Nội dung tuần 04

Luyện tập cách xây dựng các hàm toán tử của lớp đối tượng.

Các đặc điểm

Bản chất toán tử cũng là hàm

```
class SoPhuc
{
private:
	int thuc, ao;

public:
	SoPhuc();
	SoPhuc(int t, int a);
	SoPhuc operator+(const int&);

	friend ostream& operator<<(ostream&, const SoPhuc&);
};

SoPhuc::SoPhuc()
{
	thuc = ao = 0;
}

SoPhuc::SoPhuc(int t, int a)
{
	thuc = t;
	ao = a;
}

SoPhuc SoPhuc::operator+(const int& n)
{
	SoPhuc rt = *this;
	rt.thuc += n;
	return rt;
}
```

```
ostream& operator<<(ostream& os, const SoPhuc& sp)
{
	os << sp.thuc << " + " << sp.ao << "*i";
	return os;
}

void main()
{
	SoPhuc sp1(3, 5);
	SoPhuc sp2 = sp1.operator+(10);
	SoPhuc sp3 = sp1 + 10;
	SoPhuc sp4 = 10 + sp1;
	cout << sp2 << endl << sp3 << endl << sp4 << endl;
}
```

Chú ý dòng được tô vàng sẽ bị báo lỗi (giải thích???).

Ý nghĩa cài đặt hàm toán tử trong hay ngoài lớp đối tượng

```
class SoPhuc
{
private:
	int thuc, ao;

public:
	SoPhuc();
	SoPhuc(int t, int a);
	SoPhuc operator+(const int&);
	friend SoPhuc operator+(const int&, const SoPhuc&);

	friend ostream& operator<<(ostream&, const SoPhuc&);
};

SoPhuc::SoPhuc()
{
	thuc = ao = 0;
}

SoPhuc::SoPhuc(int t, int a)
{
```

```
	thuc = t;
	ao = a;
}

SoPhuc SoPhuc::operator+(const int& n)
{
	SoPhuc rt = *this;
	rt.thuc += n;
	return rt;
}

SoPhuc operator+(const int& n, const SoPhuc& sp)
{
	SoPhuc rt;
	rt.thuc = n + sp.thuc;
	rt.ao = sp.ao;
	return rt;
}

ostream& operator<<(ostream& os, const SoPhuc& sp)
{
	os << sp.thuc << " + " << sp.ao << "*i";
	return os;
}

void main()
{
	SoPhuc sp1(3, 5);
	SoPhuc sp2 = sp1.operator+(10);
	SoPhuc sp3 = sp1 + 10;
	SoPhuc sp4 = 10 + sp1;
	cout << sp2 << endl << sp3 << endl << sp4 << endl;
}
```

Xét ví dụ

```
class SoPhuc
{
private:
	int thuc, ao;
```

```
public:
	SoPhuc();
	SoPhuc(int t, int a);
	SoPhuc(const int&);
	SoPhuc operator+(const SoPhuc&);

	friend ostream& operator<<(ostream&, const SoPhuc&);
};

SoPhuc::SoPhuc()
{
	thuc = ao = 0;
}

SoPhuc::SoPhuc(int t, int a)
{
	thuc = t;
	ao = a;
}

SoPhuc::SoPhuc(const int& n)
{
	thuc = n;
	ao = 0;
}

SoPhuc SoPhuc::operator+(const SoPhuc& sp)
{
	SoPhuc rt = *this;
	rt.thuc += sp.thuc;
	rt.ao += sp.ao;
	return rt;
}

ostream& operator<<(ostream& os, const SoPhuc& sp)
{
	os << sp.thuc << " + " << sp.ao << "*i";
	return os;
}
```

```
void main()
{
	SoPhuc sp1(3, 5);
	SoPhuc sp2 = sp1.operator+(10);
	SoPhuc sp3 = sp1 + 10;
	cout << sp2 << endl << sp3 << endl;
}
```

Chú ý là không có cài đặt toán tử + **SoPhuc** với **int** nhưng hàm main vẫn thực hiện được.

