Operator Overloading

CSC10003 – Phương pháp lập trình hướng đối tượng

Nội dung thực hành lần này sẽ tập trung phân tích và hướng dẫn một số vấn đề về việc định nghĩa và cài đặt lại các toán tử trong lớp đối tượng

1. Giới thiệu toán tử và nạp chồng toán tử

- Trong lập trình, toán tử là những ký hiệu thông báo cho trình biên dịch biết cần phải thực hiện một số thao tác được định nghĩa trước, tùy theo kiểu dữ liệu.
- Một số loại toán tử được định nghĩa sẵn và thường hay sử dụng trong C++
  - Toán tử số học: + - * / % giữa các kiểu dữ liệu dạng số (int, float, …)
  - Toán tử quan hệ: thường dùng trong việc so sánh, kiểm tra điều kiện
    - == > < >= <= !=
  - Toán tử gán:
    - Toán tử gán bằng: =
    - Toán tử += -= *= /=
  - Toán tử tiền tố, hậu tố:
    - Toán tử tiền tố: ++ --
      - Ví dụ: ++a, —b
    - Toán tử hậu tố: ++ --
      - Ví dụ: a++, b—
  - Toán tử logic: && || !
  - Toán tử trên bit: & | ^ ~
- C++ cho phép thực hiện nạp chồng toán tử (Operator overloading), định nghĩa lại cách xử lý tính toán riêng cho lớp đối tượng. Toán tử là một loại phương thức đặc biệt của lớp đối tượng nhằm thể hiện trực quan hơn khi được gọi thực hiện.
  - Đoạn chương trình sau ví dụ cho lớp Phân Số, trong C++ không định nghĩa sẵn các toán tử dành cho lớp Phân Số, do đó cần cài đặt lại nạp chồng toán tử cho lớp này.
  - Các toán tử cần cài đặt nạp chồng lại để chạy được chương trình sau:
    - + >= ++ <<

```
PhanSo a(1, 2);//Constructor 2 tham số
PhanSo b(3, 4);//Constructor 2 tham số
PhanSo c = a + b; // 1/2 + 3/4 = 10/8, thay vì gọi c = a.Cong(b);
if (a >= b) c++; //1/2 >= 3/4: sai: c giữ nguyên
cout << c;//Toán tử xuất được cài lại cho kiểu PhanSo
```

```
Tu so la: 10
Mau so la: 8
Press any key to continue . . .
```

- Cần cài đặt toán tử theo đúng ý nghĩa của nó
  - Toán tử cộng + phải thực hiện việc cộng trong đó.
- Các cặp toán tử có cùng chức năng phải được cài đặt cùng nhau và có chức năng tương tự
  - x = x + y và x += y phải được cài đặt giống chức năng
- Toán tử 2 ngôi: toán tử có 2 thành phần bên trái và bên phải toán tử, rất hay được sử dụng
  - x + y, x – y, x * y, …
  - x >= y, x == y, …
  - cin >> x, cout << y
- Toán tử 1 ngôi: toán tử chỉ có 1 thành phần, có thể đứng ở bên trái hay bên phải
  - a++
  - ++a
- Việc cài đặt dựa trên việc xem xét toán tử 2 ngôi hay 1 ngôi, từ đó sẽ biết tham số đầu vào và kiểu trả về phù hợp

2. *Toán tử gán bằng =*

- Toán tử gán bằng mang ý nghĩa cho việc gán giá trị của một biến này cho biến khác
- Toán tử gán bằng là toán tử 2 ngôi, vậy khi cài đặt toán tử này cần quan tâm đến:
  - Tham số đầu vào:
    - Thường là 1 biến có chung kiểu lớp đối tượng
    - Không được phép thay đổi giá trị, truyền tham chiếu
  - Kết quả trả về:
    - Không tạo ra một lớp đối tượng khác
    - Trả về chính nó (*this) sau khi đã thay đổi nội dung bên trong
  - Thực hiện thao tác: sao chép giá trị của một biến khác có cùng kiểu đối tượng
- Lưu ý: cần phải kiểm tra việc gán lại cho chính nó, lúc này không cần làm gì cả, đoạn mã nguồn sau đây ví dụ trong trường hợp lớp Phân Số không có thuộc tính kiểu con trỏ.

```
PhanSo& operator=(const PhanSo &ps)
	{
	//Kiểm tra tránh trường hợp gán bằng chính nó a = a
	if (this == &ps)
		return *this; //Trả về chính nó, không làm gì cả
	//Nếu gán cho một biến có nội dung khác, gán lại nội dung đó
	this->tuso = ps.tuso;
	this->mauso = ps.mauso;
	//Trả về chính nó sau khi gán lại
	return *this;
}
```

