

Chương 7. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

IV. Kế thừa nhiều mức

V. Hợp thành và kế thừa

VI. Kế thừa bội

Chương 7. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm không được kế thừa

I.7. Sự kế thừa và mối quan hệ loại

I.1. Tầm quan trọng của kế thừa trong OOP

- Sự kế thừa là khái niệm trung tâm thứ hai trong OOP.
- Sự kế thừa cho phép sử dụng lại, có nghĩa là đưa một lớp đã có vào sử dụng trong một tình huống lập trình mới. Nhờ việc sử dụng lại mà ta có thể giảm được thời gian và công sức khi viết một chương trình.
- Sự kế thừa còn đóng vai trò quan trọng trong việc thiết kế hướng đối tượng. Nó giúp ta giải quyết được những chương trình phức tạp.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm không được kế thừa

I.7. Sự kế thừa và mối quan hệ loại

1.2. Sự sử dụng lại

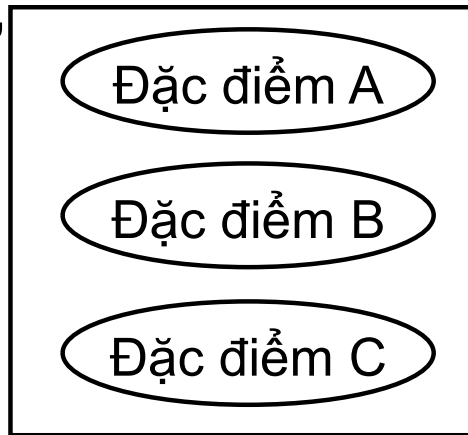
- Những người lập trình đã tìm nhiều cách để tránh viết lại mã đã có:
 - Copy mã từ một chương trình đã có sang một chương trình mới rồi sửa để nó có thể chạy được trong chương trình mới này. Công việc này thường gây ra rất nhiều lỗi và mất nhiều thời gian để sửa lỗi.
 - Tạo các hàm để trong các thư viện hàm để khi sử dụng không cần thay đổi. Đây là giải pháp tốt nhưng khi chuyển sang môi trường lập trình mới các hàm này vẫn phải thay đổi thì mới dùng được.

1.2. Sự sử dụng lại (tiếp)

- Giải pháp tốt nhất đã xuất hiện trong OOP, đó là sử dụng thư viện lớp. Bởi vì một lớp mô phỏng được các thực thể thế giới thực và để sử dụng trong môi trường mới nó cần ít thay đổi hơn các hàm. Quan trọng hơn cả là OOP cho phép thay đổi một lớp mà không cần thay đổi mã của nó: sử dụng kế thừa để rút ra một lớp từ một lớp đã có. Lớp đã có (lớp cơ sở) không bị thay đổi, còn lớp mới (lớp rút ra từ lớp đã có, lớp dẫn xuất) có thể sử dụng tất cả đặc điểm của lớp đã có và thêm vào những đặc điểm của riêng nó.

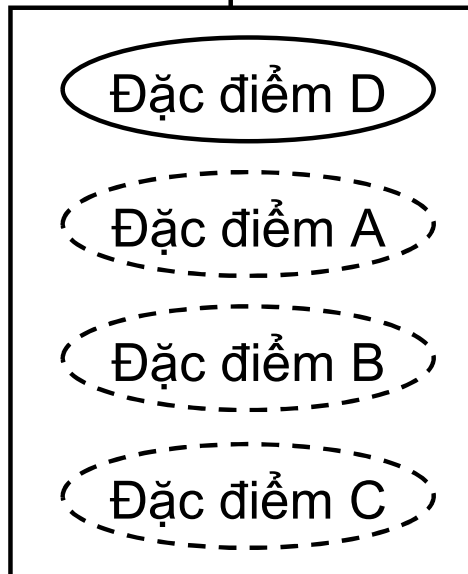
I.2. Sự sử dụng lại (tiếp)

Lớp cơ sở



Sự kế thừa

Lớp dẫn xuất



} Được định nghĩa trong lớp dẫn xuất

} Được định nghĩa trong lớp cơ sở nhưng có thể truy nhập từ lớp dẫn xuất

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm “không được kế thừa”

