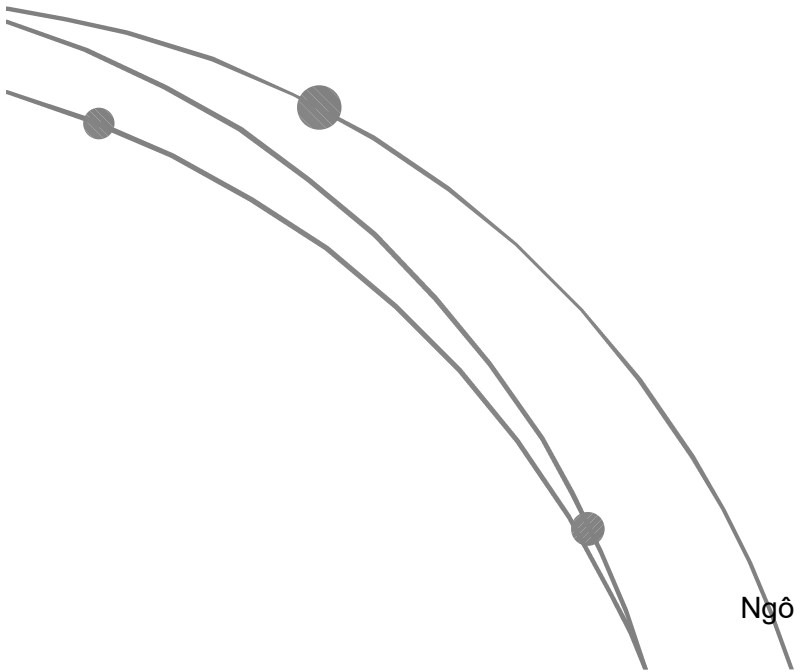
Chương 8. Đa hình động, Hàm ảo

I. Hàm ảo và đa hình động

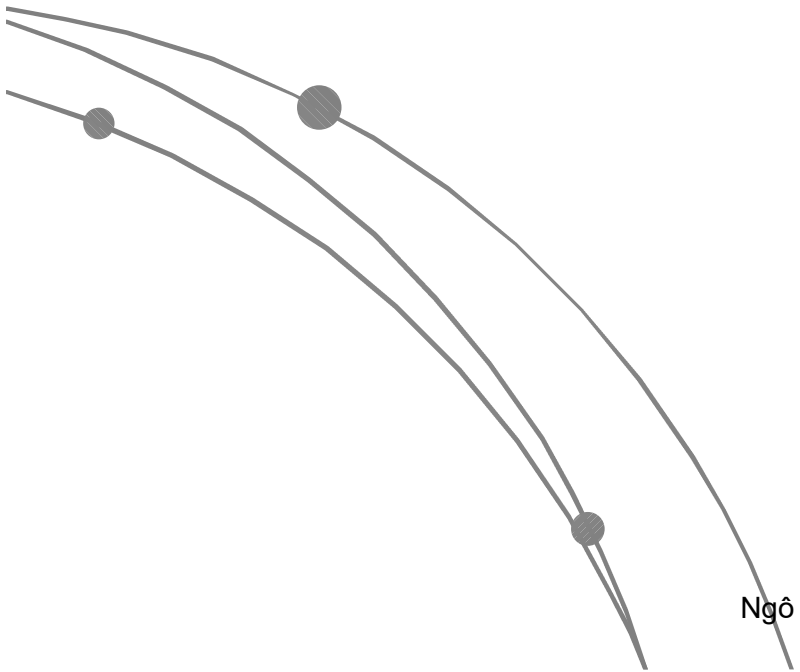
II. Ứng dụng của đa hình động

III. Lớp trừu tượng, hàm tạo và hàm hủy ảo



I. Hàm ảo và sự đa hình

1. Giới thiệu về hàm ảo và sự đa hình
2. Gọi hàm thành viên qua con trỏ lớp cơ sở
3. Sự liên kết động



I.1. Giới thiệu về hàm ảo và sự đa hình

