**CHƯƠNG 20 : LỚP DẪN XUẤT**

**1. Giới thiệu**

Từ Simula, C ++ đã vay mượn ý tưởng của các lớp và cấu trúc phân cấp lớp. Ngược lại, việc sử dụng các tính năng ngôn ngữ hỗ trợ cho các ý tưởng thiết kế sẽ phân biệt việc sử dụng C ++ hiệu quả. Một khái niệm (ý tưởng, khái niệm, v.v.) không tồn tại một cách cô lập. Ví dụ, cố gắng giải thích ô tô là gì. Bạn sẽ sớm giới thiệu các khái niệm về bánh xe, động cơ, người lái xe, người đi bộ, xe tải, xe cứu thương, đường, dầu, vé chạy quá tốc độ, nhà nghỉ, v.v. Vì chúng tôi sử dụng các lớp để biểu thị các khái niệm nên vấn đề trở thành cách thể hiện mối quan hệ giữa các khái niệm.

Khái niệm về một lớp dẫn xuất thể hiện các mối quan hệ thứ bậc, nghĩa là, để thể hiện tính chung giữa các lớp. **Ví dụ**, các khái niệm về hình tròn và hình tam giác có liên quan với nhau ở chỗ chúng đều là hình dạng; nghĩa là, họ có chung khái niệm về một hình dạng. Do đó, chúng ta định nghĩa rõ ràng lớp Circle và lớp Triangle để có chung lớp Shape. Trong trường hợp đó, lớp chung, ở đây Shape, được gọi là lớp cơ sở hoặc siêu lớp và các lớp dẫn xuất từ ​​lớp đó, ở đây Circle và Triangle, được gọi là lớp dẫn xuất hoặc lớp con. Chương này là sự khám phá ý nghĩa của ý tưởng đơn giản này, là cơ sở cho cái thường được gọi là lập trình hướng đối tượng. Các tính năng ngôn ngữ hỗ trợ xây dựng các lớp mới từ những lớp hiện có:

• Kế thừa triển khai: để tiết kiệm nỗ lực triển khai bằng cách chia sẻ các cơ sở được cung cấp bởi một lớp cơ sở

• Kế thừa giao diện: cho phép các lớp dẫn xuất khác nhau được sử dụng thay thế cho nhau thông qua giao diện được cung cấp bởi một lớp cơ sở chung

Kế thừa giao diện thường được gọi là đa hình thời gian chạy (hoặc đa hình động). Ngược lại, việc sử dụng thống nhất các lớp không liên quan đến kế thừa được cung cấp bởi các mẫu, thường được gọi là đa hình thời gian biên dịch (hoặc đa hình tĩnh).

**2 .Lớp Dẫn Xuất**

**Employee**

**↑**

**Manager**

Một lớp dẫn xuất thường được cho là kế thừa các thuộc tính từ cơ sở của nó, còn được gọi là kế thừa. Một lớp cơ sở được gọi là lớp cha và lớp dẫn xuất là lớp con.

Ta có một chương trình định nghĩa cấu trúc nhân viên:

struct Employee{

string name,

char middle\_initial;

}

Và định nghĩa cấu trúc người quảnn lý

struct Manager {

Employee emp;

list<Employee\*> group;

}

Hoặc sơ đồ:

|  |
| --- |
| first\_name  family\_name  ... |

**Employee: Manager:**

|  |
| --- |
| first\_n*ame*  *family*\_name  ... |
| group  level  ... |

**3. Hàm thành viên**

Một hàm thành viên của một lớp dẫn xuất có thể truy cập vào các thành viên riêng của lớp cơ sở của nó. Thành viên riêng sẽ trở nên vô nghĩa nếu cho phép một lập trình viên truy cập vào phần riêng của một lớp chỉ đơn giản bằng cách dẫn xuất một lớp mới từ nó

**4. Phân cấp lớp**

Bản thân một lớp dẫn xuất có thể là một lớp cơ sở. Ví dụ:

**Class Employee{/ \* ... \* /};**

**Class Manager: public Employee {/ \* ... \* /};**

**Class Director: public Manager {/ \* ... \* /};**

**20.3.2 Chức năng ảo**

Các hàm ảo khắc phục các vấn đề với giải pháp trường kiểu bằng cách cho phép lập trình viên khai báo các hàm trong một lớp cơ sở có thể được định nghĩa lại trong mỗi lớp dẫn xuất.

**class Employee {**

**public:**

**Employee(const string& name, int dept);**

**virtual void print() const;**

**// ...**

**private:**

**string first\_name , family\_name;**

**short depar tment;**

**// ...};**

Một hàm ảo có thể được sử dụng ngay cả khi không có lớp nào được dẫn xuất từ lớp của nó và một lớp dẫn xuất không cần phiên bản riêng của hàm ảo thì không cần cung cấp một lớp.Ví dụ:

**class Manager : public Employee {**

**public:**

**Manager(const string& name, int dept, int lvl);**

**void print() const;**

**// ...**

**private:**

**list<Employee∗> group;**

**short level;**

**// ...};**

**void Manager::print() const{**

**Employee::print();**

**cout << "\tlevel " << level << '\n';**

**// ...}**

**5. Kiểm soát truy cập**

Thành viên của một lớp có thể là riêng tư, được bảo vệ hoặc công khai:

• Nếu nó là private, tên của nó chỉ có thể được sử dụng bởi các hàm thành viên và bạn bè của lớp mà nó được khai báo.

• Nếu nó được bảo vệ, tên của nó chỉ có thể được sử dụng bởi các hàm thành viên và bạn bè của lớp mà nó được khai báo và bởi các hàm thành viên và bạn bè của các lớp dẫn xuất từ lớp này .

• Nếu nó là công khai, tên của nó có thể được sử dụng bởi bất kỳ chức năng nào.

người dùng chung

các hàm thành viên và bạn bè của lớp dẫn xuất

chức năng thành viên riêng và bạn bè

public:

protected:

private:

**6.Quyền truy cập vào các lớp cơ sở**

Giống như một thành viên, một lớp cơ sở có thể được khai báo là riêng tư, được bảo vệ hoặc công khai. Ví dụ:

class X : public B { /\* ... \*/ };

class Y : protected B { /\* ... \*/ };

class Z : private B { /\* ... \*/

Các chỉ số truy cập khác nhau phục vụ các nhu cầu thiết kế khác nhau:

• công khai dẫn xuất làm cho lớp dẫn xuất trở thành một kiểu con của cơ sở của nó. Ví dụ, X là một loại B. Đây là dạng dẫn xuất phổ biến nhất.

• Các cơ sở riêng hữu ích nhất khi xác định một lớp bằng cách giới hạn giao diện cho một cơ sở để có thể cung cấp các đảm bảo mạnh mẽ hơn. Ví dụ, B là một chi tiết triển khai của Z .

• Các cơ sở được bảo vệ rất hữu ích trong các cấu trúc phân cấp lớp trong đó dẫn xuất thêm là tiêu chuẩn. Giống như dẫn xuất riêng, dẫn xuất được bảo vệ được sử dụng để biểu diễn các chi tiết triển khai.. Ví dụ:

class XX : B { /\* ... \*/ }; // B is a private base

struct YY : B { /\* ... \*/ }; // B is a public base