I.7. Sự kế thừa và mối quan hệ loại

I.3. Sự kế thừa và thiết kế hướng đối tượng

- Sự kế thừa giúp cho việc thiết kế chương trình được linh động hơn, phản ánh mối quan hệ trong thế giới thực chính xác hơn.
- Trong lập trình hướng đối tượng có 2 mối quan hệ giữa các thành phần của chương trình:
 - Mối quan hệ “có”: Một công nhân có tên, mã số, lương,... Một chiếc xe đạp có khung, 2 bánh, tay lái,... Mối quan hệ “có” trong thế giới thực có thể mô phỏng trong chương trình hướng đối tượng bằng một lớp, trong lớp có các thành viên của lớp. Lớp công nhân có chứa một biến lưu trữ tên, một biến lưu trữ mã số, một biến lưu trữ lương; lớp xe đạp có một đối tượng khung, hai đối tượng bánh, một đối tượng tay lái. Mối quan hệ có được các ngôn ngữ thủ tục (C, Pascal) mô phỏng bằng cấu trúc (struct), bản ghi (record). Mối quan hệ “có” được gọi là sự hợp thành.

I.3. Sự kế thừa và thiết kế hướng đối tượng (tiếp)

- Mỗi quan hệ “loại”: Xe đạp đua, xe đạp địa hình, xe đạp thiếu nhi đều là các loại xe đạp. Tất cả các loại xe đạp đều có đặc điểm chung: hai bánh, một khung. Một xe đạp đua, ngoài các đặc điểm chung này còn có đặc điểm là lốp nhỏ và nhẹ. Một xe đạp địa hình, ngoài các đặc điểm chung của một xe đạp còn có lốp to, dày và phanh tốt. Mỗi quan hệ “loại” này được mô phỏng trong chương trình hướng đối tượng bằng sự kế thừa. Ở đây, những gì là chung, khái quát được mô tả bằng một lớp cơ sở, những gì cụ thể, rõ ràng được mô tả bằng một lớp dẫn xuất. Sự kế thừa là một công cụ rất hữu ích trong thiết kế chương trình hướng đối tượng.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm không được kế thừa

I.7. Sự kế thừa và mối quan hệ loại

I.4. Cú pháp kế thừa

```
class TenLopCoSo
```

```
{
```

```
    //Cac thanh vien cua lop co so
```

```
};
```

```
class TenLopDanXuat : public TenLopCoSo
```

```
{
```

```
    //Cac thanh vien cua lop dan xuat
```

```
};
```

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm không được kế thừa

I.7. Sự kế thừa và mối quan hệ loại

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

- Những gì được kế thừa? Tất cả dữ liệu và hàm thành viên của lớp cơ sở, tức là một đối tượng lớp dẫn xuất kế thừa tất cả dữ liệu và hàm thành viên của lớp cơ sở.
- Truy nhập dữ liệu lớp cơ sở từ lớp dẫn xuất: Mặc dù các đối tượng lớp dẫn xuất chứa các thành viên dữ liệu được định nghĩa trong lớp cơ sở nhưng trong lớp dẫn xuất ta không thể truy nhập các thành viên dữ liệu **private** của lớp cơ sở (trừ các thành viên **public** và **protected**).

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

- Gọi hàm thành viên lớp cơ sở từ lớp dẫn xuất:

- Nếu các hàm thành viên lớp cơ sở và dẫn xuất có tên khác nhau thì khi gọi hàm ta chỉ dùng tên hàm.

