Lab 11
Kế thừa

Lập trình hướng đối tượng

Mục tiêu	1. Biết tạo ra kế thừa cơ bản 2. Hiểu được thứ tự hàm tạo và hàm hủy trong kế thừa

1 Hướng dẫn khởi đầu 1 – Kế thừa

Mô tả bài tập

Cài đặt lớp **MixedFraction** để hỗ trợ hỗn số. Sau đó cài đặt hàm **ToString** để biểu diễn một hỗn số, ví dụ với phân số "12/9" sẽ hiển thị "1 1/3"

Hướng dẫn cài đặt

Bước 1: Sử dụng lớp phân số có sẵn như sau:

```
class Fraction {
protected:
    static string SEPERATOR;
    int _num;
    int _den;
public:
    int Numerator() { return _num; }
    int Denominator() { return _den; }
    void SetNumerator(int value) { _num = value; }
    void SetDenominator(int value) { _den = value; }
public:
    Fraction() {
        _num = 0;
        _den = 1;
    }

    Fraction(int num, int den) {
        _num = num;
        _den = den;
    }

    string ToString() {
        stringstream writer;
        writer << _num << SEPERATOR << _den;
        return writer.str();
    }
};
```

Bước 2: Cài đặt lớp MixedFraction hỗ trợ hỗn số như sau

```
class MixedFraction: public Fraction {
public:
    MixedFraction(int num, int den): Fraction(num, den) {
    }

    string ToString() {
        stringstream writer;

        int temp = _num;
        if (_num > _den) {
            writer << _num / _den << " ";
            temp = _num % _den;
        }

        writer << temp << SEPERATOR << _den;

        return writer.str();
    }
};
```

Bước 3: Cài đặt hàm main để kiểm thử cài đặt

```
string Fraction::SEPERATOR = "/";

int main()
{
    MixedFraction f(12, 9);
    cout << f.ToString();
}
```

2 Hướng dẫn khởi đầu 2 – Thứ tự hàm tạo

Bước 1: Hiệu chỉnh hàm tạo lớp Fraction

```
Fraction(int num, int den) {
    _num = num;
    _den = den;

    cout << "Fraction constructor";
}
```

Bước 2: Hiệu chỉnh hàm tạo lớp MixedFraction

```
MixedFraction(int num, int den): Fraction(num, den) {
    cout << "Mixed fraction constructor";
}
```

Bước 3: Chạy lại hàm main để thấy thứ tự gọi hàm tạo

```
string Fraction::SEPERATOR = "/";

int main()
{
    MixedFraction f(12, 9);
    cout << f.ToString();
}
```

Kết quả như sau:

```
Fraction constructor
Mixed fraction constructor
1 3/9
```

3 Hướng dẫn khởi đầu 3 – Thứ tự hàm hủy

Bước 1: Hiệu chỉnh bổ sung thêm hàm Hủy bên trong lớp Fraction

```
~Fraction() {
    cout << "Fraction destructor" << endl;
}
```

Bước 2: Hiệu chỉnh bổ sung thêm hàm Hủy bên trong lớp MixedFraction

```
~MixedFraction() {
    cout << "MixedFraction destructor" << endl;
}
```

Bước 3: Chạy lại hàm main để thấy thứ tự gọi hàm hủy

```
Fraction constructor
Mixed fraction constructor
1 3/9
MixedFraction destructor
Fraction destructor
```

Câu hỏi: Bạn có thể rút ra qui luật của hàm tạo trong kế thừa và hàm hủy trong kế thừa hay không?

Gợi ý: Giữa cha và con, ai được tạo trước? Ai được hủy trước?

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 kiểm tra việc định nghĩa lớp trong hàm main.

Danh sách các lớp cần cài đặt

1. Giả sử ta có lớp **Xe hơi** (Car) với các khả năng như sau:

+ **StartEngine**(): Khởi động động cơ

Để đơn giản hàm này sẽ in ra một dòng "*Starting engine*"

Cài đặt lớp Xe hơi mui trần (**Carbriolet**) khi khởi động động cơ thì sẽ bỏ đi phần mui.

