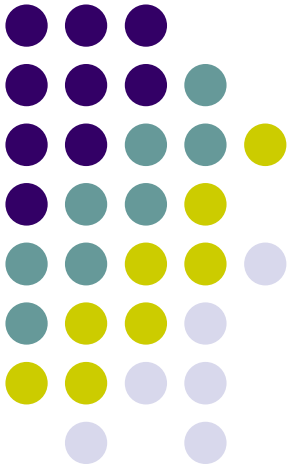
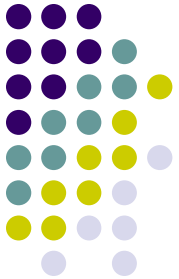
Chương 5

Tính đa hình(Polymorphism)



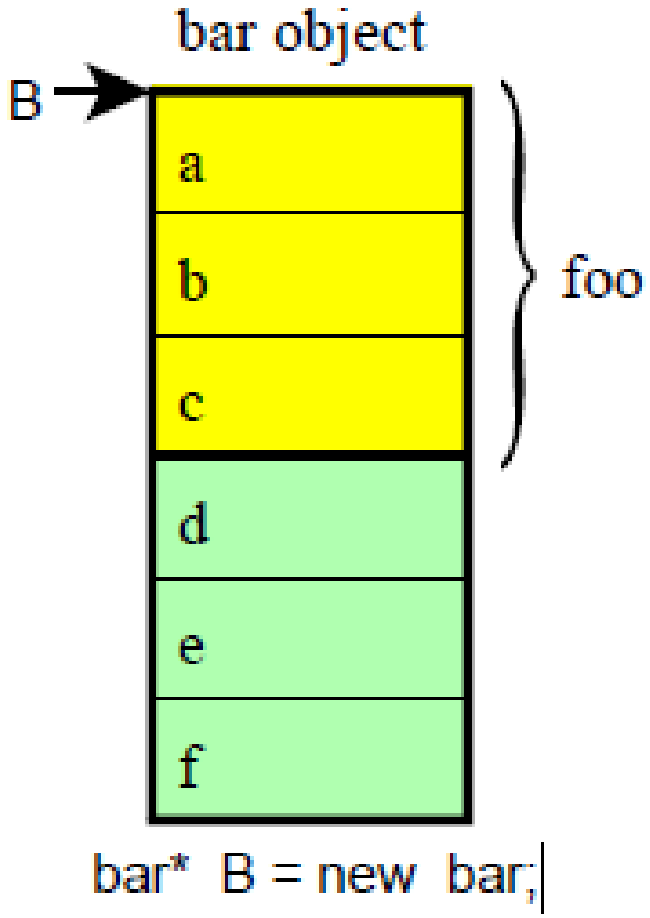
Nội dung



- Chuyển kiểu
- Kết gán sớm, Kết gán muộn
- Hàm ảo, lớp trừu tượng

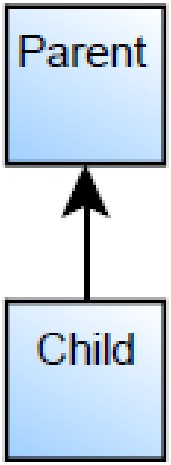
Kế thừa

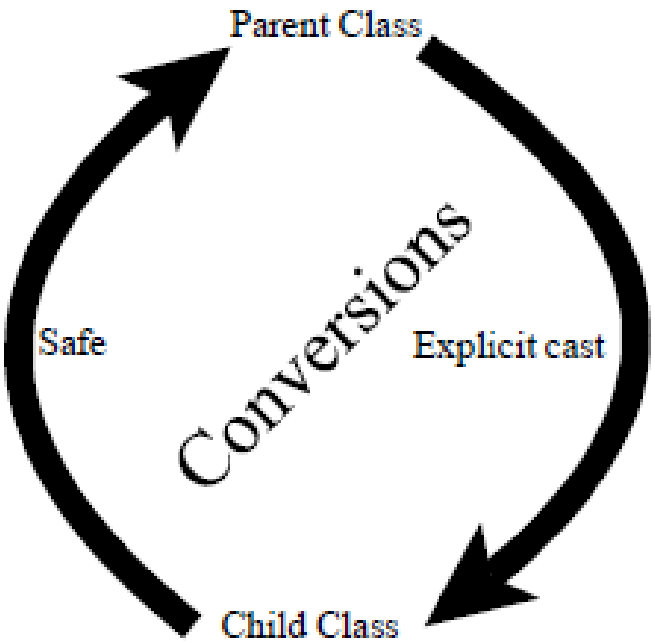
```
class foo
{
   private:
      int  a;
      int  b;
      int  c;
};


class bar :  public foo
{
   private:
      int  d;
      int  e;
      int  f;
};
```