- Nếu các hàm thành viên lớp cơ sở và dẫn xuất có tên giống nhau thì khi gọi hàm ta gắn thêm tên lớp cơ sở trước tên hàm bằng toán tử ::

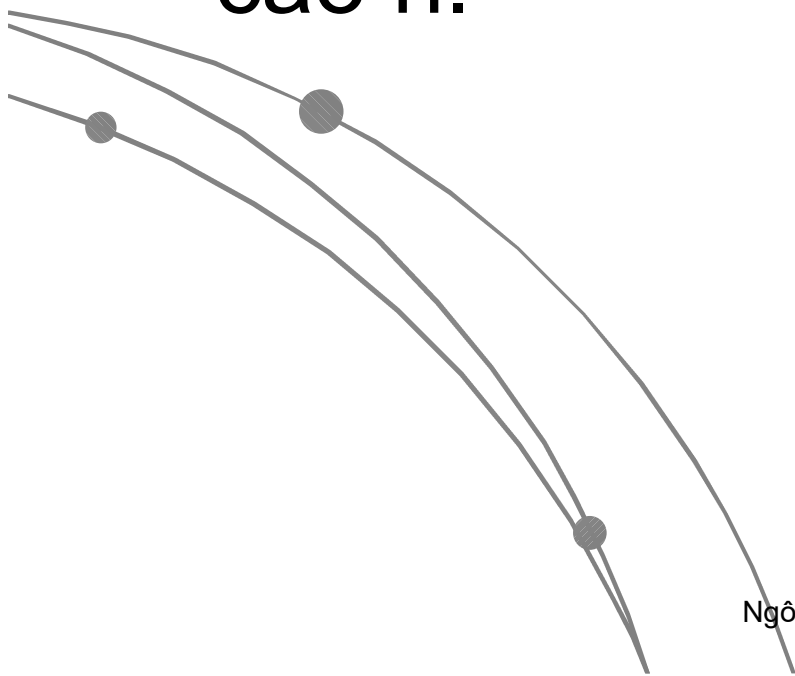
Tên_lớp_cơ_sở::Tên_thành_viên

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

- Trong OOP, các hàm thành viên lớp cơ sở và lớp dẫn xuất làm các công việc tương tự nhau luôn được chồng hàm (trùng) để cho rõ ràng và dễ nhớ.
- Nếu sử dụng chồng hàm thì khi gọi hàm thành viên lớp cơ sở phải gắn với tên lớp, nếu không trình biên dịch sẽ hiểu là gọi hàm thành viên lớp dẫn xuất.

Ví dụ

- Viết chương trình tính thể tích và diện tích bề mặt của hình trụ có bán kính r và chiều cao h . Biết rằng hình trụ là một loại hình tròn có bán kính r được kéo dài với chiều cao h .



Bài tập về nhà

- Viết chương trình quản lý nhân sự của một trường đại học. Nhân sự chia làm 3 loại: Giáo viên, Cán bộ quản lý và nhân viên phục vụ. Thông tin lưu trữ về giáo viên gồm có: tên, mã số, môn học giảng dạy. Thông tin lưu trữ về Cán bộ quản lý gồm có tên, mã số và chức vụ. Thông tin lưu trữ về Nhân viên phục vụ gồm có tên và mã số. Nhập vào một số nhân sự và đưa ra màn hình các nhân sự đã nhập.

Chương 7. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm đặc biệt của lớp cơ sở

I.7. Sự kế thừa và mối quan hệ loại

I.7. Các hàm không được kế thừa

- Có một vài hàm đặc biệt của lớp cơ sở không dùng được trong lớp dẫn xuất. Đó là các hàm làm những công việc cho riêng lớp cơ sở và lớp dẫn xuất. Có 3 hàm như vậy: hàm chồng toán tử gán =, hàm tạo (constructor) và hàm hủy (destructor).
- Hàm tạo lớp cơ sở phải tạo dữ liệu lớp cơ sở, hàm tạo lớp dẫn xuất phải tạo dữ liệu lớp dẫn xuất. Bởi vì các hàm tạo lớp cơ sở và lớp dẫn xuất tạo dữ liệu khác nhau nên chúng không thể thay thế nhau được. Do đó các hàm tạo không được tự động kế thừa.

