



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

ĐA HÌNH (tt)

C++



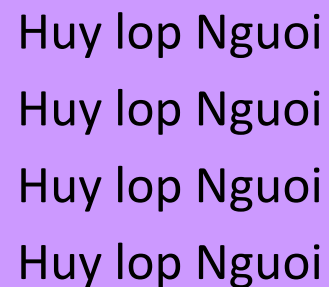
Microsoft®

Visual Studio®

Phương thức hủy bỏ ảo

❖ Trong ví dụ quản lý danh sách các đối tượng thuộc các lớp **Ngnoi**, **SinhVien**, **CongNhan**,... Thao tác dọn dẹp đối tượng là cần thiết.

```
void main(){  
    Ngnoi *a[N];  
    a[0] = new SinhVien("Vien Van Sinh", "20001234", 1982);  
    a[1] = new NuSinh("Le Thi Ha Dong", "20001235", 1984);  
    a[2] = new CongNhan("Tran Nhan Cong", 1000000, 1984);  
    a[3] = new Ngnoi("Nguyen Thanh Nhan", 1960);  
    XuatDs(4, a);  
    for ( int i = 0; i < 4; i++)    delete a[i];  
};
```



Huy lop Ngnoi
Huy lop Ngnoi
Huy lop Ngnoi
Huy lop Ngnoi

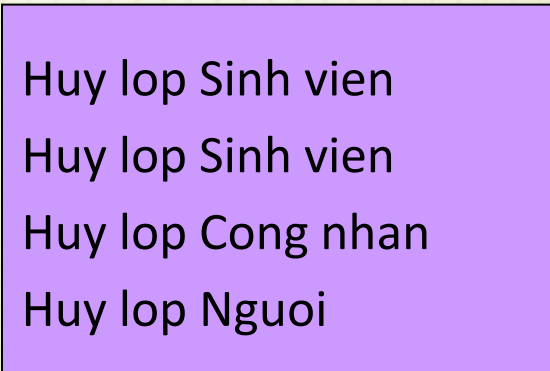
Phương thức hủy bỏ ảo

- ❖ Thông qua con trỏ thuộc lớp cơ sở `Nguoi`, chỉ có phương thức hủy bỏ của lớp `Nguoi` được gọi.
- ❖ Để bảo đảm việc dọn dẹp là đầy đủ, ta phải dùng phương thức hủy bỏ ảo.

```
class Nguoi{  
    protected:  
        char *HoTen; int NamSinh;  
    public:  
        Nguoi(char *ht, int ns):NamSinh(ns) { HoTen = strdup(ht); }  
        virtual ~Nguoi() {      cout<<“\nHuy lop Nguoi”;  
                               delete [ ] HoTen;  
        }  
        virtual void Xuat(ostream &os) const {///  
};
```

Phương thức hủy bỏ ảo

```
void main(){  
    Ngươi *a[N];  
    a[0] = new SinhVien("Vien Van Sinh", "20001234",1982);  
    a[1] = new NuSinh("Le Thi Ha Dong", "20001235",1984);  
    a[2] = new CongNhan("Tran Nhan Cong",1000000, 1984);  
    a[3] = new Ngươi("Nguyen Thanh Nhan", 1960);  
    XuatDs(4,a);  
    for ( int i = 0; i < 4; i++) delete a[i];  
}
```



Huy lop Sinh vien
Huy lop Sinh vien
Huy lop Cong nhan
Huy lop Ngươi

Phương thức hủy bỏ ảo

- Dr. Guru khuyên:
 - **Hàm hủy của lớp phải luôn là hàm ảo.**
 - ➔ Chuyển lời gọi đến hàm hủy của lớp kế thừa.

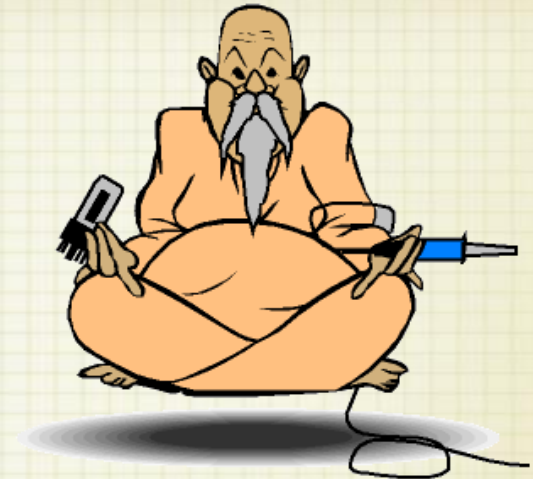
```
class GiaoVien
{
    private:
        char    *HoTen;
    public:
        virtual ~GiaoVien() { delete HoTen; }
};
```