Chuyển kiểu

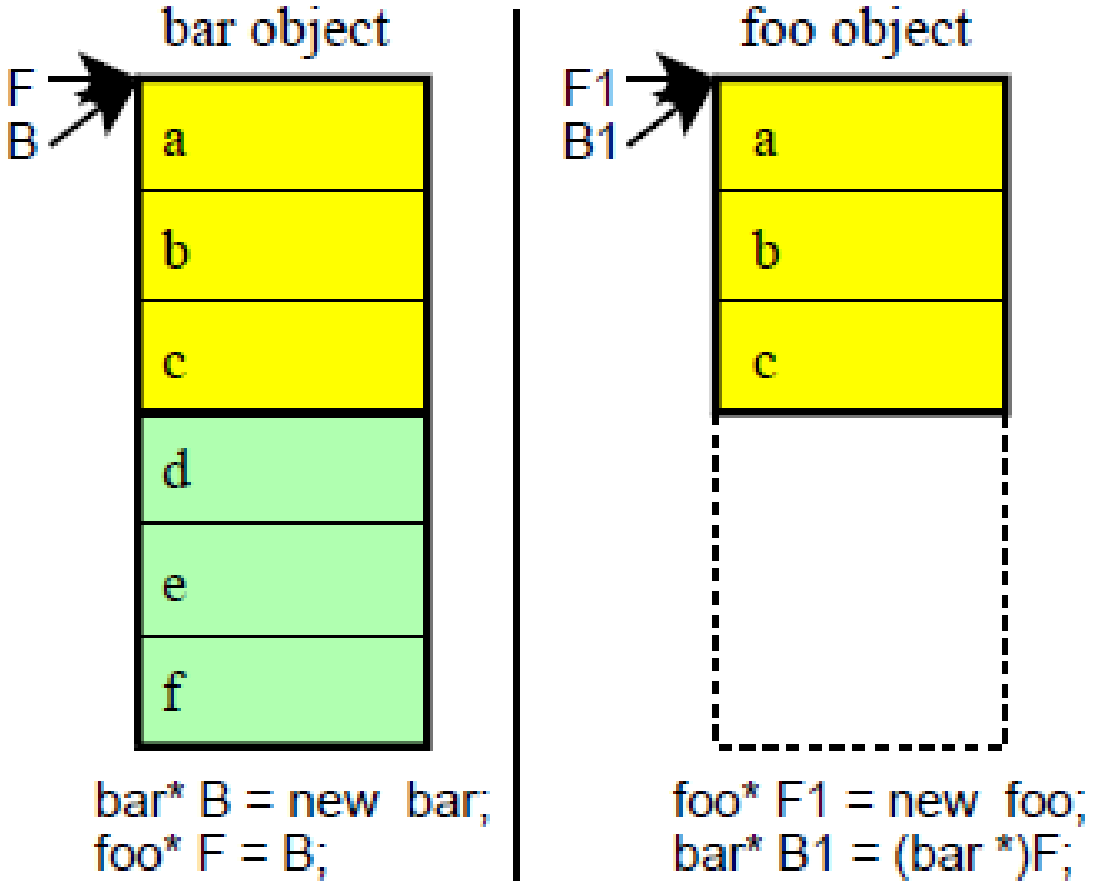
Trong C++ cho phép con trỏ đối tượng của lớp cha trỏ tới đối tượng của lớp con



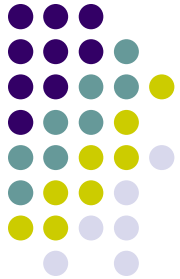



```
class  Parent { }
class  Child { }

Child     C;
Parent*   Pptr = &C;      //chuyển kiểu an toàn
Child*    Cptr = (Child *) Pptr;   //chuyển kiểu không an toàn
```

