

# Interface và Đa hình

GV. Nguyễn Minh Huy



- Khái niệm interface.
- Hàm ảo và liên kết động.
- Phương thức hủy ảo.



- **Khái niệm interface.**
- Hàm ảo và liên kết động.
- Phương thức hủy ảo.

# Khái niệm interface



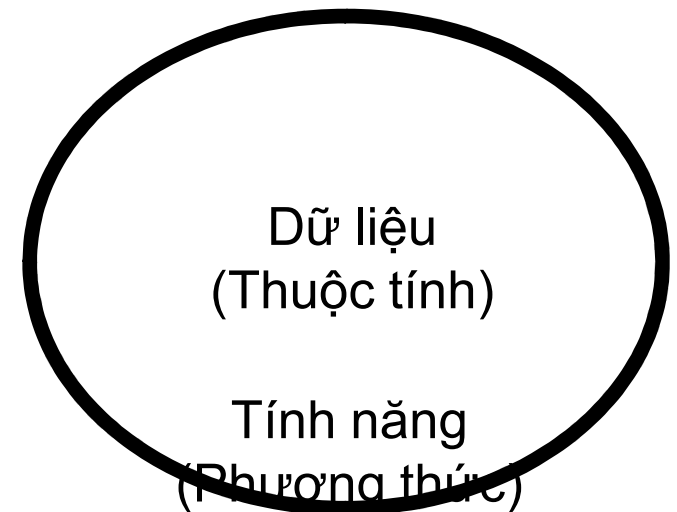
## ■ Giao tiếp giữa đối tượng và bên ngoài:

### ■ Quy tắc hộp đen:

- Thuộc tính: private, hạn chế truy xuất.
- Phương thức: public, cung cấp tính năng.

➔ Đối tượng giao tiếp qua phương thức.

➔ Phần khai báo public ➔ INTERFACE.



# Khái niệm interface



## ■ Đặc điểm Interface:

### ■ Là “phần vỏ” của lớp.

Lớp = **Interface** + Khai báo private + Cài đặt.

### ■ Không chứa cài đặt.

### ■ Giao thức giao tiếp của đối tượng.

Nhận interface PhanSo

```
class PhanSo
{
private:
    int m_tu;
    int m_mau;
public:
    PhanSo( int tu, int mau );
    PhanSo rutGon( );
    PhanSo nghichDao( );
};
```

```
void xuLy( PhanSo p )
{
    // Xử lý trên p...
}

void main()
{
    PhanSo p1( 1, 2 );
    PhanSo p2( 1, 3 );
    xuLy( p1 );
    xuLy( p2 );
}
```



## ■ Interface trong kế thừa:

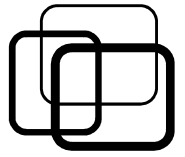
### ■ Lớp kế thừa:

- Thừa hưởng thuộc tính/phương thức từ lớp cơ sở.
- ➔ Thừa hưởng interface của lớp cơ sở.
- ➔ Có cùng giao thức giao tiếp của lớp cơ sở.

### ■ Tính đa hình trong kế thừa:

- Hàm nhận đối tượng lớp cơ sở
  - ➔ Cũng nhận đối tượng lớp kế thừa.
- Con trỏ kiểu lớp cơ sở
  - ➔ Có thể trỏ đến đối tượng lớp kế thừa.

# Khái niệm interface



## ■ Interface trong kế thừa:

```
class Animal
{
public:
    void talk() { cout << "Don't talk"; }
};
```

```
class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo"; }
};
```

```
class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau"; }
};
```

Chấp nhận interface **Animal**

```
void xuLy(Animal p)
{
    p.talk();
}

void main()
{
```

```
    Animal a;
    Cat c;
    Dog d;
```

```
    xuLy(a);
```

```
    xuLy(c);
```

```
    xuLy(d);
```

```
    Animal *p;
```

```
    p = &a;
```

```
    p = &c;
```

```
    p = &d;
```

```
}
```

Chấp nhận  
các đối  
tượng có  
interface  
**Animal**



- Khái niệm interface.
- **Hàm ảo và liên kết động.**
- Phương thức hủy ảo.



# Hàm ảo và liên kết động



## ■ Vấn đề liên kết tĩnh trong C++:

```
class Animal
{
public:
    void talk() { cout << "Don't talk"; }
};
```

```
class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo"; }
};
```

```
class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau"; }
};
```

```
void xuLy(Animal p)
{
    p.talk();
}
void main()
{
```

```
    Cat    c;
    Dog    d;
    xuLy(c);
    xuLy(d);
```

```
    Animal *p;
    p = &c;
    p->talk();
    p = &d;
    p->talk();
```

```
}
```

Liên kết  
cài đặt  
Animal  
khi dịch

Liên kết  
cài đặt  
Animal  
khi dịch



## ■ Khái niệm hàm ảo:

### ■ Hàm bình thường:

- Lời gọi hàm gắn với cài đặt hàm lúc dịch → Liên kết tĩnh.