- Toán tử gán bằng luôn được hệ thống tạo sẵn mỗi khi tạo một lớp mới, được gọi là toán tử gán bằng mặc định.
  - Việc sao chép mặc định đó sẽ sao chép theo kiểu shallow copy!!
  - Nếu trong phần thuộc tính có thuộc tính kiểu con trỏ, vậy việc shallow copy chỉ sao chép địa chỉ của con trỏ, chứ không sao chép nội dung, vậy việc sao chép mặc định của hệ thống dẫn đến việc cả 2 con trỏ đều trỏ cùng vào 1 vùng nhớ!!!
  - Nếu có thuộc tính con trỏ, **BẮT BUỘC** phải cài đặt lại:
    - Toán tử gán bằng
    - Phương thức tạo lập sao chép
    - Phương thức hủy

```
class PhanSo
{
	int *tuso;
	int *mauso;

public:
	PhanSo(const PhanSo& ps) //Sao chép
	{
		tuso = new int; //Khởi tạo vùng nhớ
		mauso = new int;
		*tuso = *ps.tuso; //Gán giá trị vùng nhớ
		*mauso = *ps.mauso;
	}
	~PhanSo() //PHương thức hủy
	{
		delete tuso; //Xóa vùng nhớ động
		delete mauso;
	}
	PhanSo& operator=(const PhanSo &ps) //Toán tử gán bằng
	{
		if (this == &ps) //Tránh a = a
			return *this;
		delete tuso; //Xóa vùng nhớ cũ
		delete mauso;
		tuso = new int;// Tạo lại vùng nhớ mới
		mauso = new int;
		*this->tuso = *ps.tuso;//Gán giá trị cho vùng nhớ mới
		*this->mauso = *ps.mauso;
		return *this;
	}
	//... Các phương thức khác
};
```

3. Toán tử số học

- Toán tử số học gồm các toán tử thường mang ý nghĩa cho việc xử lý tính toán số học:
  - Toán tử + - * / %
- Toán tử số học có thể đi kèm chung với toán tử gán bằng, bao gồm 2 thao tác xử lý: tính toán số học trước, sau đó gán giá trị sau.
  - Toán tử += -= *= /= %=
- Toán tử số học là toán tử 2 ngôi:
  - x + y
  - x += y
  - x += 1
  - …
- Lưu ý khi cài đặt nạp chồng toán tử số học:
  - Nếu chỉ 1 toán tử, không có việc gán bằng:
    - Đầu vào thường là một biến có cùng kiểu đối tượng, không được thay đổi giá trị của biến này
    - Kết quả trả về là một biến tạm chứa kết quả cộng của 2 biến
  - Nếu toán tử số học có cả việc gán bằng
    - Đầu vào thường là một biến có cùng kiểu đối tượng, không được thay đổi giá trị của biến này
    - Kết quả trả về là chính bản thân đối tượng được xử lý (biến bên trái)

```
PhanSo operator+ (const PhanSo &ps)
{
	PhanSo ketqua; //Lưu kết quả vào biến tạm
	ketqua.tuso = this->tuso * ps.mauso + this->mauso * ps.tuso;
	ketqua.mauso = this->mauso * ps.mauso;
	return ketqua;
}

PhanSo& operator+=(const PhanSo &ps)
{
	this->tuso = this->tuso * ps.mauso + this->mauso * ps.tuso;
	this->mauso = this->mauso * ps.mauso;
	return *this; //Xử lý toán học và trả về chính nó
}
```

