>>tl;
		if (tl==1)
		{
			sach *p=new sach;
			p->nhap_sach();
			DS[i]=p;
		}
		else
		{
			tapchi *p=new tapchi;
			p->nhap_tc();
			DS[i]=p;
		}
	}
```

```
        int max=DS[0]->get_tk();
         //Tim max cua tham khao
         for (i=0; i<n; i++)
                        if (max<DS[i]->get_tk())
                                        max=DS[i]->get_tk();
         //In danh muc co tham khao la max
         cout<<"\n Danh muc (sach hoac tap chi) co tham khao max: "<<endl;
         for (i=0; i<n; i++)
                        if (max==DS[i]->get_tk())
                                        DS[i]->xuat();
         //In tac gia co danh muc duoc khen thuong
         cout<<"\n Danh sach tac gia duoc khen thuong: "<<endl;
         for (i=0; i<n; i++)
                        if (DS[i]->khenthuong()==1)
                          cout<<DS[i]->get_tg()<<endl;
         return 0;
}
```

Thảo luận:

Xét Ví du1 (ví dụ 5.13 trong phần 1)

1. Chuyển các biến chuỗi là đối tượng của lớp string.
2. Thay đổi câu hỏi ở ví dụ 5.13: Hãy nhập vào một danh sách không quá 100 đối tượng bao gồm sinh viên và giáo viên. In ra danh sách các sinh viên, giáo viên được khen thưởng.

BÀI TẬP

- Xem giáo trình