- Dạng đa hình thứ hai trong LTHĐT liên quan tới sự kế thừa, hàm ảo và con trỏ. Ở đây sự đa hình thái thể hiện ở chỗ: Lời gọi tới một hàm thành viên sẽ làm cho các hàm thành viên khác nhau được thực hiện tuỳ thuộc vào kiểu đối tượng gọi hàm đó. Sự đa hình này còn được gọi là sự liên kết động.
- Hàm ảo là hàm thành viên của lớp, giống như các hàm thành viên thông thường, chỉ khác là được khai báo với từ khóa virtual đặt trước.

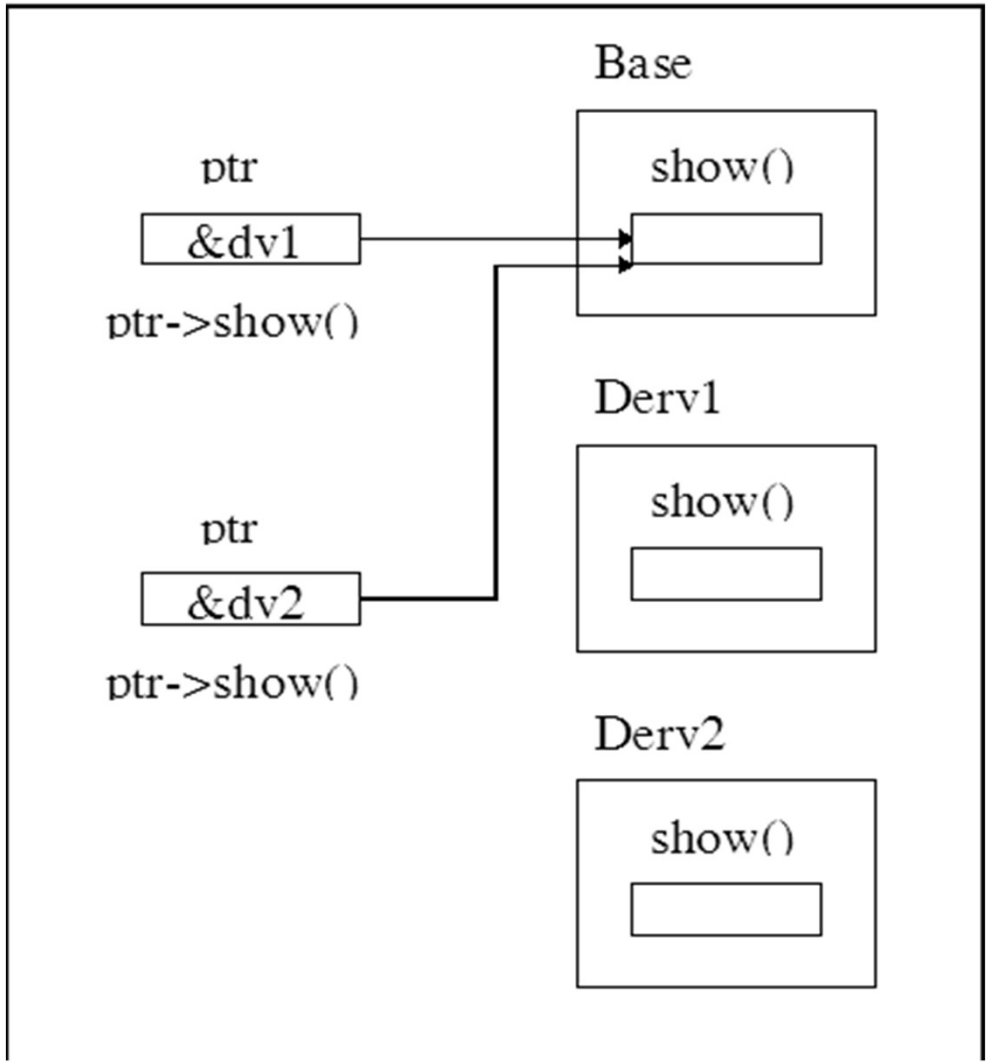
  virtual void nhap();

