NGÔN NGỮ LẬP TRÌNH

Bài 6: Kế thừa

Giảng viên: Lý Anh Tuấn

Email: tuanla@tlu.edu.vn

Nội dung

1. Cơ bản về kế thừa
   - Lớp dẫn xuất với hàm tạo
   - Bổ từ protected
   - Định nghĩa lại hàm thành viên
   - Hàm không được kế thừa
2. Lập trình với kế thừa
   - Toán tử gán và hàm tạo sao chép
   - Hàm hủy trong các lớp dẫn xuất
   - Đa kế thừa

Giới thiệu kế thừa

- Lập trình hướng đối tượng
  - Cung cấp kỹ thuật phân đoạn trừu tượng gọi là kế thừa
- Định nghĩa dạng khái quát của lớp
  - Phiên bản chuyên biệt sau đó kế thừa các tính chất của lớp khái quát
  - Và thêm vào hoặc sửa đổi các chức năng để phù hợp với việc sử dụng của nó

Cơ bản về kế thừa

- Lớp mới được kế thừa từ một lớp khác
- Lớp cơ sở
  - Lớp khái quát được các lớp khác dẫn xuất
- Lớp dẫn xuất
  - Lớp mới
  - Tự động bao gồm các biến thành viên và các hàm thành viên của lớp cơ sở
  - Sau đó có thể thêm vào các hàm và các biến thành viên

Lớp dẫn xuất

- Xét ví dụ:
  Lớp nhân viên "Employees"
- Bao gồm:
  - Các nhân viên hưởng lương theo năm
  - Các nhân viên làm việc theo giờ
- Các tập này là tập con của nhân viên
  - Có thể bao gồm cả tập các nhân viên hưởng lương theo tháng hoặc theo tuần

Lớp dẫn xuất

- Không cần kiểu "employee" tổng quát
  - Vì không có ai chỉ đơn thuần là một "employee"
- Khái niệm nhân viên tổng quát rất có ý nghĩa
  - Tất cả đều có tên
  - Tất cả đều có số bảo hiểm xã hội
  - Các hàm kết hợp của các thông tin cơ bản này giống nhau với tất cả các nhân viên
- Lớp tổng quát có thể bao chứa tất cả những dữ liệu này về các nhân viên

Lớp Employee

- Nhiều thành viên của lớp "employee" áp dụng cho tất cả các kiểu nhân viên
  - Các hàm truy cập
  - Các hàm biến đổi
  - Phần lớn các mục dữ liệu
    - SSN
    - Name
    - Pay
  - Tuy nhiên chúng ta sẽ không có các đối tượng thuộc lớp này

Lớp Employee

- Xét hàm printCheck():
  - Luôn phải định nghĩa lại nó trong các lớp dẫn xuất
  - Do các kiểu nhân viên khác nhau có thể có séc ngân hàng khác nhau
  - Không thực sự có ý nghĩa với nhân viên chưa được tách biệt
  - Do vậy hàm printCheck() trong lớp Employee chỉ thực hiện công việc:
    - Đưa ra thông điệp lỗi: "printCheck called for undifferentiated employee!! Aborting…"

Dẫn xuất từ lớp Employee

- Các lớp được dẫn xuất từ lớp Employee:
  - Tự động bao gồm tất cả các biến thành viên
  - Tự động bao gồm tất cả các hàm thành viên
- Chúng ta nói rằng lớp dẫn xuất "kế thừa" các thành viên từ lớp cơ sở
- Sau đó có thể định nghĩa lại các thành viên đã có và thêm vào các thành viên mới

Giao diện của lớp dẫn xuất HourlyEmployee

Display 14.3 Interface for the Derived Class `HourlyEmployee`

```
1
2   //This is the header file hourlyemployee.h.
3   //This is the interface for the class HourlyEmployee.
4   #ifndef HOURLYEMPLOYEE_H
5   #define HOURLYEMPLOYEE_H

6   #include <string>
7   #include "employee.h"

8   using std::string;

9   namespace SavitchEmployees
10  {
```

Giao diện của lớp dẫn xuất HourlyEmployee

```
11      class HourlyEmployee : public Employee
12      {
13      public:
14          HourlyEmployee( );
15          HourlyEmployee(string theName, string theSsn,
16                                  double theWageRate, double theHours);
17          void setRate(double newWageRate);
18          double getRate( ) const;
19          void setHours(double hoursWorked);
20          double getHours( ) const;
21          void printCheck( ) ;
22      private:
23          double wageRate;
24          double hours;
25      };

26  }//SavitchEmployees

27  #endif //HOURLYEMPLOYEE_H
```

You only list the declaration of an inherited member function if you want to change the definition of the function.