I.7. Các hàm không được kế thừa

- Toán tử gán = trong lớp dẫn xuất phải gán các giá trị cho dữ liệu của lớp dẫn xuất, toán tử = trong lớp cơ sở phải gán các giá trị cho dữ liệu của lớp cơ sở. Đây là các công việc khác nhau, bởi vậy toán tử này cũng không được tự động kế thừa.
- Hàm hủy lớp dẫn xuất hủy dữ liệu của lớp dẫn xuất. Nó không hủy các đối tượng lớp cơ sở; nó phải gọi hàm hủy lớp cơ sở để làm việc này. Hơn nữa, các hàm hủy này làm các công việc khác nhau nên chúng không được tự động kế thừa.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

I.1. Tầm quan trọng của kế thừa trong OOP

I.2. Sự sử dụng lại

I.3. Sự kế thừa và thiết kế hướng đối tượng

I.4. Cú pháp kế thừa

I.5. Truy nhập thành viên lớp cơ sở từ lớp dẫn xuất

I.6. Các hàm không được kế thừa

I.7. Sự kế thừa và mối quan hệ loại

I.8. Sự kế thừa và mối quan hệ loại

- Sự kế thừa trong LTHĐT thể hiện được mối quan hệ “loại” trong thế giới thực. Các đối tượng lớp dẫn xuất là một loại đối tượng lớp cơ sở.
- Trong C++, ta có thể gán một đối tượng lớp dẫn xuất cho một đối tượng lớp cơ sở và có thể truyền đối tượng lớp dẫn xuất cho một hàm có đối số lớp cơ sở. Tuy nhiên ta không nên làm điều này vì đối tượng lớp dẫn xuất thường có kích thước lớn hơn đối tượng lớp cơ sở.

I.8. Sự kế thừa và mối quan hệ loại

```
class alpha                                //lớp cơ sở
{
    public:
        void memfunc()                    //hàm thành viên public
        {}
};
class beta:public alpha                    //lớp dẫn xuất
{
};
void main()
{
    void anyfunc(alpha);                  //khai báo, hàm có một đối số
    alpha aa;                             //đối tượng kiểu alpha
    beta bb;                              //đối tượng kiểu beta
    aa=bb;                                //đối tượng beta được gán cho biến alpha
    anyfunc(bb);                          //đối tượng beta được truyền như đối số alpha
}
```


Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

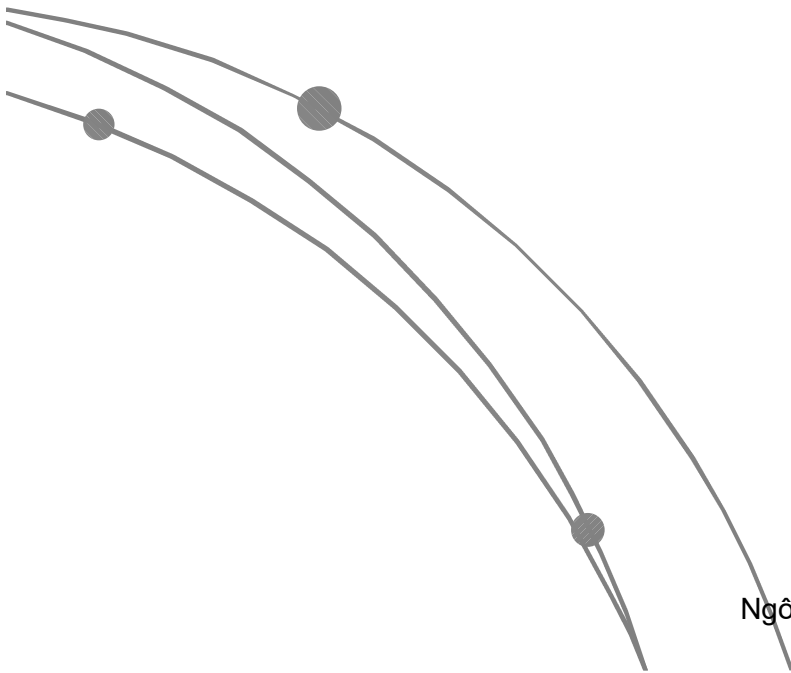
IV. Kế thừa nhiều mức

V. Hợp thành và kế thừa

VI. Kế thừa bội

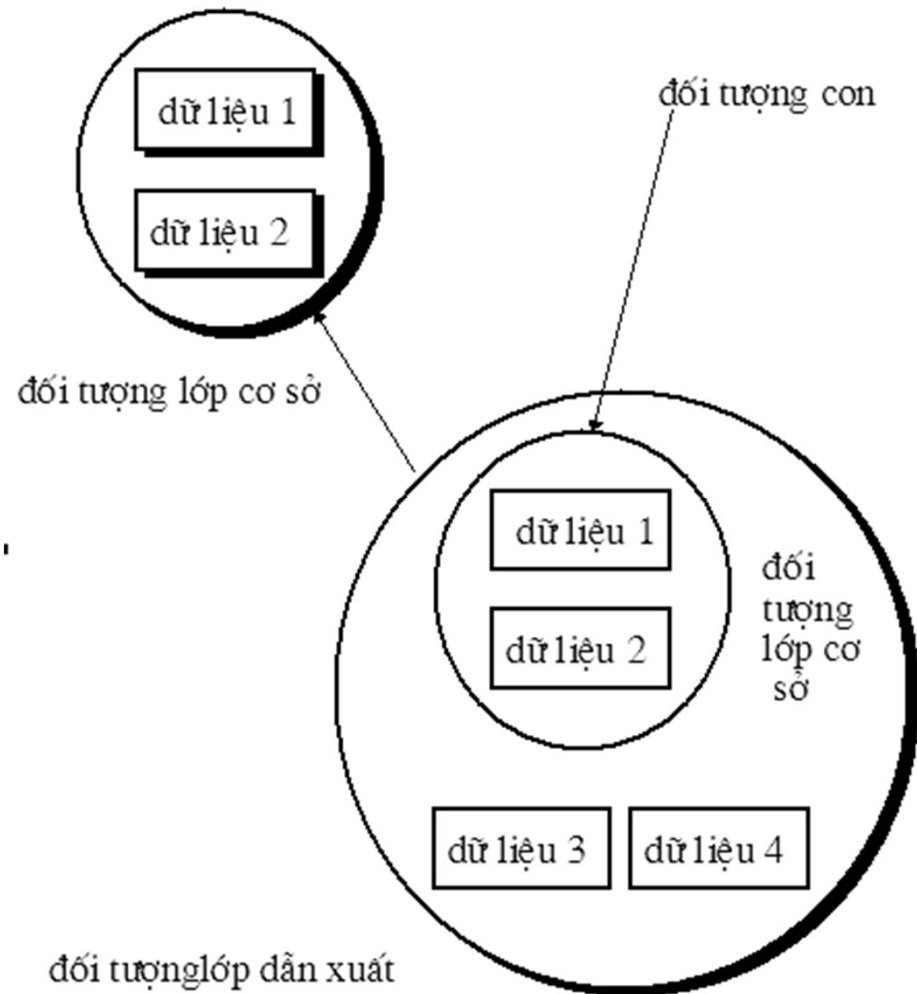
II. Hàm tạo, hàm hủy và sự kế thừa

1. Hàm tạo, hàm hủy với đối tượng lớp dẫn xuất
2. Khi nào phải viết hàm tạo cho lớp dẫn xuất



II.1. Hàm tạo, hàm hủy với đối tượng lớp dẫn xuất

- Khi chúng ta định nghĩa một đối tượng lớp dẫn xuất, không chỉ hàm tạo của nó được thực hiện mà hàm tạo trong lớp cơ sở cũng được thực hiện. Trên thực tế, hàm tạo lớp cơ sở được thực hiện trước. Bởi vì đối tượng lớp cơ sở là đối tượng con - một phần - của đối tượng lớp dẫn xuất, và chúng ta cần tạo các bộ phận trước khi tạo toàn thể.