### ■ Hàm ảo:

- Hàm đặc biệt.
- **Lời gọi hàm chỉ gắn với interface.**
- Bỏ ngỏ phần cài đặt → Liên kết động
  - ➔ Được gắn động vào lúc chạy.
  - ➔ Tùy thuộc đối tượng lúc chạy.

### ■ Trong C++:

- Khai báo: **virtual** <Chữ ký hàm>;
- **Chỉ dùng được với con trỏ đối tượng.**

# Hàm ảo và liên kết động



## ■ Liên kết động trong C++:

```
class Animal
{
public:
    virtual void talk() { cout << "Don't talk"; }
};
```

```
class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo"; }
};
```

```
class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau"; }
};
```

```
void xuLy(Animal *p)
{
    p->talk();
}
void main()
{
```

```
    Cat    c;
    Dog    d;
    xuLy(&c);
    xuLy(&d);
```

```
    Animal *p;
    p = &c;
    p->talk();
    p = &d;
    p->talk();
}
```

Liên kết  
interface  
Animal,  
Bỏ ngõ  
cài đặt

Liên kết  
interface  
Animal,  
Bỏ ngõ  
cài đặt

# Hàm ảo và liên kết động



## ■ Xây dựng interface trong C++:

- C++ cài đặt interface bằng “hàm thuần ảo”.

- Đặc điểm “hàm thuần ảo”:

- Hàm ảo, có thêm “= 0” ở cuối khai báo.
- Chỉ có khai báo không có cài đặt.
- Là “phần vỏ” để liên kết động.
- Bỏ ngỏ phần cài đặt cho lớp kế thừa.

➔ Lớp có chứa hàm thuần ảo là lớp trừu tượng.

➔ Không thể tạo đối tượng từ lớp trừu tượng..

```
class Animal
{
public:
    virtual void talk() = 0;
};
```

Hàm thuần ảo, không có cài đặt!!

# Hàm ảo và liên kết động



## ■ Xây dựng interface trong C++:

```
class Animal
{
public:
    virtual void talk() = 0;
};

class Cat: public Animal
{
public:
    void talk() { cout << "Meo meo"; }
};

class Dog: public Animal
{
public:
    void talk() { cout << "Gau gau"; }
};
```

Lớp trừu tượng

```
void xuLy(Animal *p)
{
    p->talk();
}

void main()
{
    Cat    c;
    Dog    d;
    xuLy(&c);
    xuLy(&d);

    Animal *p;
    p = new Animal; // Sai
    p = new Cat;    // Đúng
    p->talk();
}
```

Liên kết interface Animal, Bỏ ngoặc cài đặt



- Ý nghĩa của liên kết động:
  - Giao tiếp thông qua interface.
  - Cài đặt thay đổi động, tùy nhu cầu người dùng.
- ➔ Mã nguồn tổng quát.

```
void giveATalk( Animal *p )  
{  
    p->talk( );  
}
```

```
void giveATalk( int type )  
{  
    if ( type == 0 )  
    {  
        Cat c;  
        c.talk( );  
    }  
    else if ( type == 1 )  
    {  
        Dog d;  
        d.talk( );  
    }  
}
```



- Khái niệm interface.
- Hàm ảo và liên kết động.
- **Phương thức hủy ảo.**

# Phương thức hủy ảo



## ■ Xét ví dụ:

```
class GiaoVien
{
private:
    char    *m_hoTen;
public:
    ~GiaoVien() { delete m_hoTen; }
};

class GVCN : public GiaoVien
{
private:
    char    *m_lopCN;
public:
    ~GVCN() { delete m_lopCN; }
};
```

```
void main()
{
    GVCN    *p1 = new GVCN;
    delete p1;
```

~**GVCN**()  
~**GiaoVien**()

```
GiaoVien *p2 = new GVCN;
delete p2;
```

~**GiaoVien**()

Vì sao thứ tự gọi  
phương thức hủy  
khác nhau??



# Phương thức hủy ảo



## ■ Dr. Guru khuyên:

### ■ Phương thức hủy của lớp phải luôn là hàm ảo.

➔ Liên kết động đến phương thức hủy lớp kế thừa.

```
class GiaoVien
{
public:
    virtual ~GiaoVien() { delete m_hoTen; }
};
```

```
class GVCN : public GiaoVien
{
public:
    ~GVCN() { delete m_lopCN; }
};
```

```
GiaoVien *p3 = new GVCN;
```

```
delete p3;
```

~GVCN()  
~GiaoVien()





## ■ Khái niệm interface:

- Lớp = Interface + Khai báo private + Cài đặt.
- Lớp kế thừa thừa hưởng interface lớp cơ sở.
- Giả lập interface trong C++: hàm thuần ảo.