GiaoVien *p = new GVCN;

delete p;

GiaoVien()
GVCN()

~GVCN()
~GiaoVien()



Lớp cơ sở trừu tượng

Phương thức thuần ảo

❖ Lớp cơ sở trừu tượng là lớp cơ sở không có đối tượng nào thuộc chính nó.

Ví dụ: Xét các lớp Circle, Rectangle, Square kế thừa từ lớp Shape:

- Các hàm trong lớp Shape có nội dung nhưng nội dung không có ý nghĩa.
- Đồng thời ta luôn luôn có thể tạo được đối tượng thuộc lớp Shape, điều này không đúng với tư tưởng của phương pháp luận hướng đối tượng.

(Theo quan điểm của J.Rumbaugh: xây dựng một mô hình của hệ thống trong lĩnh vực ứng dụng và thêm chi tiết trong quá trình thiết kế hệ thống)

Lớp cơ sở trừu tượng

Phương thức thuần ảo

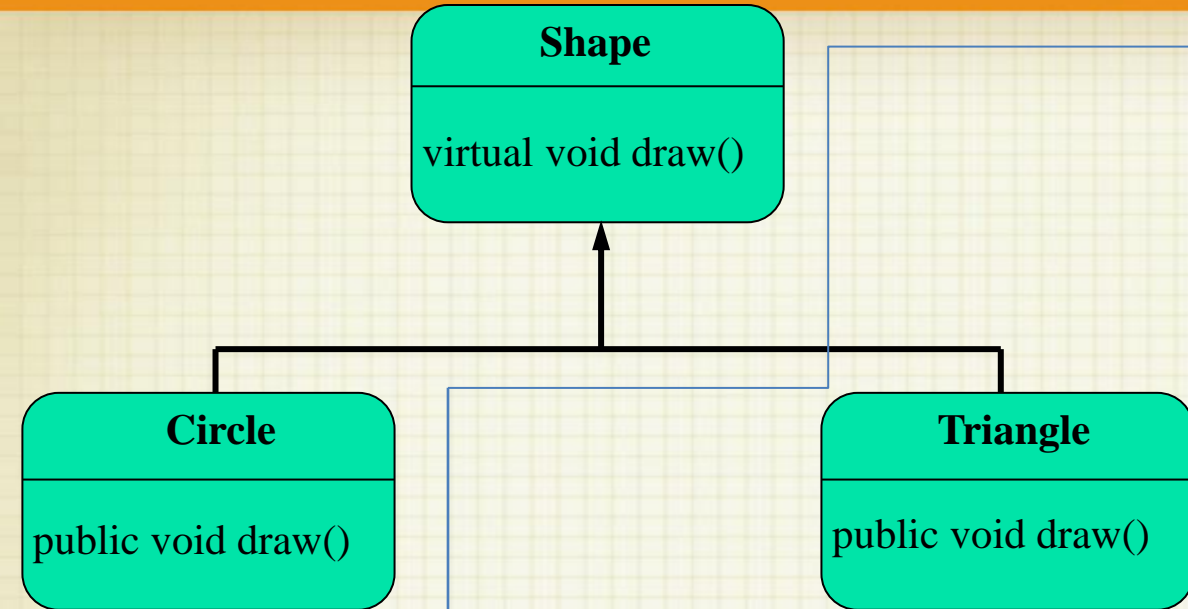
- ❖ Ta có thể thay thế cho **nội dung không có ý nghĩa** bằng **phương thức thuần ảo**, là **phương thức ảo không có nội dung**.
- ❖ Khi lớp có phương thức thuần ảo, lớp trở thành lớp cơ sở trừu tượng.
Ta không thể tạo đối tượng thuộc lớp cơ sở trừu tượng.
- ❖ Ta có thể định nghĩa phương thức thuần ảo, nhưng chỉ có các đối tượng thuộc lớp dẫn xuất có thể gọi nó.