Cài đặt bằng cách nạp đè hàm StartEngine và xuất ra dòng chữ "*Removing roof*" trước rồi xuất ra chữ "*Starting engine*" sau.

2. Giả sử có lớp **Thời gian** (**Time**) với 3 thành phần **giờ** (số nguyên), **phút** (số nguyên), **giây** (số nguyên).

+ Hàm tạo không đối sẽ tạo ra đối tượng với thời gian hiện tại.

+ Hàm tạo có đối sẽ khởi tạo với giờ phút giây truyền vào

Cài đặt lớp Thời gian theo GMT (**GmtTime**) kế thừa lớp **Time** có các đặc tính sau:

+ Thêm vào hàm **ToString**(int gmt): đối số gmt sẽ cho biết độ lệch thời gian theo giờ.

Ví dụ ta có đối tượng thời gian **GmtTime** now() ứng với 10h 15m 0s.

Nếu gọi now.ToString(0) sẽ in ra với độ lệch là 0, tức 10:15:00

Nếu gọi now.ToString(7) sẽ in ra với độ lệch là +7, tức 17:15:00

Chú ý khi vượt quá 24h! Giờ chỉ có giá trị từ 0h đến 23h59.

3. Một nông trại chăn nuôi có 3 loại gia súc: **bò**, **cừu**, và **dê**. Mỗi loại gia súc đều có thể **sinh con**, **cho sữa** và **phát ra tiếng kêu riêng** của chúng. Khi đói, các gia súc sẽ phát ra tiếng kêu để đòi ăn. Sau một thời gian chăn nuôi, người chủ nông trại muốn thống kê xem trong nông trại có bao nhiêu gia súc ở mỗi loại, tổng số lit sữa mà tất cả các gia súc của ông đã cho.

Áp dụng kế thừa, xây dựng chương trình cho phép người chủ nông trại tạo ra số lượng gia súc ngẫu nhiên lúc ban đầu.

a. Một hôm người chủ nông trại đi vắng, tất cả gia súc trong nông trại đều đói. Hãy cho biết những tiếng kêu nghe được trong nông trại.

Gợi ý: Duyệt qua mảng các gia súc, lần lượt gọi hàm Kêu của từng con vật.

Nếu là bò sẽ kêu kiểu bò, cừu kêu kiểu cừu, dê kêu kiểu dê. (Đừng quá quan trọng các con này kêu cụ thể ra sao nhé 😊, tượng trưng để hiểu bài là được)

b. Chương trình sẽ đưa ra thống kê các thông tin người chủ mong muốn (nêu trên) sau một lứa sinh và một lượt cho sữa của tất cả gia súc. Biết rằng:

- Tất cả gia súc ở mỗi loại đều sinh con (chỉ có giống cái)
- Số lượng sinh của mỗi gia súc là ngẫu nhiên.
- Tất cả gia súc ở mỗi loại đều cho sữa.
- Số lit sữa mỗi gia súc cho là ngẫu nhiên nhưng trong giới hạn sau:
    + Bò: 0 – 20 lit.
    + Cừu: 0 – 5 lit.
    + Dê: 0 – 10 lit.

5 Bài tập vận dụng khó siêu cấp vô địch

1. Cho hai lớp Hình vuông (**Square**) và Hình chữ nhật (**Rectangle**). Ta biết hình vuông là một trường hợp đặc biệt của hình chữ nhật. Lớp nào kế thừa lớp nào?

Thử cài đặt hai lớp này.

2. Cho hai lớp Hình tròn (**Circle**) và Hình Ê líp (**Ellipse**). Ta biết hình tròn là một trường hợp đặc biệt của hình ê líp. Lớp nào kế thừa lớp nào?

Thử cài đặt hai lớp này.

Chú ý: Kế thừa cơ học (chỉ vì muốn tái sử dụng thành phần, mã nguồn) mà không quan tâm ngữ nghĩa thì không phải là kế thừa đúng.

6 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 --