Lab 12
Hàm ảo, liên kết động

Lập trình hướng đối tượng

Mục tiêu	1. Hiểu về hàm ảo và hàm thuần ảo 2. Hiểu về liên kết động 3. Hiểu tại sao hàm hủy đối tượng phải là hàm ảo

1 Hướng dẫn khởi đầu 1 – Hàm ảo (virtual)

Mô tả bài tập

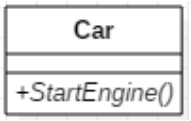
Mục tiêu hàm ảo là cung cấp **cài đặt mặc định** cho một hành vi.

Nếu lớp con không cài thì sẽ dùng cách hành xử lớp cha.

Nếu có cài thì sẽ dùng cách hành xử lớp con chứ không dùng hành xử mặc định.

Hướng dẫn cài đặt bước 1

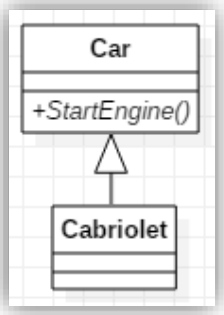
Xét lớp Xe hơi (**Car**) có cài đặt mặc định của hàm **StartEngine** như sau




```
class Car {
public:
    virtual void StartEngine() {
        cout << "Engine started!" << endl;
    }
};
```

Bước 2 – Mở rộng hệ thống – Sử dụng hành xử mặc định

Sau này ta mở rộng hệ thống, có lớp Xe hơi mui trần (**Cariolet**) kế thừa từ lớp **Car**




```
class Car {
public:
    virtual void StartEngine() {
        cout << "Engine started!" << endl;
    }
};

class Cabriolet : public Car {

};
```

Lúc này nếu ta tạo ra một xe hơi mui trần và khởi động động cơ

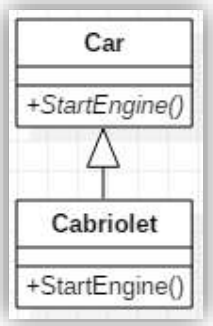
```
int main()
{
    Car* car = new Cabriolet();
    car->StartEngine();

    delete car;
}
```

Thì ta sẽ thấy kết quả:

```
Engine started!
```

Bước 3 – Nâng cấp hệ thống – Cài đặt cách hành xử riêng




```
class Car {
public:
    virtual void StartEngine() {
        cout << "Engine started!" << endl;
    }
};

class Cabriolet : public Car {
    void StartEngine() {
        cout << "Roof removed" << endl;
        Car::StartEngine();
    }
};

int main()
{
    Car* car = new Cabriolet();
    car->StartEngine();

    delete car;
}
```

Chạy lên để thấy kết quả như sau

```
Roof removed
Engine started!
```

Chú ý bên trong hàm StartEngine của lớp con đã gọi hàm StartEngine của lớp cha, và bởi vì cả cha và con đều có hàm StartEngine cùng tên nên để trình biên dịch biết gọi hàm này là từ cha hay con, ta sẽ thêm vào tên lớp cha ở phía trước!

Hệ quả là ta có: **Car**::**StartEngine**();

Người mới bắt đầu sẽ dễ bị nhầm lẫn vì cách viết này giống như gọi hàm tĩnh của lớp, tuy nhiên ý nghĩa hai việc này khác nhau nhé!

Ngoài ra có một việc cực kì quan trọng.

Cần coi lại hàm main:

```
int main()
{
    Car* car = new Cabriolet();
    car->StartEngine();

    delete car;
}
```

Biến car có kiểu là con trỏ lớp cha – Car*, lại giữ địa chỉ đến vùng nhớ có kiểu của lớp con – Cabriolet*.

Câu hỏi rất quan trọng là hàm car->StartEngine() gọi hàm ở lớp con hay lớp cha?

Dựa trên kết quả bạn thấy thì bạn kết luận được ngay là từ đâu?

Tất nhiên là hàm **StartEngine** được gọi là của lớp **con** - **Cabriolet**.

Thử sửa một chút khai báo của lớp cha – **Car**, hàm **StartEngine** lúc ban đầu:

```
class Car {
public:
    virtual void StartEngine() {
        cout << "Engine started!" << endl;
    }
};
```

Từ chỗ đang có từ khóa virtual, ta thử bỏ đi thành

```
class Car {
public:
    void StartEngine() {
        cout << "Engine started!" << endl;
    }
};
```

Chạy lại và ta thấy kết quả kì lạ:

```
Engine started!
```

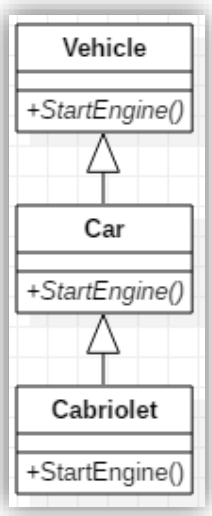
Kết quả này cho thấy hàm **StartEngine** được gọi lúc này là của **lớp cha**!!!