- Ví dụ cài đặt sau đây dành cho lớp Phân Số không có thuộc tính kiểu con trỏ, các cài dặt toán tử này phải ở bên trong lớp đối tượng
- Các toán tử số học khác cũng cài đặt tương tự, chỉ khác nhau ở phần xử lý tính toán số học

4. Toán tử quan hệ

- Toán tử quan hệ gồm các toán tử thường mang ý nghĩa cho việc thể hiện quan hệ nào đó, dùng để kiểm tra điều kiện, so sánh kết quả
  - Toán tử == > < >= <= !=
- Toán tử quan hệ là toán tử 2 ngôi, giá trị trả về thường là đúng hoặc sai
  - x == y
  - x > y
  - x <= y
  - …
- Khi cài đặt toán tử quan hệ:
  - Đầu vào thường là một biến có cùng kiểu đối tượng, không được thay đổi giá trị của biến này
  - Kết quả trả về là một biến bool
- Ví dụ sau đây cài đặt toán tử == trên lớp Phân Số không có thuộc tính kiểu con trỏ

```
bool operator==(const PhanSo &ps)
{
	int result = this->tuso * ps.mauso - this->mauso * ps.tuso;
	if (result == 0)
		return true;
	else
		return false;
}
```

5. Toán tử tiền tố, hậu tố

- Toán tử tiền tố, hậu tố là các toán tử mang ý nghĩa tính toán số học, nhưng được viết và hiểu theo một cách ngắn gọn hơn
  - Toán tử tiền tố: ++a, --a
  - Toán tử hậu tố: a++, a--
  - Chỉ áp dụng cho toán tử + và –
- Toán tử tiền tố, hậu tố là toán tử 1 ngôi:
  - Thường tăng và giảm 1 giá trị đơn vị tùy theo lớp đối tượng
  - Thường đi kèm với thao tác gán bằng, chỉ khác nhau thứ tự:
    - Tiền tố b = ++a: tăng giá trị a trước, sau đó mới gán cho b
    - Hậu tố b = a++: gán giá trị a cho b trước, sau đó mới tăng giá trị a
- Lưu ý khi cài đặt toán tử tiền tố, hậu tố:
  - Do cả 2 toán tử đều là 1 ngôi, vậy để phân biệt, đầu vào của toán tử hậu tố được thêm vào 1 biến giả để phân biệt.
  - Kết quả trả về của toán tử tiền tố là chính nó (tăng/giảm xong rồi mới gán), của hậu tố là một biến tạm (gán trước, tăng giảm xong trả về biến gán)

```
//Toán tử hậu tố a++
PhanSo operator++(int x)
{
	//Gọi phương thức sao chép, chép giá trị trước
	PhanSo result(*this);
	//Tiến hành xử lý trực tiếp trên đối tượng hiện tại
	this->tuso = this->tuso + this->mauso;
	//Trả về đối tượng sao chép, không thực hiện xử lý
	return result;
}

//Toán tử tiền tố ++a
PhanSo& operator++()
{
	//Do việc xử lý xong mới gán, nên chỉ cần xử lý và trả về chính nó
	this->tuso = this->tuso + this->mauso;
	return *this;
}
```