Ví dụ chuyển kiểu

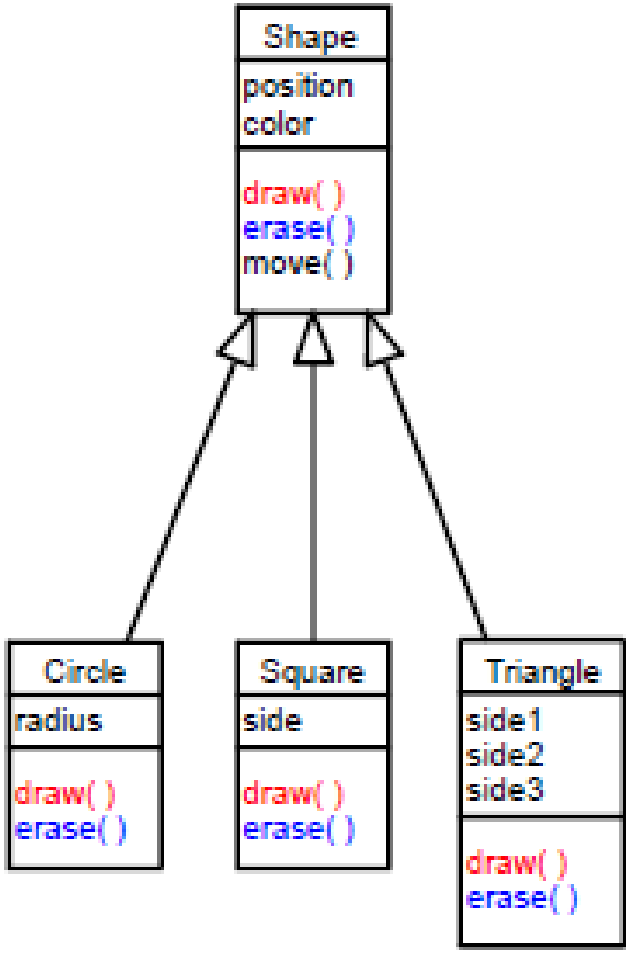


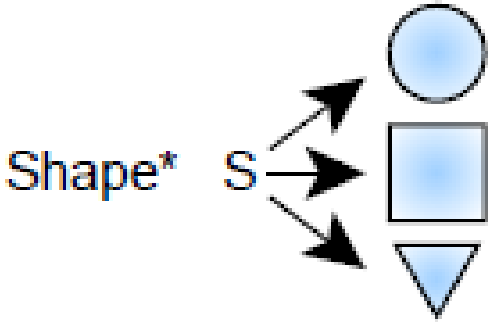
Kết gán sớm và kết gán muộn



- Kết gán sớm(early binding)
  - Hàm thành phần gọi từ con trỏ đối tượng được xác định ngay tại thời điểm dịch
- Kết gán muộn(late binding)
  - Hàm thành phần gọi từ con trỏ đối tượng được xác định tại thời điểm thực thi chương trình phụ thuộc vào đối tượng cụ thể đang được trỏ đến.

Vấn đề kết gán sớm

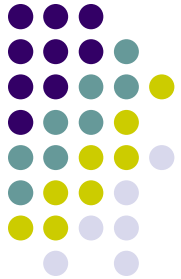





```
Shape *s = new Circle();
S->draw() //non- polymorphic
```

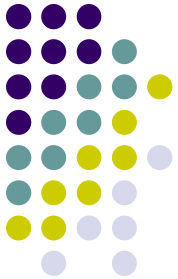
Hàm draw() nào được gọi???

Tính đa hình



- Một đối tượng có khả năng gọi các phương thức khác nhau với cùng một lời gọi hàm(same message).
- Kết gán phương thức chính xác tại thời điểm thực thi dựa vào đối tượng cụ thể được cấp phát.