## ■ Hàm ảo:

- Chỉ ràng buộc với interface, không ràng buộc cài đặt.
- Khai báo dùng “**virtual**”.

## ■ Phương thức hủy ảo:

- Phương thức hủy phải luôn là hàm ảo.



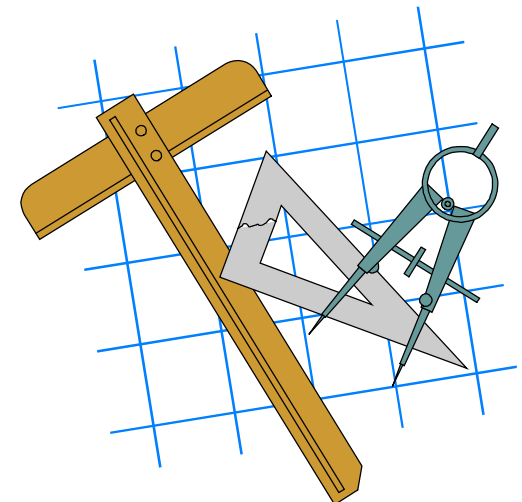


## ■ Bài tập 8.1:

```
class A {  
public:  
[yyy] void f1() { cout << "Good morning.\n"; f2(); }  
[zzz] void f2() { cout << "Good afternoon.\n"; }  
};  
class B: public A {  
public:  
    void f1() { cout << "Good evening.\n"; f2(); }  
    void f2() { cout << "Good night.\n"; }  
};  
void main()  
{  
    A *p = new B;  
    p->f1();  
}
```

Cho biết những gì xuất hiện trên màn hình trong các trường hợp:

- a) [yyy] trống, [zzz] trống.
- b) [yyy] trống, [zzz] virtual.
- c) [yyy] virtual, [zzz] trống.
- d) [yyy] virtual, [zzz] virtual.





## ■ Bài tập 8.2:

Cho 2 loại hình:

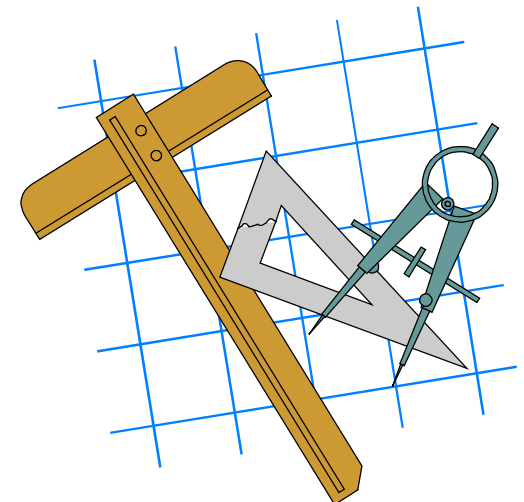
- Hình tam giác: biểu diễn bởi 3 đỉnh.
- Hình chữ nhật: biểu diễn bởi 2 điểm - trên trái và dưới phải.

Viết hàm nhận vào danh sách 2 loại hình trên và xuất thông tin các hình có trong danh sách.

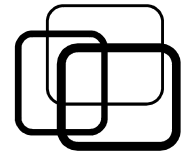
Nếu thêm vào loại hình mới là hình tròn.

- Hình tròn: biểu diễn bởi tâm và bán kính.

Khi đó, chương trình sẽ thay đổi như thế nào?



# Bài tập



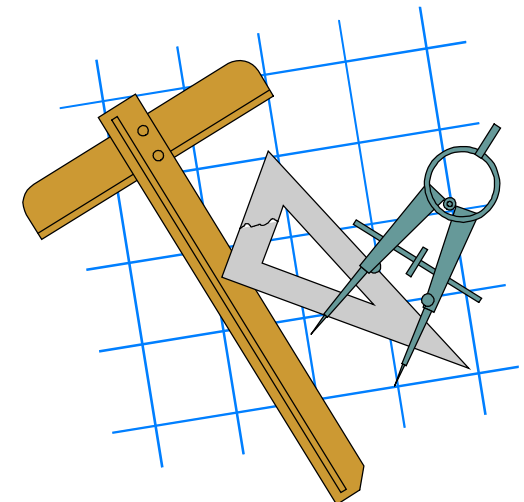
## ■ Bài tập 8.3:

Cho bảng tốc độ chạy của các động vật như sau:

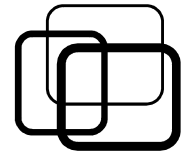
Động vật	Tốc độ
Báo	100km/h
Linh dương	80km/h
Sư tử	70km/h
Chó	60km/h
Người	30km/h

Viết hàm nhận vào 2 động vật trong bảng trên và so sánh tốc độ chạy giữa chúng.

