# ĐA XẠ - ĐA HÌNH - POLYMORPHISM

1. TS. Nguyễn Tấn Trần Minh Khang
2. ThS. Võ Duy Nguyên
3. ThS. Nguyễn Hoàng Ngân
4. Hồ Thái Ngọc – Source code.

# 1. MỤC TIÊU

# Mục tiêu

- Hiểu được cơ chế hoạt động của phương thức ảo.
- Ứng dụng được phương thức ảo.
- Thi cao học đề thi hay hỏi phần này.
- Phỏng vấn xin việc người ta cũng rất hay hỏi.

# 2. VÍ DỤ DẪN NHẬP

# Ví dụ dẫn nhập

- Hãy thực hiện đoạn chương trình dưới đây và cho biết kết quả của việc chạy chương trình trong bốn trường hợp:
- *Trường hợp 1:* **XXXX** là khoảng trắng,
  **YYYY** là khoảng trắng.
- *Trường hợp 2:* **XXXX** là virtual,
  **YYYY** là khoảng trắng.
- *Trường hợp 3:* **XXXX** là khoảng trắng,
  **YYYY** là virtual.
- *Trường hợp 4:* **XXXX** là virtual,
  **YYYY** là virtual.

# Ví dụ dẫn nhập

```
11.#include <iostream>
12.class A
13.{
14.|    public:
15.|        XXXX void Sketchy()
16.|        {
17.|        |    cout<<"\n A's Sketchy()";
18.|        |    Sketchy(-1);
19.|        }
20.|        YYYY void Sketchy(int num)
21.|        {
22.|        |    cout<<"\n A's Sketchy("<<num<<")";
23.|        }
24.};
```

# Ví dụ dẫn nhập

```
11.class B:public A
12.{
13.|   public:
14.|      void Sketchy()
15.|      {
16.|      |   cout<<"\n B's Sketchy()";
17.|      |   Sketchy(-2);
18.|      }
19.|      void Sketchy(int num)
20.|      {
21.|      |   cout<<"\n B's Sketchy("<<num<<")";
22.|      }
23.};
```

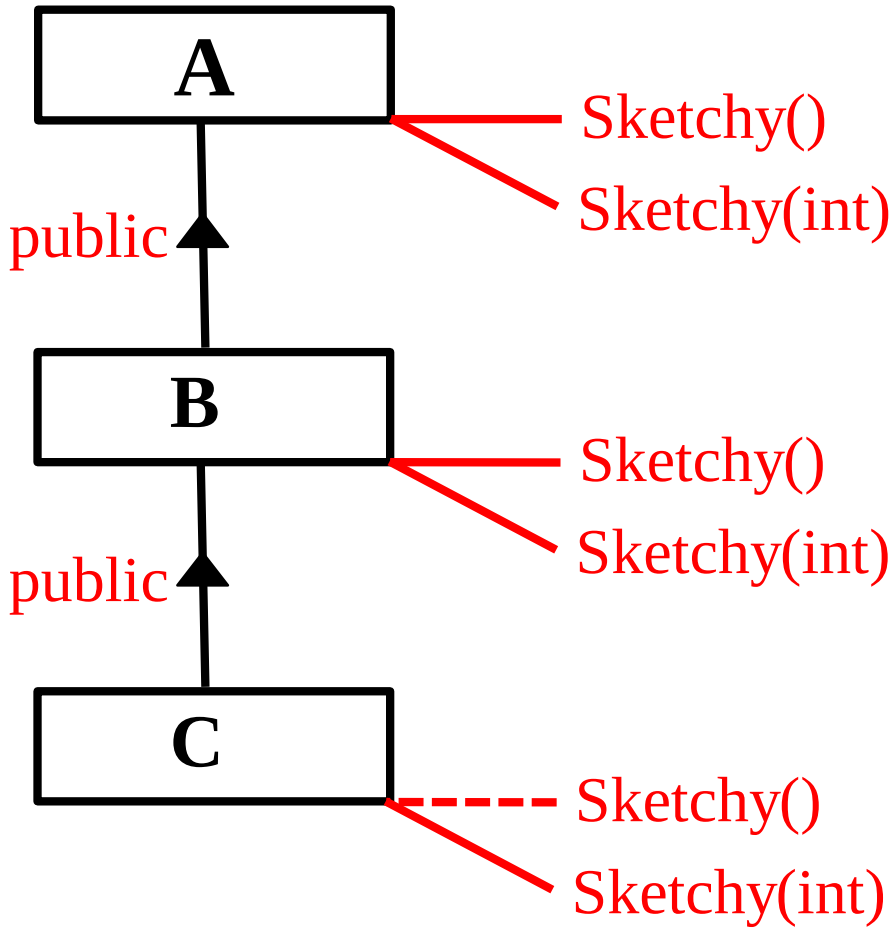
# Ví dụ dẫn nhập

```
11.class C:public B
12.{
13.|    public:
14.|        void Sketchy(int num)
15.|        {
16.|        |   cout<<"\n C's Sketchy("<<num<<")";
17.|        }
18.};
19.void Curious(A* wacky)
20.{
21.|    wacky->Sketchy();
22.|    ((C*)wacky)->Sketchy(123);
23.}
```