Lớp cơ sở trừu tượng

Phương thức thuần ảo

- ❖ Phương thức thuần ảo chỉ có ý nghĩa cho việc tổ chức sơ đồ phân cấp các lớp, nó đóng vai trò chứa sẵn chỗ trống cho các lớp con điền vào với phiên bản phù hợp.
- ❖ Bản thân các lớp dẫn xuất của lớp cơ sở trừu tượng cũng có thể là lớp cơ sở trừu tượng.

Ví dụ



```
class Shape //Abstract
{ public :
    virtual void draw() = 0; //Pure virtual Function
};
```

```
class Circle : public Shape { //No draw() - Abstract
public:
    void print(){
        cout << "I am a circle" << endl; }
};
class Rectangle : public Shape {
public :
    void draw() { // Override Shape::draw()
        cout << "Drawing Rectangle" << endl;}
};
```

| | | |
|--------------|-----|-----|
| Shape p; | ??? | err |
| Shape *s; | ??? | OK |
| Rectangle r; | ??? | OK |
| Circle c; | ??? | err |

Ví dụ - Xét đoạn chương trình 1

```
class CPolygon
{ protected:
    int width, height;
public:
    void set_values (int a, int b) { width=a; height=b; }
    virtual int area ( ) { return (0); }
};
class CRectangle: public CPolygon
{ public:
    int area ( ) { return (width * height); }
};
class CTriangle : public CPolygon
{ public:
    int area ( ) { return (width * height/2); }
};
```

Output

20
10
0

```
int main ()
{
    CRectangle rect;
    CTriangle trgl;
    CPolygon poly;
    CPolygon * ppoly1 = &rect;
    CPolygon * ppoly2 = &trgl;
    CPolygon * ppoly3 = &poly;
    ppoly1->set_values (4,5);
    ppoly2->set_values (4,5);
    ppoly3->set_values (4,5);
    cout << ppoly1->area() << endl;
    cout << ppoly2->area() << endl;
    cout << ppoly3->area() << endl;
```


Ví dụ - Xét đoạn chương trình 2

```
class CPolygon {
protected:
    int width, height;
public:
    void set_values (int a, int b)
    { width=a; height=b; }
    virtual int area (void) = 0;
};

class CRectangle: public CPolygon {
public:
    int area (void)
    { return (width * height); }
};
```

```
class CTriangle: public CPolygon {
public:
    int area (void)
    { return (width * height / 2); }
};

int main ()
{
    CRectangle rect;
    CTriangle trgl;
    CPolygon * ppoly1 = &rect;
    CPolygon * ppoly2 = &trgl;
    ppoly1->set_values (4,5);
    ppoly2->set_values (4,5);
    cout << ppoly1->area() << endl;
    cout << ppoly2->area() << endl;
}
```

Output

20

10

Ví dụ - Xét đoạn chương trình 3

```
class CPolygon {
protected:
    int width, height;
public:
    void set_values (int a, int b)
        { width=a; height=b; }
    virtual int area (void) =0;
    void printarea (void)
        { cout << this->area() << endl; }
};

class CRectangle: public CPolygon {
public:
    int area (void)
        { return (width * height); }
};
```

```
class CTriangle: public CPolygon {
public:
    int area (void)
        { return (width * height / 2); }
};

int main () {
    CRectangle rect;
    CTriangle trgl;
    CPolygon * ppoly1 = &rect;
    CPolygon * ppoly2 = &trgl;
    ppoly1->set_values (4,5);
    ppoly2->set_values (4,5);
    ppoly1->printarea();
    ppoly2->printarea();
    return 0;
}
```

Output

20

10

Ví dụ - Xét đoạn chương trình 4

```
class CPolygon {  
protected:  
    int width, height;  
public:  
    void set_values (int a, int b)  
        { width=a; height=b; }  
    virtual int area (void) =0;  
    void printarea (void)  
        { cout << this->area() << endl; }  
};
```

```
class CRectangle: public CPolygon {  
public:  
    int area (void)  
        { return (width * height); }  
};
```

```
class CTriangle: public CPolygon {  
public:  
    int area (void)  
        { return (width * height / 2); }  
};  
int main ()  
{  
    CPolygon * ppoly1 = new CRectangle;  
    CPolygon * ppoly2 = new CTriangle;  
    ppoly1->set_values (4,5);  
    ppoly2->set_values (4,5);  
    ppoly1->printarea();  
    ppoly2->printarea();  
    delete ppoly1;  
    delete ppoly2;  
    return 0;  
}
```

