Lập trình hướng đối tượng và C++

Bài 8: Tương ứng bội

TS. Nguyễn Hiếu Cường
Bộ môn CNPM, Khoa CNTT, Trường Đại học GTVT
Email: cuonggt@gmail.com

Hà Nội, 2020

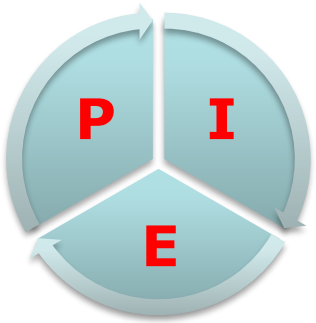
Nội dung chính

1. Giới thiệu môn học
2. Các khái niệm cơ bản
3. Hàm trong C++
4. Lớp và đối tượng
5. Định nghĩa chồng toán tử
6. Hàm tạo và hàm huỷ
7. Dẫn xuất và thừa kế
8. **Tương ứng bội**
9. Khuôn hình

Khái niệm tương ứng bội

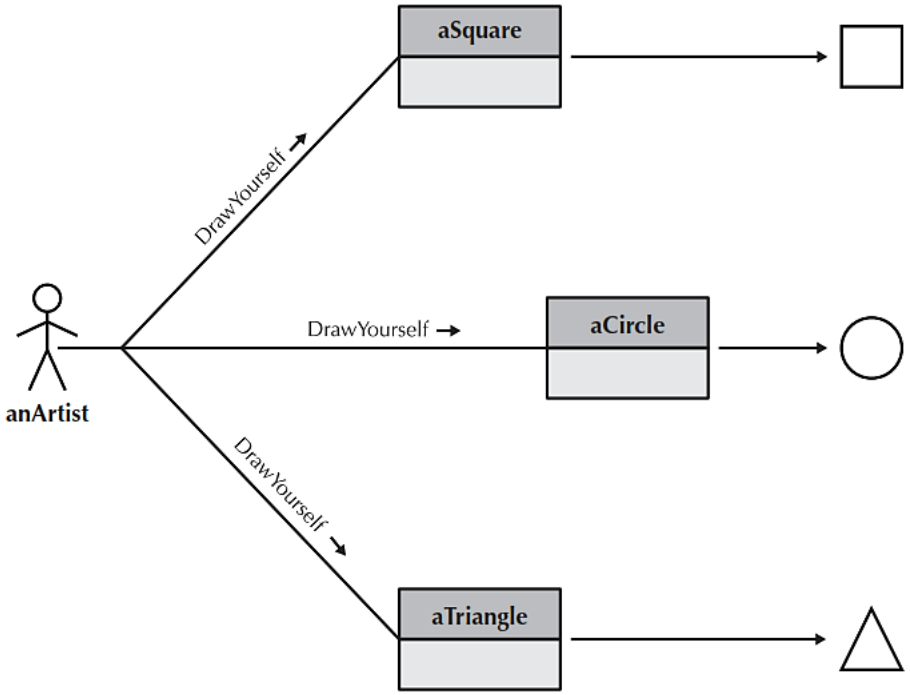
- Tương ứng bội (Polymorphism)
  - Còn gọi là **"Đa hình"**

  - Là một trong các "trụ cột" của OOP
    - Polymorphism
    - Inheritance
    - Encapsulation