I.2. Gọi hàm thành viên qua con trỏ lớp cơ sở

- Con trỏ lớp cơ sở có thể chứa địa chỉ của đối tượng các lớp dẫn xuất. Bởi vì đối tượng lớp dẫn xuất là một loại đối tượng lớp cơ sở nên các con trỏ trỏ tới đối tượng của một lớp dẫn xuất có kiểu phù hợp với các con trỏ trỏ tới đối tượng của lớp cơ sở.
- Khi một lớp cơ sở và các lớp dẫn xuất của nó có các hàm thành viên trùng nhau, nếu các hàm này được gọi qua con trỏ lớp cơ sở thì hàm được thực hiện luôn là hàm thành viên lớp cơ sở.

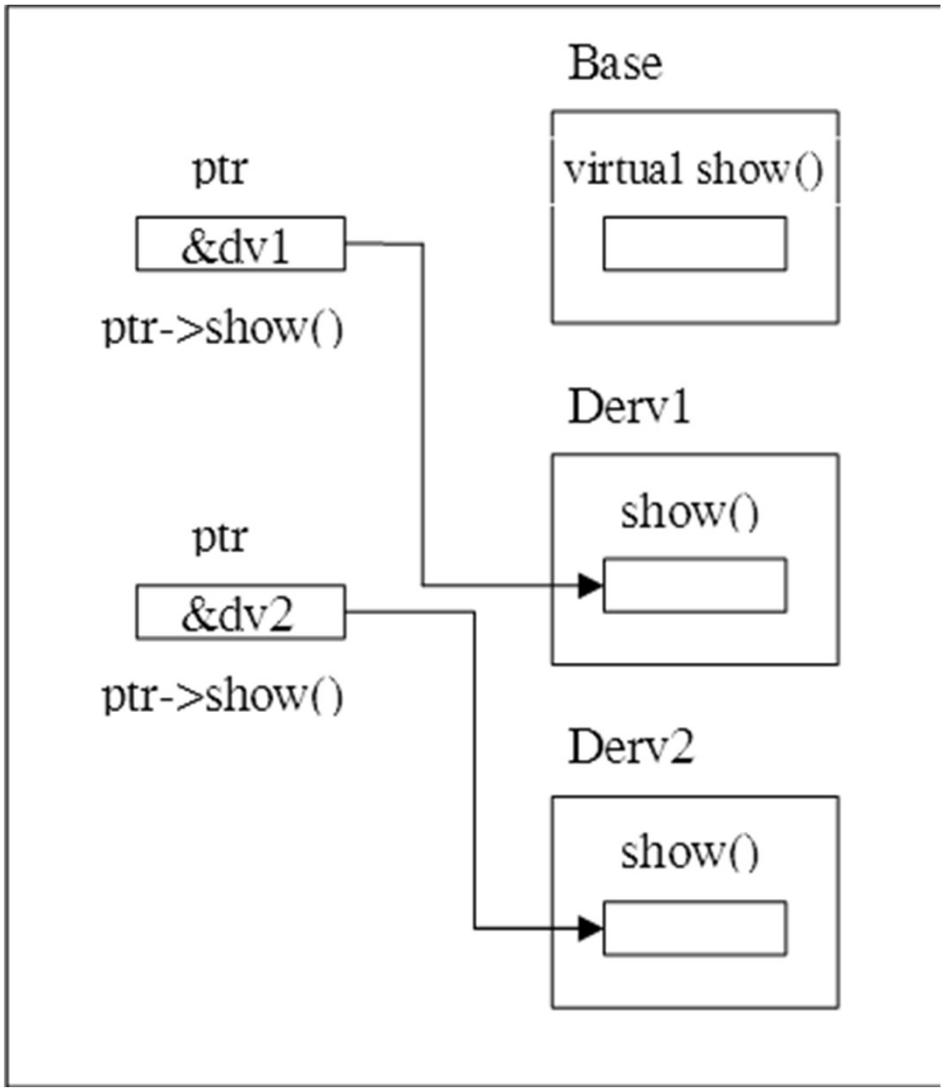
I.2. Gọi hàm thành viên qua con trỏ lớp cơ sở

- Sở dĩ các hàm thành viên lớp cơ sở luôn được thực hiện vì trình biên dịch bỏ qua nội dung của con trỏ và chọn hàm thành viên phù hợp với kiểu con trỏ là lớp cơ sở.