Output

20

10

Bài toán di truyền nhóm máu

Đầu những năm 1900, dựa trên sự hiện diện của các kháng nguyên trên màng hồng cầu, các nhà khoa học đã xác định rằng con người có 4 nhóm máu khác nhau: O, A, B và AB. Hệ thống phân loại nhóm máu này (gọi là hệ thống nhóm máu ABO) cung cấp cho bác sĩ các thông tin quan trọng để lựa chọn nhóm máu phù hợp trong việc truyền máu. Và đồng thời có thể tiên đoán được nhóm máu tương đối của người con dựa trên nhóm máu của cha mẹ theo cơ chế di truyền học.

Nhóm máu của người con khi biết được nhóm máu của cha và mẹ:

| | | Nhóm máu người cha | | | | |
|-------------------|----|--------------------|--------------|-----------|-------|-------------------------------------|
| | | A | B | AB | O | |
| Nhóm máu người mẹ | A | A / O | A, B, AB / O | A, B / AB | A / O | Dự đoán khả năng nhóm máu người con |
| | B | A, B, AB / O | B / O | A, B / AB | B / O | |
| | AB | A, B / AB | A, B / AB | A, B / AB | A / B | |
| | O | A / O | B / O | A / B | O | |

Bài toán di truyền nhóm máu

Ngoài ra còn có thêm hệ thống phân loại Rh (Rhesus) .


Căn cứ vào sự khác biệt khi nghiên cứu về sự vận chuyển oxy của hồng cầu thì các hồng cầu có thể mang ở mặt ngoài một protein gọi là Rhesus. Nếu có kháng nguyên D thì là nhóm Rh+ (dương tính), nếu không có là Rh- (âm tính). Các nhóm máu A, B, O, AB mà Rh- thì được gọi là âm tính A-, B-, O-, AB-. Nhóm máu Rh- chỉ chiếm 0,04% dân số thế giới. Đặc điểm của nhóm máu Rh này là chúng chỉ có thể nhận và cho người cùng nhóm máu, đặc biệt phụ nữ có nhóm máu Rh- thì con rất dễ tử vong.

Người có nhóm máu Rh+ chỉ có thể cho người cũng có nhóm máu Rh+ và nhận người có nhóm máu Rh+ hoặc Rh-; Người có nhóm máu Rh- có thể cho người có nhóm máu Rh+ hoặc Rh- nhưng chỉ nhận được người có nhóm máu Rh- mà thôi.

Trường hợp người có nhóm máu Rh- được truyền máu Rh+, trong lần đầu tiên sẽ không có bất kỳ phản ứng tức thì nào xảy ra nhưng nếu tiếp tục truyền máu Rh+ lần thứ 2 sẽ gây ra những hậu quả nghiêm trọng do tai biến truyền máu. Tương tự với trường hợp mẹ Rh- sinh con (lần đầu và lần thứ hai trở đi) .

Bài toán di truyền nhóm máu

Khả năng tương thích: :  Có thể cho - nhận.

 : Không thể cho - nhận.

| Bảng khả năng tương thích hồng cầu | | | | | | | | |
|------------------------------------|---|---|---|---|---|---|---|---|
| Người nhận | Người cho | | | | | | | |
| | O- | O+ | A- | A+ | B- | B+ | AB- | AB+ |
| O- |  |  |  |  |  |  |  |  |
| O+ |  |  |  |  |  |  |  |  |
| A- |  |  |  |  |  |  |  |  |
| A+ |  |  |  |  |  |  |  |  |
| B- |  |  |  |  |  |  |  |  |
| B+ |  |  |  |  |  |  |  |  |
| AB- |  |  |  |  |  |  |  |  |
| AB+ |  |  |  |  |  |  |  |  |

