**Lập trình hướng đối tượng**

Copy constructor, assignment

Getter & setter

Nội dung của module gồm có:

1. Copy constructor
2. Copy assignment operator
3. Getter & setter
4. Các bài tập vận dụng

C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Bộ môn Công nghệ phần mềm

Khoa Công nghệ thông tin

Đại học Khoa học tự nhiên TP HCM

# Constructor nâng cao

## Copy constructor

Hàm tạo sao chép (copy constructor) dùng để khởi tạo một đối tượng từ một đối tượng khác có cùng kiểu.

**Các tính chất của hàm tạo sao chép:**

* Hàm tạo sao chép có đầy đủ tính chất của một hàm tạo thông thường.
* Thường được dùng để tạo bản sao của đối tượng.
* Trong C++, hàm tạo sao chép của lớp A được khai báo như sau:

class A

{

public:

A(const A &obj);

};

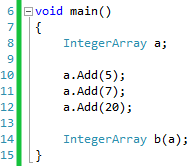
Bất kỳ lớp đối tượng nào cũng có hàm tạo sao chép. Trong trường hợp chúng ta không khai báo hàm tạo sao chép cho lớp đối tượng, hàm tạo sao chép mặc định *(default copy constructor)* sẽ được tự động được thêm vào.

**Tutorial 1:** Ví dụ minh họa nếu không có hàm tạo sao chép thì sẽ có vấn đề với chương trình.

Xét lớp mảng động IntegerArray có cài đặt như sau:

|  |  |
| --- | --- |
| **IntegerArray.h** | **IntegerArray.cpp** |
|  |  |

Khảo sát hàm main có code như sau:



Chạy thử chương trình ta sẽ nhận được thông báo lỗi lúc runtime. Nếu đặt breakpoint (F9) tại hàm hủy, ngay câu lệnh delete ta sẽ biết được nguyên nhân lỗi là tại đây.

Phân tích chương trình chúng ta thấy rằng lớp IntegerArray không khai báo hàm dựng sao chép, do đó hàm dựng sao chép mặc định sẽ được tự động thêm vào.

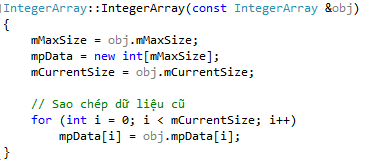
Hàm dựng sao chép mặc định tạo bản sao từ đối tượng bằng cách sao chép từng thuộc tính của đối tượng đó, nhưng lại không sao chép vùng nhớ cấp phát cho những thuộc tính kiểu con trỏ. Như vậy mpData của đối tượng b và mpData của đối tượng a sẽ **sử dụng chung một vùng nhớ**. Khi kết thúc hàm main(), đối tượng b bị hủy trước, hàm hủy của b được gọi để hủy vùng nhớ cấp phát cho m\_pElement của b. Kế đó đối tượng a bị hủy, hàm hủy của a cũng được gọi để hủy vùng nhớ cấp phát cho mpData của a. **Nhưng bấy giờ vùng nhớ này đã bị hủy trước đó**. Chính điều này gây ra lỗi chương trình.

Hơn nữa việc 2 mảng sử dụng chung một vùng nhớ để lưu các phần tử rõ ràng là sẽ gây ra những sai lệch về mặt ngữ nghĩa của chương trình.

Để giải quyết vấn đề này, chúng ta phải xây dựng hàm tạo sao chép cho lớp IntegerArray. Trong file IntegerArray.h, thêm vào khai báo hàm dựng sao chép cho lớp IntegerArray như sau:

IntegerArray(const IntegerArray &obj);

Trong file IntegerArray.cpp viết cài đặt cho hàm dựng sao chép



Chạy lại đoạn chương trình với hàm main như cũ, ta thấy lần này sau khi bổ sung hàm tạo sao chép thì không còn lỗi nữa.

***Như vậy, khi lớp đối tượng có thuộc tính kiểu con trỏ và cấp phát vùng nhớ thì phải xây dựng hàm dựng sao chép cho lớp đối tượng đó để sao chép vùng nhớ đã cấp phát.***

**Câu hỏi**:

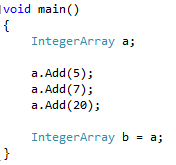
1. Tại sao khi khai báo hàm dựng sao chép kiểu của dữ liệu lại là const A& ? Có thể bỏ const đi được không? Có thể bỏ & đi được không?

2. Phân biệt đối số của hàm theo dạng tham trị, tham biến, tham chiếu.

## Copy assignment operator

**Tutorial 2**: Làm quen với lỗi nếu không có toán tử gán sao chép.

Xây dựng hàm main như sau:



Biên dịch và chạy thử chương trình chúng ta nhận được thông báo lỗi ngay tại hàm hủy của lớp IntegerArray. **Chuyện gì đã xảy ra???**

Phân tích toán tử gán chúng ta thấy rằng khi gán đối tượng x bằng đối tượng y (x, y cùng kiểu) thì tất cả các thuộc tính của x sẽ được gán giá trị bằng các thuộc tính tương ứng của y. Điều này có nghĩa là trong ví dụ trên mpData của đối tượng b sẽ được gán giá trị bằng mpData của đối tượng a. Như vậy mpData của b và mpData của a sẽ **cùng trỏ đến một vùng nhớ**. Đây chính là vấn đề chúng ta đã gặp phải khi đề cập đến hàm tạo sao chép ở trên.