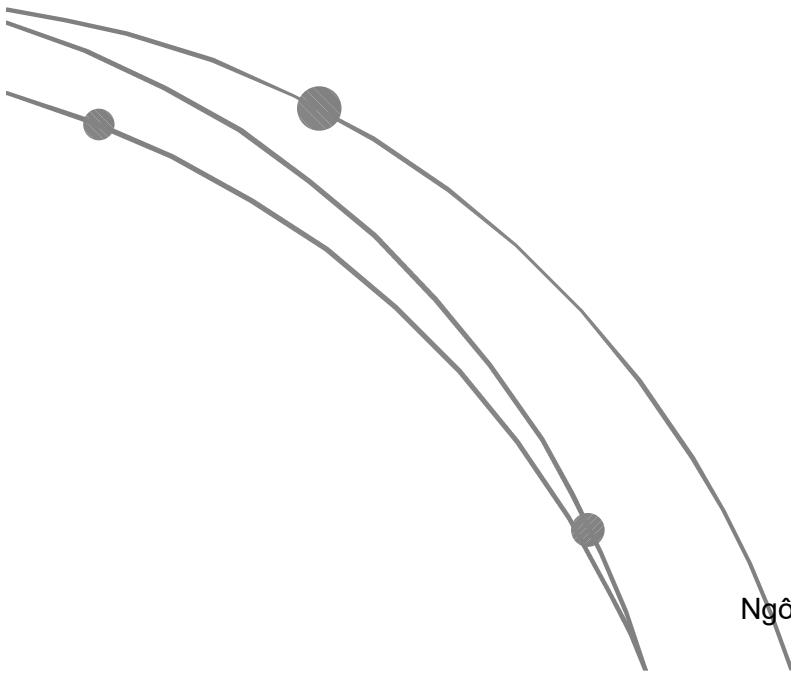
II.1. Hàm tạo, hàm hủy với đối tượng lớp dẫn xuất

- Như vậy, khi tạo đối tượng lớp dẫn xuất, các hàm tạo lớp cơ sở được gọi trước sau đó hàm tạo lớp dẫn xuất mới được gọi.
- Khi đối tượng bị hủy, hàm hủy được gọi theo thứ tự ngược lại: đối tượng lớp dẫn xuất được hủy trước sau đó đến đối tượng lớp cơ sở.
- Trong danh sách khởi tạo của hàm tạo lớp dẫn xuất có thể gọi hàm tạo lớp cơ sở. Ví dụ:
danxuat(int n):coso(n)

{ }

II. Hàm tạo, hàm hủy và sự kế thừa

1. Hàm tạo, hàm hủy với đối tượng lớp dẫn xuất
2. Khi nào phải viết hàm tạo cho lớp dẫn xuất



II.2. Khi nào phải viết hàm tạo cho lớp dẫn xuất

- Chúng ta không thể tạo đối tượng bằng một hàm tạo mà không có trong lớp của đối tượng đó. Bởi vậy, ngay cả khi chúng ta có một hàm tạo n đối số trong lớp cơ sở thì chúng ta vẫn phải định nghĩa một hàm tạo $\geq n$ đối số trong lớp dẫn xuất.
- Bất kỳ khi nào chúng ta cần hàm tạo có đối số để tạo đối tượng lớp dẫn xuất thì ta phải viết hàm tạo này trong lớp dẫn xuất.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