I.2. Gọi hàm thành viên qua con trỏ lớp cơ sở

- Để gọi được hàm thành viên của lớp dẫn xuất qua con trỏ lớp cơ sở ta cho các hàm thành viên của lớp cơ sở là hàm ảo.
- Khi dùng hàm ảo, trình biên dịch lựa chọn hàm để thực hiện dựa trên nội dung của con trỏ, chứ không phải tên kiểu của con trỏ. Đây là sự đa hình thái, vì một lời gọi hàm mà có thể thực hiện các hàm khác nhau, tuỳ thuộc vào nội dung của con trỏ.



I.3. Sự liên kết động

- Nếu trong lớp cơ sở có hàm ảo trùng tên với một hàm thành viên lớp dẫn xuất thì khi gọi hàm thành viên lớp dẫn xuất này qua con trỏ lớp cơ sở trình biên dịch sẽ không biết gọi hàm nào. Bởi vậy trình biên dịch phải sắp xếp để lựa chọn hàm thực hiện tại thời điểm chạy chương trình.
- Việc lựa chọn một hàm tại thời điểm chạy chương trình được gọi là sự liên kết động (**dynamic binding**). Còn việc lựa chọn hàm thực hiện theo cách thông thường, tại thời điểm biên dịch, được gọi là sự liên kết tĩnh (**static binding**).

I.3. Sự liên kết động

- Sự liên kết động cần nhiều thời gian và bộ nhớ hơn sự liên kết tĩnh: Lời gọi hàm lâu hơn, đối tượng lớp dẫn xuất lớn hơn.
- Tóm lại, để cài đặt sự liên kết động cần có kế thừa, hàm ảo, con trỏ và sự trùng hàm thành viên.

II. Ứng dụng của sự đa hình động

1. Mảng con trỏ lớp cơ sở trỏ tới các đối tượng của các lớp dẫn xuất khác nhau
2. Phân lập các phần chương trình

Ví dụ: Tính diện tích các hình: Hình tam giác biết 3 cạnh a,b,c; hình chữ nhật biết 2 cạnh a,b; hình tròn biết bán kính r; hình trụ biết bán kính r và chiều cao h. Nhập vào một số hình. Đưa ra diện tích các hình đã nhập. Yêu cầu cài đặt đa hình động.

II.1. Mảng con trỏ trỏ tới các đối tượng của các lớp khác nhau

- Một ứng dụng của sự đa hình là sử dụng mảng con trỏ lớp cơ sở để chứa địa chỉ của các đối tượng lớp dẫn xuất khác nhau.
- Ví dụ: Viết chương trình quản lý giảng viên và sinh viên. Thông tin về giảng viên có tên và số bài báo đã đăng, thông tin về sinh viên có tên và điểm TBC. Nhập vào một số giảng viên và sinh viên. Đưa ra màn hình thông tin về các giảng viên và sinh viên đã nhập, có kèm theo đánh giá: giảng viên giỏi nếu có số bài báo >=20, sinh viên giỏi nếu có điểm TBC>=9.0. Y/c cài đặt đa hình động.

II.2. Phân lập các phần chương trình

- Sự đa hình thái cũng có thể được sử dụng để giúp cho việc phân lập, hay gỡ bỏ sự phụ thuộc của một phần chương trình vào phần chương trình khác.
- Các chương trình OOP được chia thành hai phần được viết bởi những người lập trình khác nhau tại các thời điểm khác nhau. Đó là phần tạo lớp và phần sử dụng các lớp.