6. *Toán tử nhập, xuất*

- Toán tử nhập, xuất thường phải kết hợp với một cách thức nhập xuất dữ liệu
  - Nhập, xuất trên console: liên quan đến istream (cin), ostream (cout)
  - Nhập, xuất trên file : liên quan đến fstream (ifstream, ofstream)
- Toán tử nhập, xuất là toán tử 2 ngôi, trong đó 1 ngôi là đối tượng hiện tại đang xét, 1 ngôi xác định cách thức nhập xuất dữ liệu
- Phải cài đặt bên ngoài lớp do việc xác định cách thức dữ liệu thuộc về hệ thống
  - Cần khai báo trước bên trong lớp đối tượng, sau đó cài đặt bên ngoài
  - Nếu trong lớp có thuộc tính có tầm vực private, vậy cài đặt bên ngoài lớp không thể truy xuất thuộc tính để nhập xuất (do private), ta sử dụng khái niệm hàm bạn với từ khóa **friend**
- Hàm bạn, lớp bạn (friend) :
  - Truy xuất trực tiếp thuộc tính private của lớp chứa hàm/lớp này
  - Có tính chất 1 chiều :
    - A là bạn của B thì A được truy xuất thuộc tính private của B, nhưng B không được truy xuất thuộc tính private của A
- Khi cài đặt :
  - Đầu vào 2 tham số: là istream hay ostream (với console) và đối tượng Phân Số
  - Đầu ra là istream hay ostream (có dấu &)

```
class PhanSo
{
private:
	int tuso;
	int mauso;
public:
//Khai báo trước hàm bạn, cài đặt bên ngoài lớp do nhập xuất từ cin và
cout của hệ thống
	friend istream& operator>> (istream& is, PhanSo &ps);
	friend ostream& operator<< (ostream& os, PhanSo &ps);
};

//Đầu vào là istream (cin) ở bên trái, bên phải là PhanSo
istream& operator>> (istream& is, PhanSo &ps)
{
	cout << "Nhap tu so: ";
	is >> ps.tuso; //Thay cin bằng is
	cout << "Nhap mau so: ";
	is >> ps.mauso;
	return is;
}

//Đầu vào là ostream (cout) ở bên trái, bên phải là PhanSo
ostream& operator<< (ostream& os, PhanSo &ps)
{
	//Thay cout bằng os
	os << "Tu so la: " << ps.tuso << endl;
	os << "Mau so la: " << ps.mauso << endl;
	return os;
}

void main()
{
	PhanSo a;
	cin >> a;
	cout << a;
}
```

7. Bài tập thực hành

Bài tập 1: Viết chương trình C++ cài đặt lại lớp Phân Số, trong đó thuộc tính của lớp Phân Số được định nghĩa như sau:

```
class PhanSo
{
	int *tuso;
	int *mauso;
};
```

- Cài đặt nạp chồng các toán tử cho lớp Phân Số, các toán tử 2 ngôi với cả 2 ngôi đều có chung kiểu đối tượng Phân Số (lưu ý cần cài đặt tất cả các phương thức tạo lập, hủy có liên quan để chương trình không bị lỗi):
  - Toán tử gán bằng =
  - \+ - * /
  - += -= *= /=
  - == < > <= >= !=
  - ++ -- (cả tiền tố lẫn hậu tố)
  - >> << (nhập xuất ra console, xuất theo định dạng tử số / mẫu số)

Bài tập 2: Viết chương trình C++ cài đặt lớp Mảng số nguyên với ý nghĩa cài đặt một số toán tử hỗ trợ cho việc xử lý trên mảng động, trong đó các thuộc tính của lớp mảng số nguyên được định nghĩa như sau:

```
class MangSoNguyen
{
	int* dulieu; //Con trỏ trỏ đến 1 mảng động kiểu số nguyên
	int kichthuoc; //Biến chứa kích thước của mảng số nguyên
};
```

- Cài đặt nạp chồng toán tử cho lớp Mảng số nguyên này, lưu ý cần cài đặt tất cả các phương thức tạo lập, hủy có liên quan để chương trình không bị lỗi
  - Toán tử = cho phép gán lại từng giá trị số nguyên của một mảng số nguyên khác
  - Toán tử +: cho phép cộng 2 mảng số nguyên theo cơ chế: cộng từng giá trị phần tử tại từng vị trí, nếu 2 mảng có kích thước khác nhau, chỉ cộng những phần tử nào có giá trị tương ứng, giá trị khác giữ nguyên, ví dụ:
    - Mảng a:

1	2	3

    - Mảng b:

4	5	6

    - Mảng a + b:

5	7	9

  - Toán tử tiền tố, hậu tố ++ cho phép tăng từng phần tử của mảng số nguyên lên 1
  - Toán tử nhập, xuất << >> cho phép nhập và xuất từng phần tử của mảng số nguyên