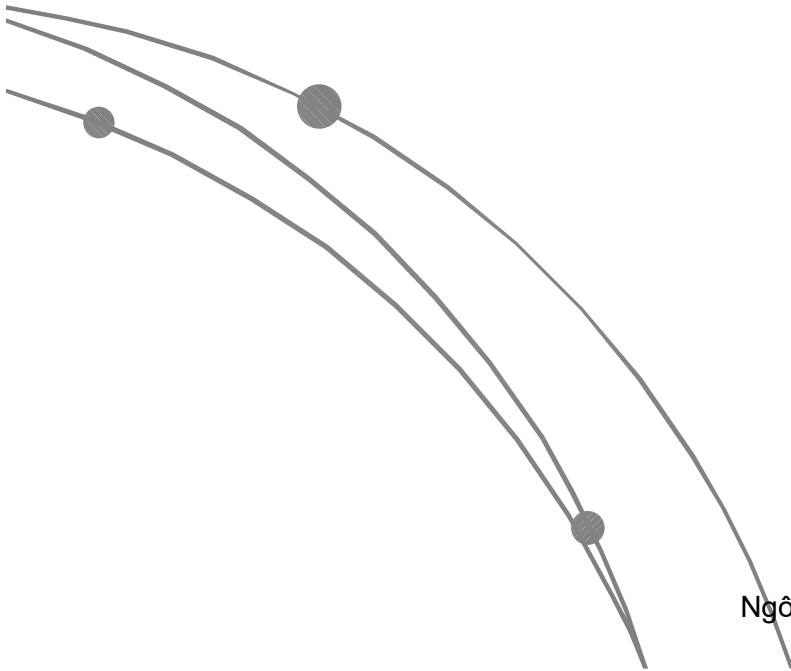
IV. Kế thừa nhiều mức

V. Hợp thành và kế thừa

VI. Kế thừa bội

III. Điều khiển việc truy nhập lớp cơ sở

1. Các định danh truy nhập
2. Che giấu dữ liệu lớp cơ sở



III.1. Các định danh truy nhập

- Khi chưa có sự kế thừa, các hàm thành viên lớp có thể truy nhập tới tất cả những gì có trong lớp dù nó là public hay private, nhưng bên ngoài lớp đó chỉ có thể truy nhập tới các thành viên public.
- Khi sự kế thừa xuất hiện, khả năng truy nhập mở rộng hơn cho lớp dẫn xuất. Các hàm thành viên của lớp dẫn xuất có thể truy nhập các thành viên public và protected của lớp cơ sở nhưng vẫn không thể truy nhập các thành viên private. Bên ngoài lớp dẫn xuất vẫn chỉ có thể truy nhập các thành viên public của lớp dẫn xuất.

III.1. Các định danh truy nhập

Sự kế thừa và khả năng truy nhập

Định danh truy nhập	có thể truy nhập từ trong lớp	có thể truy nhập từ lớp dẫn xuất	có thể truy nhập từ bên ngoài lớp
public	có	có	có
protected	có	có	không
private	có	không	không

III.2. Che giấu dữ liệu lớp cơ sở

- Dữ liệu trong lớp cơ sở nên để ở private hay protected? Để dữ liệu ở protected có thuận lợi là không cần viết thêm các hàm lớp cơ sở để truy nhập dữ liệu từ lớp dẫn xuất. Tuy nhiên, đây không phải là cách tốt nhất.
- Nói chung, dữ liệu lớp nên để private (trừ một số trường hợp đặc biệt). Dữ liệu public có thể bị thay đổi bởi bất kỳ hàm nào ở bất kỳ đâu trong chương trình, điều này nên tránh. Dữ liệu protected có thể bị thay đổi bởi các hàm trong bất kỳ lớp dẫn xuất nào.
- Bất kỳ người nào rút ra một lớp từ một lớp khác đều có quyền truy nhập tới dữ liệu **protected** của lớp đó. Sẽ an toàn và tin cậy hơn nếu lớp dẫn xuất không thể truy nhập trực tiếp dữ liệu lớp cơ sở.

III.2. Che giấu dữ liệu lớp cơ sở

- Một giao diện lớp bao gồm các hàm dùng để truy nhập cái gì đó. Thiết kế các lớp cho chúng ta hai giao diện: một giao diện **public** để bên ngoài lớp sử dụng và một giao diện **protected** để các lớp dẫn xuất sử dụng. Không nên để giao diện nào truy nhập trực tiếp dữ liệu. Một thuận lợi của việc để dữ liệu lớp cơ sở ở **private** là chúng ta có thể thay đổi nó mà không làm ảnh hưởng tới các lớp dẫn xuất.
- Để có truy nhập dữ liệu lớp cơ sở, chúng ta viết thêm các hàm thành viên cho lớp cơ sở. Các hàm này nên để ở **protected**, bởi vì chúng là phần giao diện **protected** được dùng bởi lớp dẫn xuất và các lớp không phải là lớp dẫn xuất không thể truy nhập được.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