Copy Constructor và Operator =

```
class HS
{
private:
	char *hoTen;

public:
	HS(void);
	HS(const char *ht);
	HS(const HS& hs);
	~HS(void);
	const HS& operator=(const HS& hs);
};

HS::HS(void)
{
	hoTen = NULL;
}

HS::HS(const char *ht)
{
	int len = strlen(ht);
	hoTen = new char[len + 1];
	strcpy_s(hoTen, len + 1, ht);
}

HS::HS(const HS& hs)
{
	cout << "Copy Constructor..." << endl;
```

```
	int len = strlen(hs.hoTen);
	hoTen = new char[len + 1];
	strcpy_s(hoTen, len + 1, hs.hoTen);
}

HS::~HS(void)
{
	if (hoTen != NULL)
	{
		delete[] hoTen;
	}
}

const HS& HS::operator=(const HS& hs)
{
	cout << "Operator = ..." << endl;
	if (hoTen != NULL)
	{
		delete[] hoTen;
	}
	int len = strlen(hs.hoTen);
	hoTen = new char[len + 1];
	strcpy_s(hoTen, len + 1, hs.hoTen);
	return *this;
}

void main()
{
	HS hs1("sdfdgfg");
	cout << "hs2(hs1)" << endl;
	HS hs2(hs1);
	cout << endl << "hs3 = hs1" << endl;
	HS hs3 = hs1;
	cout << endl << "hs2 = hs3" << endl;
	hs2 = hs3;
	cout << endl;
}
```

```
hs2(hs1)
Copy Constructor...

hs3 = hs1
Copy Constructor...

hs2 = hs3
Operator = ...

Press any key to continue . . .
```

Chú ý 2 dòng lệnh được highlight đều là phép gán nhưng chỉ 1 lần hàm Operator= được gọi. Như vậy hàm toán tử gán chỉ được gọi khi đối tượng đã được khởi tạo trước đó (hs2 đã được khởi tạo trước). ***Do vậy nguyên tắc khi có sử dụng cấp phát động thì phải thu hồi trong hàm toán tử gán***.

Ví dụ class SoNguyenLon

Phần khai báo

```
#define MAXLEN 100

class SoNguyenLon
{
private:
	int mangSo[MAXLEN];
	int soCS;
	static SoNguyenLon snlMax;

public:
	SoNguyenLon(void);
	SoNguyenLon(const int& cs, const int& scs);
	SoNguyenLon(const unsigned int& n);
	SoNguyenLon(const SoNguyenLon& snl);
	~SoNguyenLon(void);

	SoNguyenLon operator+(const SoNguyenLon& snl);
	SoNguyenLon operator-(const SoNguyenLon& snl);
	bool operator>(const SoNguyenLon& snl);
	const SoNguyenLon& operator=(const SoNguyenLon& snl);
	friend SoNguyenLon operator+(const unsigned int& n, const SoNguyenLon& snl);
	friend SoNguyenLon operator-(const unsigned int& n, const SoNguyenLon& snl);
	friend ostream& operator<<(ostream& os, const SoNguyenLon& snl);

	static SoNguyenLon SNLMax();
};
```

