# **Bài 10. Class, operator overloading, friend, template**

**Mục tiêu:** *Luyện tập cú pháp class, định nghĩa toán tử, con trỏ this.*

*Bài này kế thừa sản phẩm bài trước.*

**Yêu cầu nộp bài** *: Toàn bộ code phần A. Bài (\*) sẽ bù điểm cho các bài tập trước.*

***Lưu ý: chép bài hoặc cho chép bài sẽ dẫn đến trượt môn học!***

## Thực hành

1. ***Con trỏ this.*** Với class Fraction đã viết từ bài trước (bổ sung hoàn thiện nếu chưa có) hãy sửa để các phép toán + thay vì tạo kết quả là một Fraction mới thì ghi kết quả vào chính Fraction hiện hành, sau đó dùng con trỏ **this** để trả về chính đối tượng hiện hành.   
   Gợi ý: bạn sẽ phải gỡ từ khóa const ra khỏi các biến thành viên tử số và mẫu số.  
   Thử nghiệm các toán tử trong hàm testA1.  
   Bây giờ hãy thử chạy đoạn sau xem kết quả có đúng là ½ + 1 + ½ == 2:  
    Fraction f1(1,2), f2(1,1);

Fraction sum = f1 + f2 + f1;

sum.print();

Hãy thử giải thích xem tại sao tổng trên lại không ra kết quả bằng 2?

Gợi ý (bôi đen để xem): chính vì phép cộng thứ nhất làm thay đổi giá trị của f1 từ ½ thành 3/2, dẫn đến biểu thức thực ra tương đương với ½ + 1 + 3/2.  
Đó chính là lí do vì sao Fraction và nhiều kiểu dữ liệu mà ta hay dùng trong các phép toán nên được cài đặt ở dạng immutable (không thể thay đổi giá trị) - cách mà bạn đã cài đặt trong bài 13.

1. ***Hàm friend***. Với cài đặt Fraction đã có, hãy viết một hàm get\_max() lấy hai tham số là hai đối tượng Fraction và trả về đối tượng có giá trị lớn hơn. get\_max không phải hàm thành viên của Fraction. Thay vào đó, hãy cho get\_max là friend của lớp Fraction.   
   Thử nghiệm get\_max trong hàm testA2
2. ***Overload phép so sánh***. Hãy định nghĩa lại các toán tử > và < cho class Fraction.   
   Thử nghiệm trong hàm testA3
3. ***Template***. Với các toán tử > và < của Fraction đã được định nghĩa, hãy sửa nội dung hàm get\_max để nó sử dụng > và < của Fraction thay vì trực tiếp truy nhập các biến thành viên của các đối tượng Fraction. Bạn có thấy rằng sau sửa đổi đó nội dung của get\_max cho Fraction giống hệt nội dung bạn có thể viết cho get\_max của các loại dữ liệu khác? Hãy sửa để biến get\_max thành hàm template.  
   Thử nghiệm get\_max cho các loại dữ liệu khác nhau trong hàm testA4.  
   Bây giờ liệu có cần cho get\_max là friend của Fraction nữa không?
4. ***Thư viện <algorithm>***. Toán tử > và < mà bạn đã định nghĩa thực ra đã đủ để có thể sử dụng một hàm max() có sẵn trong thư viện algorithm, bạn không phải tự viết get\_max nữa. Hãy thử #include <algorithm>, và dùng hàm max() trong đó, ví dụ:

Fraction f(2,3), f2(1,2);

Fraction m2 = max(f,f2);

m2.print();

Bạn sẽ thấy hàm max đó chạy như thể nó đã được viết sẵn cho Fraction. Thực ra đó là một hàm dạng template với nội dung tương tự như get\_max mà bạn đã làm. Nhưng bạn chỉ dùng được hàm max đó cho Fraction sau khi bạn đã định nghĩa các phép toán > và < của Fraction một cách phù hợp (không tin hãy thử comment out cài đặt của > và < và dịch lại, hoặc sửa để các cài đặt đó trả về kiểu dữ liệu linh tinh gì đó khác, hoặc tham số hàm không phải Fraction...).

Trong thư viện algorithm còn nhiều hàm cài đặt sẵn các công việc thường gặp, chẳng hạn như sắp xếp - sort, đổi giá trị - swap…. (tra cứu chi tiết tại <http://www.cplusplus.com/reference/algorithm/> ) Hãy tự học tiếp để sử dụng.

1. ***(\*) Overload <<.*** Bạn chắc đã thấy là dùng hàm print khá bất tiện, nếu có thể viết cout << f; với f là một đối tượng Fraction thì sẽ tiện hơn nhiều. Bạn hãy định nghĩa phép toán << cho Fraction. Tự tìm hiểu cách làm tại đây: http://www.learncpp.com/cpp-tutorial/93-overloading-the-io-operators/

1. ***Overload phép gán.*** Với cài đặt danh sách liên kết đôi SLinkedList bạn đã làm ở bài trước (bổ sung để hoàn thiện nếu bài nộp trước chưa hoàn thiện, hoặc lấy từ bài mẫu), phép gán (=) hiện đang là phép gán nông. Nói cách khác, sau khi thực hiện phép gán mặc định thì danh sách đích có chung các nút với danh sách nguồn. Việc này dẫn tới lỗi bộ nhớ khi trả về một danh sách được khai báo là biến địa phương (các nút trong danh sách nguồn bị hủy khi danh sách đó bị hủy ở thời điểm hàm kết thúc).
   1. Hãy thử nghiệm tình huống lỗi đó (viết một hàm tạo một danh sách là biến địa phương sau đó trả về danh sách đó, thử in danh sách nhận được từ lời gọi hàm nói trên)
   2. Hãy định nghĩa lại toán tử phép gán = để thực hiện copy sâu: tạo bản sao của toàn bộ chuỗi các nút, thay vì chỉ sao chép các con trỏ head và tail.

## B. Câu hỏi trắc nghiệm.

<Thầy Khôi sẽ bổ sung vào đây>