Để giải quyết vấn đề này, chúng ta phải định nghĩa lại toán tử gán cho lớp IntegerArray. Trong file IntegerArray.h, thêm vào khai báo toán tử gán cho lớp IntegerArray như sau:

IntegerArray& operator =(const IntegerArray &obj);

Trong file IntegerArray.cpp viết cài đặt cho hàm dựng sao chép. Bạn có thể tự cài đặt! Nhớ return \*this ở cuối :D. Lí do là để có thể viết a = b = c = d = e; Đó là điểm khác biệt!

Trong C++ có luật Rule of three, tức là nếu hễ cài đặt 1 trong 3 hàm destructor, copy constructor, copy assignment operator thì phải cài đặt luôn hai cái còn lại. Đọc thêm ở phần mở rộng.

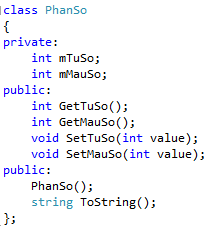
# Getter & setter

## 2.1. Sơ lược về getter & setter

Trong hướng đối tượng, để tránh việc can thiệp trực tiếp vào thuộc tính của một đối tượng (tính đóng gói) hoặc để cung cấp phương thức tương tác với các thuộc tính của đối tượng, chúng ta cần cung cấp các phương thức getter và setter cho mỗi thuộc tính của đối tượng. Getter là phương thức trả ra giá trị của một thuộc tính của đối tượng. Setter là phương thức khởi tạo giá trị cho một thuộc tính của đối tượng.

**Tutorial 3**: Làm quen với getter & setter

Xét lớp PhanSo có thiết kế như sau:



**Câu hỏi thảo luận liên quan đến Coding convention.**

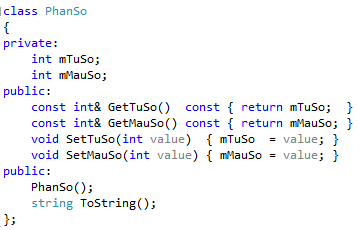
1. Có trường phái cho rằng nên đặt tên là LayTuSo thay cho GetTuSo, DatTuSo thay cho SetTuSo. Bạn chọn cách nào? Tại sao?

2. Có trường phái dùng toàn bộ tên các biến đối số của các hàm set tên là value như thiết kế ở trên, có trường phái cho rằng nên đặt tên cụ thể, dễ nhớ. Bạn chọn cách nào, tại sao?

3. Nên bố trí các thành phần private, protected, public của lớp như thế nào? Cái nào trước, cái nào sau? Có trường phái cho rằng nên đặt các thuộc tính và hàm public lên trên cùng vì đó là cái được đọc và quan tâm đầu tiên. Có trường phái viết như sau: trên cùng là hằng số, tiếp theo đến các thuộc tính private, tới các hàm get, set, rồi tới hàm tạo và hàm hủy, rồi mới tới các phương thức. Cho biết thứ tự sắp xếp các thành phần trong lớp của bạn và lí do tại sao.

## 2.2. Mở rộng khi viết getter và setter

**Tutorial 4**: Nâng cao về getter



**Câu hỏi thảo luận**:

1. Có trường phái cho rằng đối với các hàm get và set có thể cài đặt thẳng ở trong khai báo lớp luôn chứ không cần phải làm như thường lệ đối với các hàm, tức khai báo trong file .h rồi cài đặt ở file .cpp. Cho biết lựa chọn của bạn và tại sao?

2. Có trường phái cho rằng khi thiết kế lớp, đối với các hàm get, kiểu trả về cần phải có const A &. Theo bạn tại sao lại phải viết hàm get theo kiểu này?

Thông tin thêm là từ khóa const ở cuối hàm (thường chỉ dùng cho hàm thành viên của lớp) sẽ biến hàm đó thành hàm hằng, tức không thể thao tác thay đổi với thuộc tính của lớp.

**Để dễ hiểu**, hàm const int& GetTuSo() const thật ra sẽ có thêm đối số đầy đủ là:

const int& GetTuSo(const PhanSo\* this);

# Bài tập vận dụng

1. Thao tác với lớp Phân số. Thời gian dự kiến: 4 tiếng.

Cho lớp PhanSo có thiết kế như sau:

a. Cài đặt các phương thức của lớp PhanSo theo thiết kế trên.

b. Đọc danh sách các phân số từ tập tin PhanSoList.txt vào một vector và xuất ra màn hình danh sách phân số vừa đọc được. Cấu trúc tập tin PhanSoList.txt có dạng:

|  |  |
| --- | --- |
| **5**  3/2  10/50  6/19  51/3  9/21 | Số lượng phân số  Phân số thứ nhất  Phân số thứ hai  … |

c. Xuất ra màn hình các thông tin sau:

+ Phân số lớn nhất

+ Phân số nhỏ nhất

+ Tổng các phân số

+ Tích các phân số

d. Tạo ra một danh sách mới gồm các phân số chưa tối giản. Tối giản các phân số này và xuất ra màn hình danh sách nhỏ này.

2. Cài đặt lớp biểu diễn danh sách các sinh viên (các thuộc tính gồm có: MSSV, họ tên, ngày tháng năm sinh, số CMND, số điện thoại, email).

Đọc danh sách sinh viên từ tập tin.

Sắp xếp tăng dần theo họ tên, xuất ra màn hình.

3. Cài đặt lớp biểu diễn một lớp học.

Đọc danh sách lớp học từ tập tin.

Sắp xếp lớp học theo sĩ số.

Xuất ra tập tin khác.

# Các vấn đề mở rộng và đọc thêm

**Đọc thêm**

**1.** Rule of three: Một khi đã lỡ viết 1 trong 3 hàm destructor, copy constructor, copy assignment constructor thì phải viết luôn 2 hàm còn lại.

<http://en.wikipedia.org/wiki/Rule_of_three_(C%2B%2B_programming)>