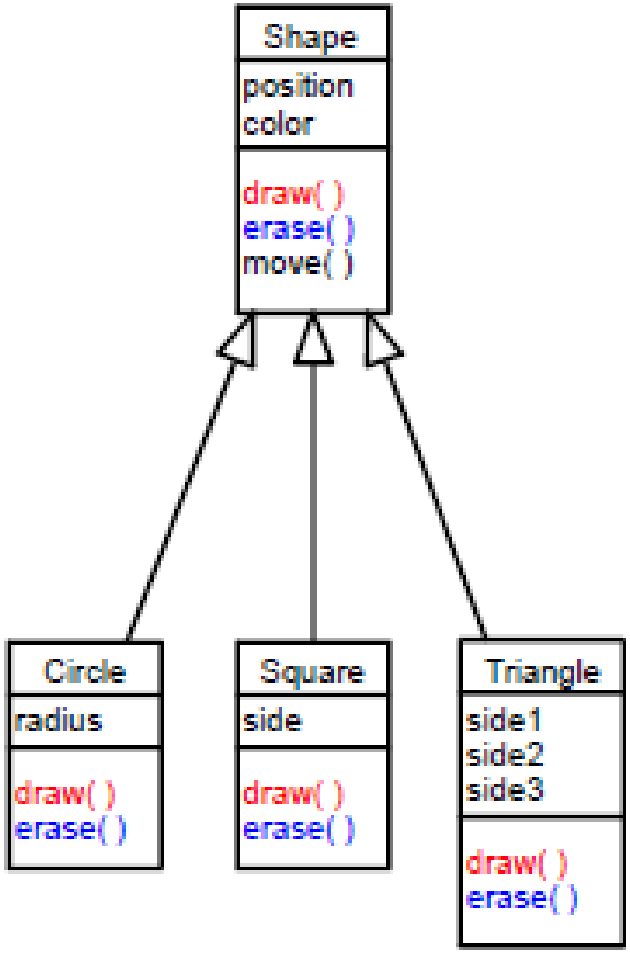
Phương thức ảo – Hàm đa hình



- Cú pháp

  virtual <kiểu trả về> <tên hàm>([ds tham số])
- Được khai báo trong lớp cơ sở và định nghĩa lại trong lớp dẫn xuất
- Cho phép ghép kiểu động từ lớp cha sang lớp con
- Chú ý:
  - Định nghĩa các phương thức ảo như các phương thức thông thường
  - Sử dụng con trỏ để truy cập tới hàm ảo
  - Không có hàm khởi tạo ảo nhưng có thể có hàm huỷ ảo

Giải pháp kết gán muộn




```
class Shape
{
   public:
        ...
      virtual  void  draw( );
};
```

```
Shape *s = new Circle();
S->draw(); //polymorphic
```

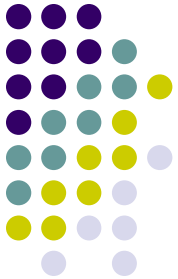
Hàm draw() trong lớp Circle sẽ được gọi

Ví dụ

```
class Parent
{
   public:
                  void    funcA( );
   virtual        void    funcB( );
                  void    funcC( );
};

class Child : public Parent
{
   public:
                  void    funcA( );
   virtual        void    funcB( );
};
```

```
int  main( )
{
   Parent*  P1 = new  Parent;
   Parent*  P2 = new  Child;
   Child*   C = new  Child;


   P1->funcA( );   //_____
   P1->funcB( );   //_____
   P1->funcC( );   //_____
   P2->funcA( );   //_____
   P2->funcB( );   //_____
   P2->funcC( );   //_____
   C->funcA( );    //_____
   C->funcB( );    //_____
   C->funcC( );    //_____
```

Phương thức ảo thuần túy (Pure Virtual Functions)

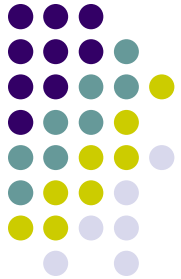


- Là phương thức ảo không có định nghĩa trong lớp cơ sở
- Cú pháp

virtual <kiểu trả về> <tên hàm>([ds tham số]) = 0;

- Cung cấp một phương thức thống nhất làm giao diện chung

Lớp trừu tượng(Abstract Class)



- Là lớp có ít nhất một phương thức ảo thuần túy

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

- Lớp dẫn xuất cần phải định nghĩa lại phương thức ảo thuần túy
- Không thể tạo đối tượng của một lớp trừu tượng, nhưng con trỏ có kiểu là lớp trừu tượng được phép khai báo