# Ví dụ dẫn nhập

```
11.void main()
12.{
13.|    A* inky = new B;
14.|    inky->Sketchy();
15.|    inky->Sketchy(23);
16.|    Curious(inky);
   |
17.|    B* pinky = new C;
18.|    pinky->Sketchy();
19.|    pinky->Sketchy(46);
20.|    Curious(pinky);
21.}
```

# Ví dụ dẫn nhập



# 3. KHÁI NIỆM ĐA XẠ

# Khái niệm đa xạ

- Khái niệm: Đa xạ là cơ chế **tầm vực động** (dynamic scope) cho phép "*xác định*" đúng hành vi (phương thức – method) của đối tượng (object) khi yêu cầu thực hiện.
- Việc "*xác định*" được thực hiện theo nguyên tắc tự nhiên: đối tượng thuộc lớp nào sẽ gọi thực hiện phương thức của lớp đối tượng (class) đó.
- Tầm vực động (dynamic scope) là cơ chế gọi thực hiện phương thức thông qua con trỏ đối tượng.

Syntax

# 4. CÚ PHÁP ĐA XẠ

# Cú pháp đa xạ

```
11.class CCoSo
12.{
13.|    private:
14.|        ...
15.|    protected:
16.|        ...
17.|    public:
18.|        ...
19.|        virtual KDL <TenPhuongThuc>(<ThamSo>);
20.};
```

# Cú pháp đa xạ

```
11.class CDanXuat::<Từ khóa dẫn xuất> CCoSo
12.{
13.|    private:
14.|        ...
15.|    protected:
16.|        ...
17.|    public:
18.|        ...
19.|        KDL <TenPhuongThuc>(<ThamSo>);
20.};
```

```
11.class CCoSo
12.{
13.|    private: …
14.|    protected: …
15.|    public: …
16.|        ...
17.|        virtual KDL <TenPhuongThuc>(<ThamSo>);
18.};
19.class CDanXuat::<Từ khóa dẫn xuất> CCoSo
20.{
21.|    private: …
22.|    protected: …
23.|    public: …
24.|        ...
25.|        KDL <TenPhuongThuc>(<ThamSo>);
26.};
```

# Cú pháp đa xạ

- Một phương thức được khai báo bắt đầu với từ khóa virtual thì được gọi là phương thức ảo và phương thức này được gọi thực hiện theo cơ chế đa xạ nếu lời gọi thực hiện phương thức được thông qua một con trỏ đối tượng.

- Các phương thức ở lớp dẫn xuất cùng tên và cùng danh sách tham số đầu vào thì cũng sẽ là phương thức ảo nếu ở lớp cơ sở phương thức cùng tên và cùng tham số là phương thức ảo.