Bài toán di truyền nhóm máu

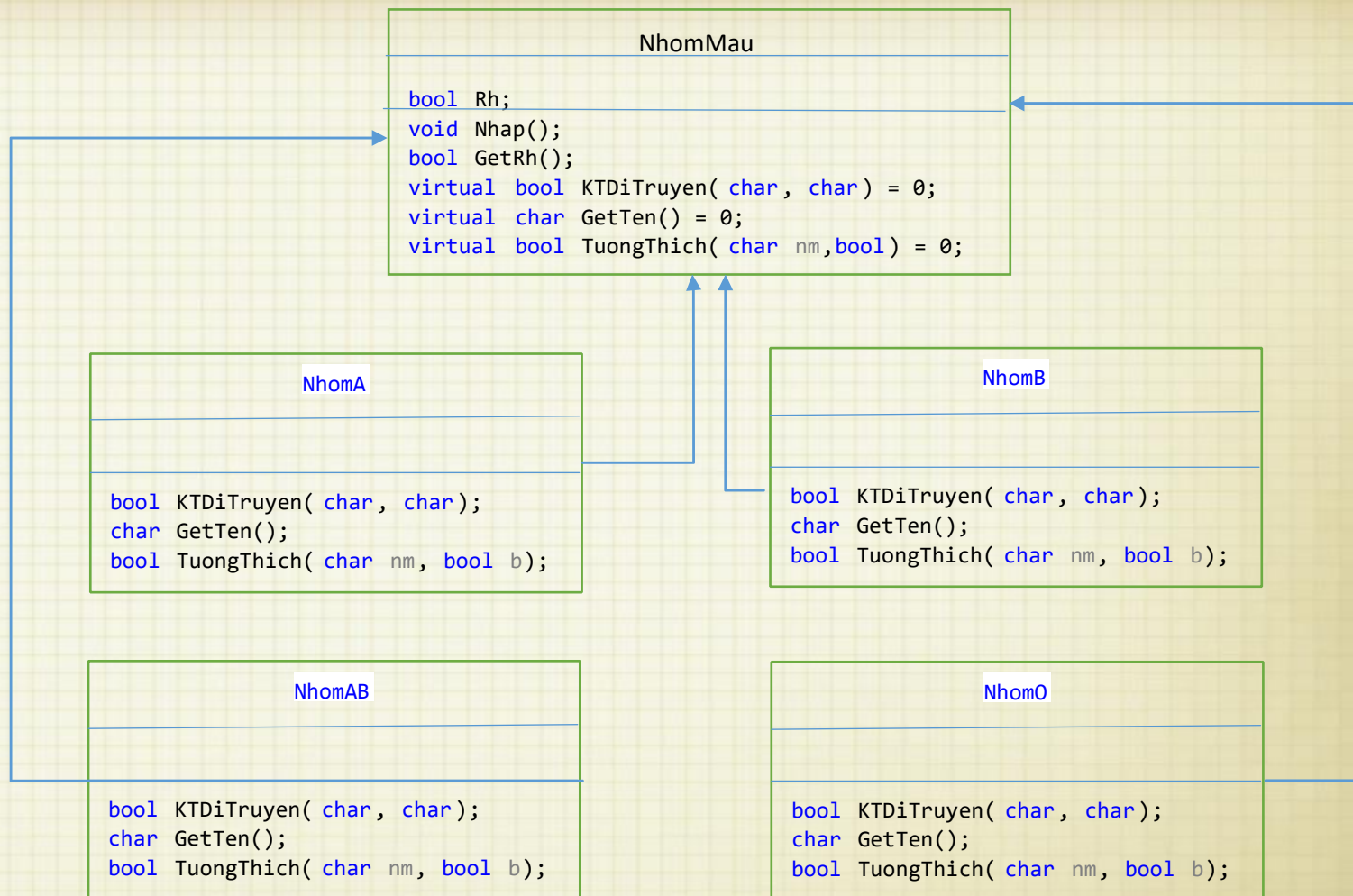
Áp dụng kiến thức lập trình hướng đối tượng (kế thừa, đa hình)

- Thiết kế sơ đồ chi tiết các lớp đối tượng (1.5đ).
- Xây dựng chương trình thực hiện các yêu cầu sau:
 1. Nhập danh sách các nhóm máu của một nhóm người. (1đ)
 2. Cho một bộ 3 nhóm máu của 3 người là cha, mẹ, con. Hãy kiểm tra và đưa ra kết quả nhóm máu có phù hợp với quy luật di truyền hay không? (1đ)
 3. Chọn một người X trong danh sách. Hãy liệt kê tất cả các người còn lại trong danh sách có thể cho máu người X này. (1đ)

Hướng dẫn

- Vẽ sơ đồ lớp đối tượng
- Nhập danh sách nhóm máu
- Kiểm tra quy luật di truyền
- Danh sách người có thể cho máu người X

Sơ đồ lớp đối tượng :



Q & A