Giao diện lớp HourlyEmployee

- Bắt đầu giống như các giao diện khác
  - Cấu trúc #ifndef
  - Bao gồm các thư viện cần thiết
  - Cũng bao gồm employee.h
- Đầu đề là:
  class HourlyEmployee : public Employee
  { …
  - Chỉ rõ được kế thừa công khai từ lớp Employee

Thêm vào lớp HourlyEmployee

- Giao diện lớp dẫn xuất chỉ liệt kê các thành viên mới hoặc được định nghĩa lại
  - Vì tất cả những thành viên được kế thừa khác đã được định nghĩa rồi
  - Tức là: tất cả các nhân viên đều có ssn, name, vân vân
- HourlyEmployee thêm vào
  - Các hàm tạo
  - Các biến thành viên wageRate, hours
  - Các hàm thành viên setRate(), getRate(), setHours(), getHours()

Định nghĩa lại lớp HourlyEmployee

- HourlyEmployee định nghĩa lại:
  - Hàm thành viên printCheck()
  - Hàm này nạp chồng thi hành hàm printCheck() từ lớp Employee
- Định nghĩa của nó phải nằm trong sự thi hành của lớp HourlyEmployee
  - Giống như các hàm thành viên khác được khai báo trong giao diện của HourlyEmployee

Thuật ngữ kế thừa

- Thường bắt trước các mối quan hệ gia đình
- Lớp cha
  - Chỉ lớp cơ sở
- Lớp con
  - Chỉ lớp dẫn xuất
- Lớp tổ tiên
  - Lớp là cha của cha …
- Lớp hậu duệ
  - Ngược lại với tổ tiên

Hàm tạo trong lớp dẫn xuất

- Hàm tạo lớp cơ sở không được kế thừa trong lớp dẫn xuất
  - Nhưng chúng có thể được gọi trong hàm tạo lớp dẫn xuất
- Hàm tạo lớp cơ sở phải khởi tạo tất cả các biến thành viên lớp cơ sở
  - Các biến này được kế thừa bởi lớp dẫn xuất
  - Hàm tạo lớp dẫn xuất cần gọi tới hàm tạo lớp cơ sở
    - Đây là công việc đầu tiên của hàm tạo lớp dẫn xuất

Ví dụ hàm tạo lớp dẫn xuất

- Xét cú pháp của hàm tạo HourlyEmployee:

```
HourlyEmployee::HourlyEmployee(string theName,
                               string theNumber, double theWageRate,
                               double theHours)
                 : Employee(theName, theNumber),
                 wageRate(theWageRate), hours(theHours)
{
      //Deliberately empty
}
```

- Phần sau : là phần khởi tạo
  - Bao gồm lời gọi tới hàm tạo Employee

Một hàm tạo HourlyEmployee khác

- Một hàm tạo khác:

```
HourlyEmployee::HourlyEmployee()
                : Employee(),   wageRate(0),
                                hours(0)
{
    //Deliberately empty
}
```

- Phiên bản mặc định của hàm tạo lớp cơ sở được gọi (không đối số)
- Luôn nên gọi một trong các hàm tạo lớp cơ sở

Hàm tạo: Không có lời gọi lớp cơ sở

- Hàm tạo lớp dẫn xuất luôn nên gọi đến một trong các hàm tạo lớp cơ sở
- Nếu bạn không làm điều này:
  - Hàm tạo mặc định lớp cơ sở sẽ tự động được gọi
- Định nghĩa hàm tạo tương đương:

```
HourlyEmployee::HourlyEmployee()
                              : wageRate(0), hours(0)
{ }
```

Lỗi thường gặp: Dữ liệu private lớp cơ sở

- Lớp dẫn xuất kế thừa các biến thành viên private
  - Nhưng vẫn không thể truy cập trực tiếp chúng
  - Ngay cả thông qua các hàm thành viên lớp dẫn xuất
- Các biến thành viên private chỉ có thể được truy cập bằng tên trong các hàm thành viên của lớp mà ở đó chúng được định nghĩa