II.2. Phân lập các phần chương trình

- Việc lập trình sẽ đơn giản hơn nếu chương trình của người sử dụng lớp chỉ phải làm việc với một lớp thay vi nhiều lớp khác nhau. Điều này thực hiện được bằng sự đa hình, đó là dùng các tham chiếu hoặc các con trỏ trỏ tới các đối tượng để truyền và trả về từ một hàm.
- Ví dụ 1: Sử dụng đối số là tham chiếu

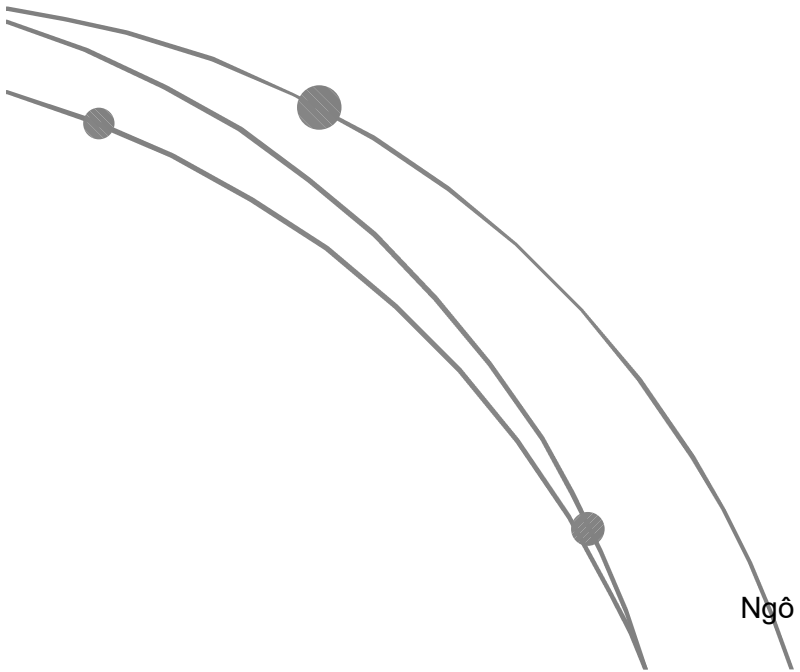
II.2. Phân lập các phần chương trình

- Ví dụ 2: Sử dụng đối số là con trỏ

- Ví dụ 3: Viết lại chương trình quản lý giảng viên và sinh viên trong đó có sử dụng hàm để truyền và trả về các đối tượng khác nhau.

III. Lớp trừu tượng, hàm tạo và hàm hủy ảo

1. Lớp trừu tượng (abstract class)
2. Hàm tạo ảo và hàm hủy ảo



III.1. Lớp trừu tượng

- Lớp trừu tượng là lớp mà không có đối tượng nào được tạo ra từ nó, nó chỉ đóng vai trò là lớp cơ sở cho các lớp dẫn xuất. Các lớp trừu tượng được cài đặt trong C++ bằng các hàm ảo tinh khiết (**pure virtual function**).
- Hàm ảo tinh khiết là một hàm ảo mà khi khai báo hàm có thêm ký hiệu =0 vào sau khai báo hàm. Thân của hàm ảo có thể có hoặc không có.

  Ví dụ: virtual void show()=0;

  hoặc virtual void show()=0

  { //Các lệnh của hàm }

III.1. Lớp trừu tượng

- Dấu bằng ở đây không phải là toán tử gán, giá trị 0 không được gán cho cái gì. Cú pháp =0 chỉ đơn giản là cách chỉ cho trình biên dịch biết rằng một hàm là tinh khiết.
- Để trình liên kết ngăn chặn việc tạo một đối tượng từ lớp trừu tượng, chúng ta phải định nghĩa ít nhất một hàm ảo tinh khiết trong lớp trừu tượng đó.