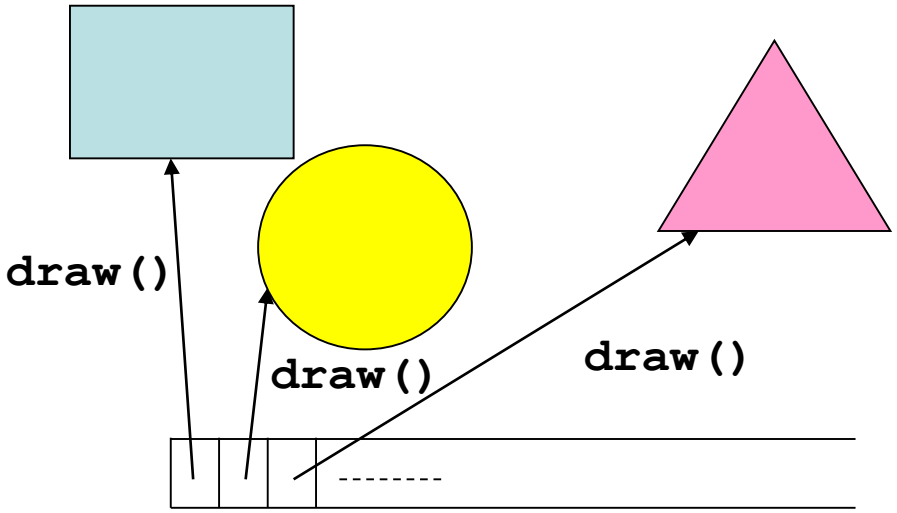
Khái niệm về tương ứng bội

- Xử lý các đối tượng của các lớp có liên quan theo một cách chung



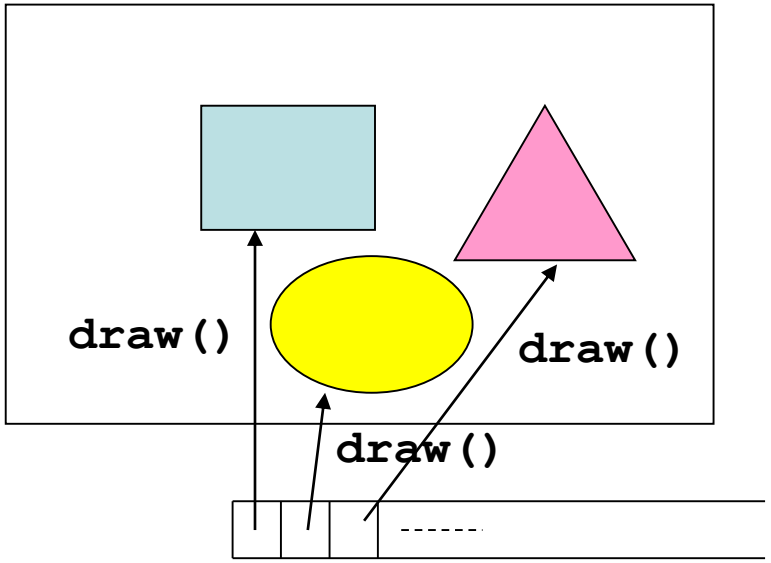
Tác dụng của tương ứng bội

- "Một giao diện, nhiều cài đặt"
- Chương trình đơn giản, rõ ràng và dễ bảo trì hơn
- Dễ dàng phát triển chương trình mà không cần sửa đổi nhiều



Ví dụ

```
Shape* shapes[10];
...
for (i = 0; i < numShapes; i++)
{
    ...
    shapes[i] -> draw();
}
```



Phương thức tĩnh và phương thức động

- Phương thức tĩnh
  - Thông điệp truyền tới một đối tượng thì phương thức của lớp (mà đối tượng đó được khai báo) sẽ được thực hiện
  - Liên kết tĩnh = liên kết sớm (early binding): compile time
- Phương thức động (phương thức ảo)
  - Thông điệp truyền tới một đối tượng thì phương thức của lớp (tương ứng với đối tượng đó) sẽ được thực hiện
  - Liên kết động = liên kết muộn (late binding): run time

Phương thức động

- Phương thức động còn gọi là phương thức ảo (virtual function)
- Tên phương thức ảo phải hoàn toàn giống nhau ở tất cả các lớp (trong cùng hệ thống phân cấp lớp)
- Phương thức ảo được định nghĩa như phương thức thông thường nhưng thêm từ khóa virtual ở phía trước

```
virtual void draw();
```

Ví dụ (phương thức ảo)

```
class Shape {
public:
  virtual void draw()
  {
    cout<<"draw shape \n";
  }
  void paint()
    {
    cout<<"paint shape \n";
  }
};
class Square: public Shape {
public:
  void draw(){cout<<"draw square \n";}
  void paint(){cout<<"paint square"; }
};
class Circle: public Shape {
public:
  void draw() {cout<<"draw circle";}
  void paint() {cout<<"paint circle";}
};
```

```
int main()
{
  Shape *ptr;

  Circle c;
  Square s;
  ptr= &c;        // ptr chứa đc &c ?
  ptr->draw();
  ptr->paint();

  ptr= &s;
  ptr->draw();
  ptr->paint();
}
```

Có gì khác nhau giữa **draw()** và **paint()** ?

Lớp cơ sở trừu tượng

- Phương thức ảo thuần túy
  - Dùng trong trường hợp "chung chung"

```
virtual void draw() = 0;
```

- Lớp cơ sở trừu tượng
  - Là lớp trong đó có ít nhất một phương thức ảo thuần túy

```
class Shape
{
public:
  virtual void draw()=0;
  void paint();
};
```

  - Lớp cơ sở trừu tượng không có đối tượng!

Tóm tắt

- Tương ứng bội (tính đa hình)
  - Ý nghĩa của tương ứng bội?
  - Phương thức ảo?
  - Liên kết sớm
  - Liên kết muộn?

Bài tập (xác định kết quả)

```
class A  {
  int a;
public:
  A() {a = 5;}
  void xuat()  {cout<<a;}
};
class B: public A  {
  int a;
public:
  B() { a = 1;}
  void xuat() {cout<<a; }
};
void main()  {
  A *ob,x;
  B y;
  ob=&x; ob->xuat();
  ob=&y; ob->xuat();
}
```

```
class A  {
  int a;
public:
  A() {a=5;}
  virtual void xuat()  {cout<<a;}
};
class B: public A  {
  int a;
public:
  B() {a=1; }
  void xuat() {cout<<a; }
};
void main()  {
  A *ob,x;
  B y;
  ob=&x; ob->xuat();
  ob=&y; ob->xuat();
}
```

1. Virtual functions allow you to

   a. create an array of type pointer-to-base class that can hold pointers to derived classes.

   b. create functions that can never be accessed.

   c. group objects of different classes so they can all be accessed by the same function code.

   d. use the same function call to execute member functions of objects from different classes.