IV. Kế thừa nhiều mức

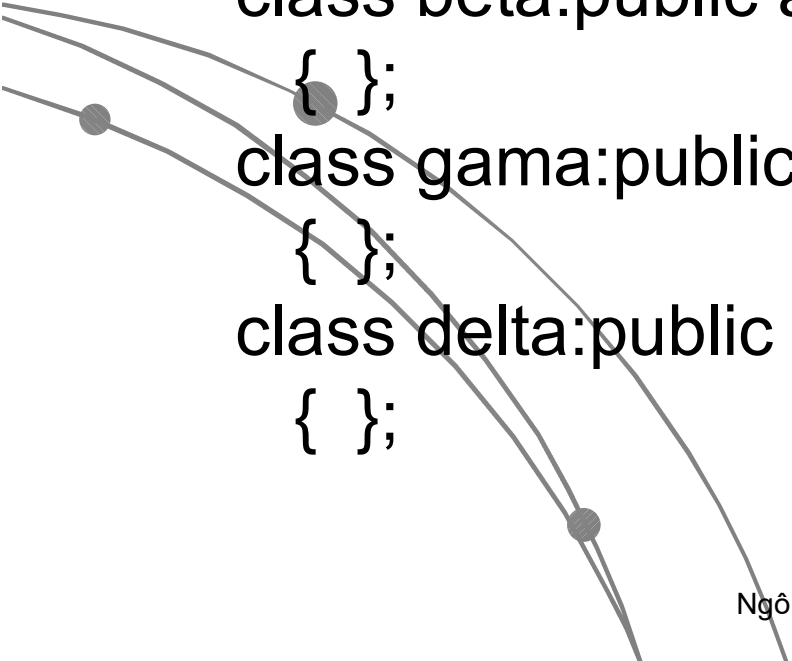
V. Hợp thành và kế thừa

VI. Kế thừa bội

IV. Kế thừa nhiều mức

- Sự kế thừa nhiều mức có nghĩa là không chỉ lớp **beta** có thể rút ra từ lớp **alpha**, lớp **gama** cũng có thể rút ra từ **beta**, lớp **delta** có thể rút ra từ **gama** v.v...

```
class alpha
{ };
class beta:public alpha
{ };
class gama:public beta
{ };
class delta:public gama
{ };
```



IV. Kế thừa nhiều mức

- Một lớp có thể truy nhập tới tất cả các lớp tổ tiên của nó, cụ thể là các hàm thành viên **delta** có thể truy nhập tới dữ liệu **public** hoặc **protected** trong **gama**, **beta**, và **alpha**. Tất nhiên là chúng không thể truy nhập tới các thành viên **private** của bất kỳ lớp nào trừ của chính nó.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

IV. Kế thừa nhiều mức

V. Hợp thành và kế thừa

VI. Kế thừa bội

V. Hợp thành và kế thừa

- Sự hợp thành là đặt một đối tượng bên trong một đối tượng khác hay, đứng về phía lập trình, là định nghĩa một đối tượng của một lớp ở bên trong mô tả của một lớp khác.

```
class alpha
{ };
class beta
{
private:
alpha obj;
};
```

V. Hợp thành và kế thừa

- Khi nào sử dụng sự kế thừa tốt hơn sự hợp thành?
 - Khi cần mối quan hệ “loại” giữa các lớp.
 - Khi cần một mảng đối tượng lớp cơ sở để chứa các đối tượng của các lớp dẫn xuất.
 - Khi cần một mảng con trỏ lớp cơ sở để chứa địa chỉ các đối tượng của các lớp dẫn xuất.

Chương 15. Sự kế thừa

I. Giới thiệu về kế thừa

II. Hàm tạo, hàm huỷ và sự kế thừa

III. Điều khiển việc truy nhập lớp cơ sở

IV. Kế thừa nhiều mức

V. Hợp thành và kế thừa

VI. Kế thừa bội

VI. Kế thừa bội

- Kế thừa bội có nghĩa là một lớp dẫn xuất kế thừa từ hai hay nhiều lớp cơ sở khác nhau.

```
class Base1
```

```
{ };
```

```
class Base2
```

```
{ };
```

```
class Derv:public Base1,public Base2
```

```
{ };
```

Bài tập

- Thời điểm là một loại ngày tháng và cũng là một loại thời gian nhưng có thêm địa điểm. Ngày tháng có ngày, tháng, năm. Thời gian có giờ và phút. Nhập vào một thời điểm, đưa ra ngày và thời gian của thời điểm đó ở dạng Địa điểm, dd/mm/yy - h:mm.
- Ví dụ: Hà Nội, 30/09/19 – 14:27