III.2. Hàm tạo và hàm hủy ảo

- Câu hỏi đặt ra là “các hàm tạo có bao giờ là ảo không?”. Không, không bao giờ. Các hàm ảo không thể tồn tại cho đến khi hàm tạo đã hoàn thành nhiệm vụ của nó, bởi vậy các hàm tạo không thể là ảo.
- Ngoài ra, khi tạo một đối tượng trình biên dịch cần biết loại đối tượng tạo ra. Do đó không có các hàm tạo ảo.
- Trái lại, các hàm huỷ có thể và thường nên là ảo.

III.2. Hàm tạo và hàm hủy ảo

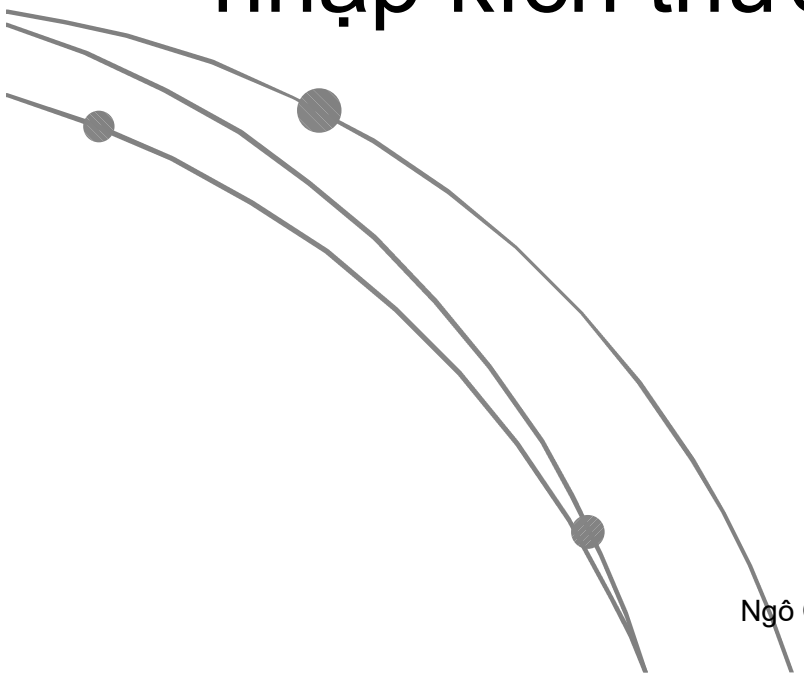
- Khi một lớp cơ sở có hàm ảo thì hàm hủy cũng nên để ảo. Nếu không để là ảo thì hàm hủy lớp dẫn xuất sẽ không được thực hiện khi hủy đối tượng lớp dẫn xuất thông qua con trỏ lớp cơ sở.
- Ví dụ:

III.2. Hàm tạo và hàm hủy ảo

- Khi nào một lớp cơ sở cần hàm hủy ảo? Khi thỏa mãn 3 điều kiện sau:
  - Cần tạo các lớp dẫn xuất từ lớp cơ sở
  - Các đối tượng lớp dẫn xuất được hủy qua con trỏ lớp cơ sở
  - Các hàm hủy trong lớp cơ sở và dẫn xuất thực hiện các công việc quan trọng chẳng hạn như giải phóng bộ nhớ.
- Tóm lại, cứ khi nào trong lớp cơ sở có hàm ảo thì nên để hàm hủy của nó là ảo.

Bài tập

Bài 1. Viết chương trình tính diện tích của các hình: hình chữ nhật có 2 cạnh, hình tròn có bán kính. Yêu cầu trong chương trình có cài đặt đa hình động cho hàm nhập kích thước và hàm tính diện tích.



Bài tập

Bài 2. Một nhân sự nói chung có họ tên và ngày sinh. Giảng viên là nhân sự nhưng có thêm mã gv và các môn học giảng dạy. Sinh viên là nhân sự nhưng có thêm mã sv và điểm TBC. Cán bộ quản lý là gv có thêm chức vụ. Nhập vào một số loại nhân sự. Đưa ra thông tin về các nhân sự đã nhập. Y/c cài đặt đa hình động.