2. True or false: A pointer to a base class can point to objects of a derived class.

3. If there is a pointer `p` to objects of a base class, and it contains the address of an object of a derived class, and both classes contain a nonvirtual member function, `ding()`, then the statement `p->ding();` will cause the version of `ding()` in the _____ class to be executed.

4. Write a declarator for a virtual function called `dang()` that returns type `void` and takes one argument of type `int`.

5. Deciding—after a program starts to execute—what function will be executed by a particular function call statement is called _____.

6. If there is a pointer, `p`, to objects of a base class, and it contains the address of an object of a derived class, and both classes contain a virtual member function, `ding()`, the statement `p->ding();` will cause the version of `ding()` in the _____ class to be executed.

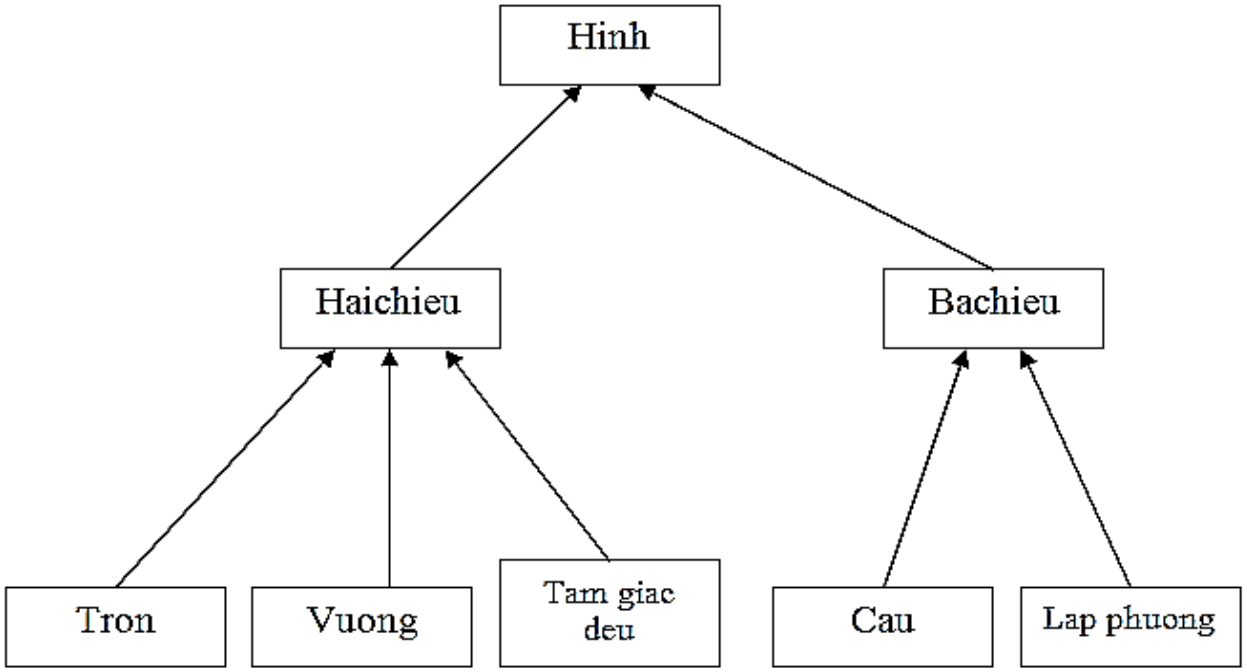
Bài tập

- Xác định kết quả chương trình sau:

```
class A     {
public:
    A() { cout << "A constructor\n"; }
    void m1() { cout << "A.m1\n"; m2();  }
    virtual void m2() { cout << "A.m2\n"; }
};
class B : public A     {
public:
    B() { cout << "B constructor\n";}
    void m1() { cout << "B.m1\n"; }
    void m2() { cout << "B.m2\n"; }
};
void func(A &a) { a.m1(); }
int main()   {
    B b;
    func(b);
}
```

Bài tập

Xây dựng các lớp theo cây kế thừa sau:



Yêu cầu:

- Tất cả các loại hình đều có chung phương thức `ten()` để xác định tên hình là gì và phương thức `in()` để thể hiện tên hình
- Các hình hai chiều đều có phương thức `dt()` để tính diện tích
- Các hình ba chiều đều có phương thức `tt()` để tính thể tích

Nội dung chính

1. Giới thiệu môn học
2. Các khái niệm cơ bản
3. Hàm trong C++
4. Lớp và đối tượng
5. Định nghĩa chồng toán tử
6. Hàm tạo và hàm huỷ
7. Dẫn xuất và thừa kế
8. Tương ứng bội
9. Khuôn hình

ĐÃ XONG!