Nếu thêm vào con ngựa chạy 60km/h, khi đó chương trình sẽ thay đổi như thế nào?



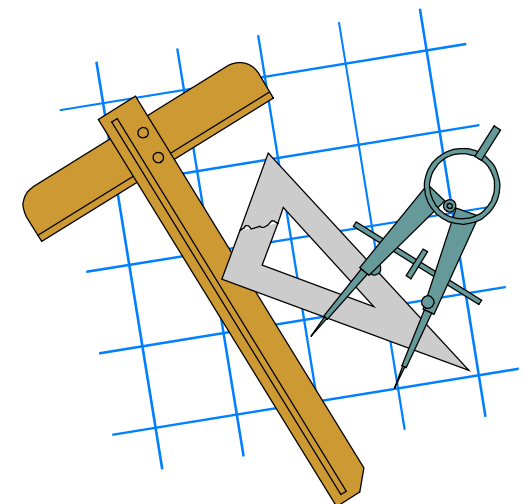
# Bài tập



## ■ Bài tập 8.4:

Cho bảng cách thức hoạt động của các loài động vật như sau:

	<b>Ăn</b>	<b>Di chuyển</b>	<b>Sinh sản</b>
<b>Cá mập</b>	Tạp	Bơi	Đẻ trứng
<b>Cá chép</b>	Phiêu sinh	Bơi	Đẻ trứng
<b>Sư tử</b>	Tạp	Chạy	Đẻ con
<b>Bò</b>	Cỏ	Chạy	Đẻ con
<b>Cá voi</b>	Phiêu sinh	Bơi	Đẻ con
<b>Chim sẻ</b>	Sâu bọ	Bay	Đẻ trứng
<b>Đại bàng</b>	Tạp	Bay	Đẻ trứng
<b>Cá sấu</b>	Tạp	Bò, Bơi	Đẻ trứng
<b>Tắc kè</b>	Tạp	Bò	Đẻ trứng
<b>Dơi</b>	Tạp	Bay	Đẻ con





## ■ Bài tập 8.4:

Hãy xây dựng các interface **X**, **Y**, **Z**, **U**, **V** và áp dụng cho những động vật trong bảng để chúng có thể tham gia vào những hoạt động sau:

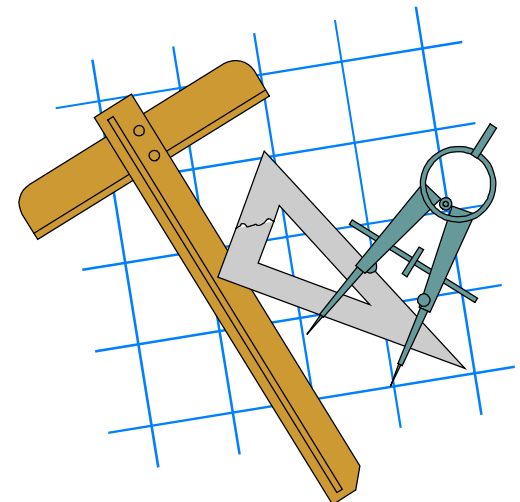
```
void thiBoi(X dv1, X dv2)
{
    dv1.boi();
    dv2.boi();
}
```

```
void thiBay(Y dv1, Y dv2)
{
    dv1.bay();
    dv2.bay();
}
```

```
void thuanHoaThu(Z dv)
{
    Z con = dv.deCon();
}
```

```
void nuoiCa(U dv)
{
    dv.boi();
    U con = dv.deTrung();
}
```

```
void nuoiBoSat(V dv)
{
    dv.bo();
    dv.anTap();
    V con = dv.deTrung();
}
```





## ■ Bài tập 8.5 (\*):

Cho lớp Line và Rectangle dùng vẽ đường thẳng và hình chữ nhật:

```
class Line
{
private:
    Point m_p1;
    Point m_p2;
public:
    Line(Point, Point);
    void drawLine();
};
```

Hãy viết hàm vẽ đường thẳng và hình chữ nhật từ một danh sách hình cho trước.

Yêu cầu:

- Sử dụng (không chỉnh sửa) lớp Line và Rectangle.
- Hàm vẽ phải không đổi khi thêm hình mới vào.

```
class Rectangle
{
private:
    Point m_p1;
    Point m_p2;
public:
    Rectangle(Point, Point);
    void drawRect();
};
```