> Tóm lại từ khóa virtual tạo ra liên kết động cho con trỏ. Tùy thuộc con trỏ đang trỏ đến đối tượng cụ thể nào mà hành động tương ứng của lớp sẽ được gọi.

2 Hướng dẫn khởi đầu 2 – Hàm thuần ảo (pure virtual)

Hàm thuần ảo là hàm ảo nhưng chỉ có phần khai báo mà không có cài đặt. Việc cài đặt cụ thể hoàn toàn phó thác cho lớp kế thừa.

Cải tiến thiết kế về xe hơi ở trên ta sẽ có một lớp mô tả thông tin về phương tiện di chuyển tổng quát (**Vehicle**) như sau:




```
class Vehicle {
public:
    virtual void StartEngine() = 0;
};

class Car: public Vehicle {
public:
    void StartEngine() {
        cout << "Engine started!" << endl;
    }
};

class Cabriolet : public Car {
    void StartEngine() {
        cout << "Roof removed" << endl;
        Car::StartEngine();
    }
};
```

Hàm main như sau:

```
int main()
{
    Vehicle* car = new Cabriolet();
    car->StartEngine();

    delete car;
}
```

Kết quả:

```
Roof removed
Engine started!
```

Một lần nữa chú ý từ khóa virtual của lớp cha giúp cho con trỏ car gọi thực thi hàm đúng của lớp con.

3 Hướng dẫn khởi đầu 3 – Hàm hủy ảo

Nếu hiểu được từ khóa virtual giúp con trỏ lớp cha gọi hàm ứng với kiểu dữ liệu mà nó đang giữ thì hàm hủy ảo cũng tương tự, nhờ có từ khóa virtual mà con trỏ lớp cha gọi hàm hủy của lớp con mà nó đang trỏ tới.

Xét ví dụ như mô tả bên dưới (sử dụng mã nguồn ở hướng dẫn trước, chỉ thêm hàm hủy)

```
class Vehicle {
public:
    virtual void StartEngine() = 0;
};

class Car: public Vehicle {
public:
    void StartEngine() {
        cout << "Engine started!" << endl;
    }

    ~Car() {
        cout << "Car is deleted" << endl;
    }
};

class Cabriolet : public Car {
public:
    void StartEngine() {
        cout << "Roof removed" << endl;
        Car::StartEngine();
    }

    ~Cabriolet() {
        cout << "Cabriolet is deleted" << endl;
    }
};

int main()
{
    Vehicle* car = new Cabriolet();
    car->StartEngine();

    delete car;
}
```

Chạy chương trình lên và ta ... không thấy hàm hủy nào được gọi! Lý do là gì?

```
Roof removed
Engine started!
```

Lí do là con trỏ car đang có kiểu **Vehicle***, mặc dù nó đang trỏ đến đối tượng kiểu **Cabriolet***. Tuy nhiên khi gọi hàm delete car thì hàm hủy của lớp Vehicle sẽ được gọi. Mà ta không cài đặt hàm hủy thì trình biên dịch sẽ gọi hàm hủy mặc định (vốn không làm gì cả - hàm rỗng).

Vậy chỉ cần sửa lại hàm hủy của lớp **Vehicle** thành hàm hủy ảo như sau:

```
class Vehicle {
public:
    virtual void StartEngine() = 0;
    virtual ~Vehicle() {}
};
```

Chạy lại mã nguồn sau khi thay đổi để thấy như sau:

```
Roof removed
Engine started!
Cabriolet is deleted
Car is deleted
```

4 Bài tập vận dụng

Yêu cầu

1. Thực hiện định nghĩa lớp theo thiết kế cho trước vào tập tin .h.
2. Thực hiện cài đặt lớp trong tập tin .cpp cho lớp tương ứng.
3. Viết các đoạn mã nguồn theo yêu cầu như bên dưới

Danh sách các lớp cần cải tiến cụ thể

1. Viết lại bài tập **nông trại** ở tuần trước sử dụng hàm thuần ảo.

Gợi ý: áp dụng hàm thuần ảo trong các hàm **Kêu**, **Cho sữa** và **Sinh con**.

+ Tất cả các con vật có thể có chung lớp cha là gì?

2. Viết bài tập vẽ hình: **đường thẳng**, **hình chữ nhật**, hình **ê líp** sử dụng hàm thuần ảo.

Tự phát sinh n hình thuộc về 3 loại hình trên và ngẫu nhiên giá trị các thuộc tính luôn.

Lần lượt lặp qua từng hình, vẽ hình tương ứng. Để đơn giản hàm Draw chỉ cần cout là đang vẽ hình gì.

+ Tất cả các hình trên có thể kế thừa từ lớp cha nào?

5 Hướng dẫn nộp bài

+ Mỗi bài tương ứng với 1 folder. Chỉ chứa file .h và .cpp và Makefile
+ Tổng hợp tất cả folder vào folder MSSV.
+ Nén lại tất cả thành một tập tin duy nhất MSSV.zip.

Để nộp bài, nén tất cả lại và đặt tên với định dạng MSSV.zip

-- HẾT --