Lỗi thường gặp: Hàm thành viên private lớp cơ sở

- Điều tương tự cũng xảy ra với các hàm thành viên lớp cơ sở
  - Không thể được truy cập bên ngoài giao diện và sự thi hành của lớp cơ sở
  - Thậm chí trong các định nghĩa hàm thành viên lớp dẫn xuất

Lỗi thường gặp: Hàm thành viên private lớp cơ sở

- Dễ mắc lỗi hơn so với các biến thành viên
  - Các biến thành viên có thể được truy cập gián tiếp bằng các hàm thành viên truy cập hoặc biến đổi
  - Các hàm thành viên đơn giản là không khả dụng
- Điều cần lưu ý
  - Hàm thành viên private chỉ nên là các hàm phụ trợ
  - Chỉ nên sử dụng chúng trong lớp chúng được định nghĩa

Bổ từ protected

- Là sự phân loại mới cho các thành viên lớp
- Cho phép truy cập bằng tên trong lớp dẫn xuất
  - Nhưng không thể truy cập ở nơi nào khác
  - Không được truy cập bằng tên trong các lớp khác
- Trong lớp nó được định nghĩa → hành động như private
- Xem như được bảo vệ trong lớp dẫn xuất
  - Cho phép các dẫn xuất trong tương lai
- Cảm giác như điều này vi phạm việc che dấu thông tin

Định nghĩa lại hàm thành viên

- Giao diện của lớp dẫn xuất:
  - Chứa các khai báo cho các hàm thành viên mới
  - Chứa các khai báo cho các hàm thành viên kế thừa được thay đổi
  - Các hàm thành viên kế thừa không được khai báo
- Sự thi hành của lớp dẫn xuất sẽ:
  - Định nghĩa các hàm thành viên mới
  - Định nghĩa lại các hàm kế thừa đã khai báo

Định nghĩa lại vs. Nạp chồng

- Rất khác nhau
- Định nghĩa lại trong lớp dẫn xuất
  - Danh sách tham số giống nhau
  - Về cơ bản là viết lại hàm tương tự
- Nạp chồng
  - Danh sách tham số khác nhau
  - Định nghĩa hàm mới nhận các tham số khác
  - Các hàm được nạp chồng phải có các ký hiệu khác nhau

Một ký hiệu hàm

- Định nghĩa của một ký hiệu bao gồm:
  - Tên hàm
  - Chuỗi các kiểu trong danh sách tham số
    - bao gồm thứ tự, số lượng, các kiểu
- Ký hiệu không bao gồm:
  - Kiểu trả về
  - Từ khóa const
  - &

Truy cập hàm cơ sở được định nghĩa lại

- Khi được định nghĩa lại trong lớp dẫn xuất, định nghĩa của lớp cơ sở không mất đi
- Có thể sử dụng nó theo cách sau:

```
Employee       JaneE;
HourlyEmployee SallyH;
JaneE.printCheck(); → gọi hàm printCheck  của Employee
SallyH.printCheck(); → gọi hàm printCheck của
                                                HourlyEmployee
SallyH.Employee::printCheck(); → gọi hàm printCheck của
                                                Employee
```

- Ở đây không có ý nghĩa, nhưng sẽ hữu ích trong một số trường hợp

Hàm không được kế thừa

- Tất cả các hàm thông thường trong lớp cơ sở được kế thừa trong lớp dẫn xuất
- Ngoại trừ:
  - Các hàm tạo (đã được xét)
  - Các hàm hủy
  - Hàm tạo sao chép
  - Toán tử gán

Toán tử gán và hàm tạo sao chép

- Toán tử gán được nạp chồng và hàm tạo sao chép không được kế thừa
  - Nhưng có thể được sử dụng trong các định nghĩa lớp dẫn xuất
  - Thường được sử dụng
  - Tương tự như cách hàm tạo lớp dẫn xuất gọi tới hàm tạo lớp cơ sở

Ví dụ toán tử gán

- Cho “Derived” được dẫn xuất từ “Base”:

```
Derived& Derived::operator =(const Derived & rightSide)
{
        Base::operator =(rightSide);
        …
}
```

