

# Lab 7

## 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: Tạo mới dự án

- Chọn loại dự án là **C++ / Console Application**.
- Đặt tên solution là: **Inheritance**. Đặt tên project là **MixedFraction**
- Nếu sử dụng Visual Studio 2017 trở lên cần **vô hiệu hóa Precompiled header** bằng cách nhấn phải vào project chọn Properties. Vào mục **C / C++ > All Options**, tìm tới tùy chọn **Precompiled header** và chọn **Not using precompiled headers**.

**Bước 2: 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 3: 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 4: 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

- Thực hiện định nghĩa lớp theo thiết kế cho trước vào tập tin .h.
- Thực hiện cài đặt lớp trong tập tin .cpp cho lớp tương ứng.
- Viết các đoạn mã nguồn kiểm tra việc định nghĩa lớp trong hàm main.

### Lớp cần cài đặt

- 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 đó, 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 lít 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.
  - 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.
  - 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 lít 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.

-- HẾT --