- Lưu ý dòng lệnh
  - Gọi toán tử gán từ lớp cơ sở
    - Việc này bao gồm tất cả các biến thành viên được kế thừa
  - Sau đó thiết lập các biến mới của lớp dẫn xuất

Ví dụ hàm tạo sao chép

- Xét:

```
Derived::Derived(const Derived& Object)
                                    : Base(Object), …
{…}
```

- Sau : là lời gọi tới hàm tạo sao chép cơ sở
  - Thiết lập các biến thành viên kế thừa của đối tượng lớp dẫn xuất được tạo
  - Lưu ý Object thuộc kiểu Derived, nhưng nó cũng thuộc kiểu Base, do vậy đối số là đúng

Hàm hủy trong lớp dẫn xuất

- Nếu các hàm hủy lớp cơ sở chính xác:
  - Dễ dàng viết hàm hủy lớp dẫn xuất
- Khi hàm hủy lớp dẫn xuất được gọi:
  - Tự động gọi hàm hủy lớp cơ sở
  - Do vậy không cần lời gọi tường minh
- Do vậy hàm hủy lớp dẫn xuất chỉ cần quan tâm đến các biến lớp dẫn xuất
  - Và bất cứ dữ liệu nào chúng trỏ tới
  - Hàm hủy lớp cơ sở tự động xử lý dữ liệu được kế thừa

Thứ tự gọi hàm hủy

- Xét:
  lớp B dẫn xuất từ lớp A
  lớp C dẫn xuất từ lớp B

  A ← B ← C
- Khi đối tượng của lớp C đi ra ngoài phạm vi:
  - Hàm hủy lớp C được gọi trước nhất
  - Sau đó hàm hủy lớp B được gọi
  - Cuối cùng hàm hủy lớp A được gọi
- Ngược lại với thứ tự gọi các hàm tạo

Mối quan hệ "là một" và "có một"

- Sự kế thừa:
  - Được xem là mối quan hệ lớp "là một"
  - Ví dụ, HourlyEmployee "là một" Employee
  - Một Convertible "là một" Automobile
- Lớp bao gồm các đối tượng của một lớp khác như là dữ liệu thành viên của nó
  - Được xem là mối quan hệ lớp "có một"
  - Ví dụ, một lớp "có một" đối tượng của lớp khác là dữ liệu của nó

Kế thừa protected và private

- Các dạng kế thừa mới
  - Cả hai hiếm khi được sử dụng
- Kế thừa protected:

  class SalariedEmployee : protected Employee
  {…}
  - Các thành viên public trong lớp cơ sở trở thành protected trong lớp dẫn xuất
- Kế thừa private

  class SalariedEmployee : private Employee
  {…}
  - Tất cả thành viên trong lớp cơ sở trở thành private trong lớp dẫn xuất

Đa kế thừa

- Lớp dẫn xuất có thể có nhiều hơn một lớp cơ sở
  - Cú pháp bao gồm các lớp cơ sở được tách biệt bởi dấu phẩy:

    class derivedMulti : public base1, base2
    {…}
- Khả năng nhập nhằng là rất cao
- Chứa đựng nhiều rủi ro
  - Một số người cho rằng không nên sử dụng đa kế thừa
  - Chắc chắn chỉ nên được sử dụng bởi những người lập trình có kinh nghiệm

Tóm tắt

- Kế thừa cung cấp việc sử dụng lại mã lệnh
  - Cho phép một lớp dẫn xuất từ một lớp khác, cộng thêm các thuộc tính
- Các đối tượng lớp dẫn xuất kế thừa các thành viên lớp cơ sở
  - Và có thể thêm các thành viên
- Các biến thành viên private trong lớp cơ sở không thể được truy cập bằng tên trong lớp dẫn xuất
- Các hàm thành viên private không được kế thừa

Tóm tắt

- Có thể định nghĩa lại các hàm thành viên được kế thừa
  - Để thể hiện sự khác biệt trong lớp dẫn xuất
- Các thành viên protected trong lớp cơ sở
  - Có thể được truy cập bằng tên trong các hàm thành viên lớp dẫn xuất
- Toán tử gán được nạp chồng không được kế thừa
  - Nhưng có thể được gọi từ lớp dẫn xuất
- Các hàm tạo không được kế thừa
  - Được gọi từ hàm tạo của lớp dẫn xuất