Phần cài đặt

```
SoNguyenLon::SoNguyenLon(void)
{
	soCS = 1;
	mangSo[soCS - 1] = 0;
	if (*this > snlMax)
	{
		snlMax = *this;
	}
}

SoNguyenLon::SoNguyenLon(const int& cs, const int& scs)
{
	int csR = cs;
	if (csR < 1)
	{
		csR = 1;
	}
	if (csR > 9)
	{
		csR = 9;
	}
	soCS = abs(scs);
	if (soCS < 1)
	{
		soCS = 1;
	}
	if (soCS > MAXLEN)
	{
		soCS = MAXLEN;
	}
	for (int i=0; i<soCS; ++i)
	{
		mangSo[i] = cs;
	}
	if (*this > snlMax)
	{
		snlMax = *this;
```

```
	}
}

SoNguyenLon::SoNguyenLon(const unsigned int& n)
{
	unsigned int temp = n;
	soCS = 0;
	while (temp > 9)
	{
		mangSo[soCS++] = temp % 10;
		temp /= 10;
	}
	mangSo[soCS++] = temp;
	if (*this > snlMax)
	{
		snlMax = *this;
	}
}

SoNguyenLon::SoNguyenLon(const SoNguyenLon& snl)
{
	soCS = snl.soCS;
	for (int i=0; i<soCS; ++i)
	{
		mangSo[i] = snl.mangSo[i];
	}
}

SoNguyenLon::~SoNguyenLon(void)
{
}

bool SoNguyenLon::operator>(const SoNguyenLon& snl)
{
	if (soCS > snl.soCS)
	{
		return true;
	}
	if (soCS < snl.soCS)
```

```
	{
		return false;
	}
	for (int i=soCS-1; i>=0; --i)
	{
		if (mangSo[i] == snl.mangSo[i])
		{
			continue;
		}
		if (mangSo[i] > snl.mangSo[i])
		{
			return true;
		}
		return false;
	}
	return false;
}

const SoNguyenLon& SoNguyenLon::operator=(const SoNguyenLon& snl)
{
	soCS = snl.soCS;
	for (int i=0; i<soCS; ++i)
	{
		mangSo[i] = snl.mangSo[i];
	}
	return *this;
}

SoNguyenLon SoNguyenLon::operator+(const SoNguyenLon& snl)
{
	SoNguyenLon snlKQ;
	const SoNguyenLon *snlSCSMax = (soCS > snl.soCS) ? this : &snl;
	const SoNguyenLon *snlSCSMin = (soCS < snl.soCS) ? this : &snl;
	int soCSMin = (soCS > snl.soCS) ? snl.soCS : soCS;
	int nho = 0;
	for (int i=0; i<snlSCSMin->soCS; ++i)
	{
		snlKQ.mangSo[i] = mangSo[i] + snl.mangSo[i] + nho;
		nho = snlKQ.mangSo[i] / 10;
```

```
                snlKQ.mangSo[i] %= 10;
        }
        for (int i=snlSCSMin->soCS; i<snlSCSMax->soCS; ++i)
        {
                snlKQ.mangSo[i] = snlSCSMax->mangSo[i] + nho;
                nho = snlKQ.mangSo[i] / 10;
                snlKQ.mangSo[i] %= 10;
        }
        snlKQ.soCS = snlSCSMax->soCS;
        if (nho > 0)
        {
                snlKQ.mangSo[snlKQ.soCS++] = 1;
        }
        if (snlKQ > snlMax)
        {
                snlMax = snlKQ;
        }
        return snlKQ;
}

SoNguyenLon SoNguyenLon::operator-(const SoNguyenLon& snl)
{
        SoNguyenLon snlKQ;
        int nho = 0, i;
        if (soCS >= snl.soCS)
        {
                for (i=0; i<snl.soCS; ++i)
                {
                        snlKQ.mangSo[i] = mangSo[i] - snl.mangSo[i] - nho;
                        nho = 0;
                        if (snlKQ.mangSo[i] < 0)
                        {
                                snlKQ.mangSo[i] += 10;
                                nho = 1;
                        }
                }
                for (; i<soCS; ++i)
                {
                        snlKQ.mangSo[i] = mangSo[i] - nho;
```

```
                        nho = 0;
                        if (snlKQ.mangSo[i] < 0)
                        {
                                snlKQ.mangSo[i] += 10;
                                nho = 1;
                        }
                }
                snlKQ.soCS = soCS;
                while(snlKQ.mangSo[snlKQ.soCS-1] == 0)
                {
                        snlKQ.soCS--;
                }
        }
        return snlKQ;
}

SoNguyenLon operator+(const unsigned int& n, const SoNguyenLon& snl)
{
        SoNguyenLon snlTemp(n);
        SoNguyenLon snlKQ = snlTemp + snl;
        if (snlKQ > SoNguyenLon::snlMax)
        {
                SoNguyenLon::snlMax = snlKQ;
        }
        return snlKQ;
}

SoNguyenLon operator-(const unsigned int& n, const SoNguyenLon& snl)
{
        SoNguyenLon snlTemp(n);
        return snlTemp - snl;
}

ostream& operator<<(ostream& os, const SoNguyenLon& snl)
{
        for (int i=snl.soCS-1; i>=0; --i)
        {
                os << snl.mangSo[i];
        }
```

```
        return os;
}

SoNguyenLon SoNguyenLon::SNLMax()
{
        return snlMax;
}
```

Bài tập

Bài 1

Khai báo và cài đặt lớp Ngày sao cho hàm main sau chạy đúng

```
void main()
{
        Ngay n1;                    //1/1/1
        Ngay n2(02,10,2014);        //2/10/2014
        Ngay n3(-10,16,2000);       //10/04/2001
        Ngay n4(1000);              //27/9/3
        Ngay n5 = n2 + n3;          //12/2/4016
        Ngay n6 = n1 + 5000;        //10/10/15
        Ngay n7 = 1234 + n4;        //14/2/7
        Ngay n8 = 190 + n6 + n7;    //2/7/23
        Ngay n9 = n8 - n6;          //1/9/7
        Ngay n10 = 12000 - n9;      //9/2/26
        if (n10 > n6)
        {
                n10 = n2 - 1000 + n6;
        }
        cout << n1 << endl << n2 << endl << n3 << endl << n4 << endl;
        cout << n5 << endl << n6 << endl << n7 << endl << n8 << endl;
        cout << n9 << endl << n10 << endl;
}
```

Bài 2

Khai báo và cài đặt lớp Thời Gian để chạy đúng với hàm main sau:

```
void main()
```

```
{
	ThoiGian tg1;				//00:00:00
	ThoiGian tg2(1212);			//00:20:12
	ThoiGian tg3(125,45);			//02:05:45
	ThoiGian tg4(12,239,-78);		//16:00:18
	ThoiGian tg5 = tg2 + tg3;		//02:25:57
	ThoiGian tg6 = 5000 + tg2;		//01:43:32
	ThoiGian tg7 = tg4 - tg6;		//14:16:46
	ThoiGian tg8 = 12300 - tg4;		//00:00:00
	ThoiGian tg9, tg10;
	if (tg8 <= tg3)
	{
		tg9 = tg1 + tg2 + 36000;	//10:20:12
	}
	if (12345 <= tg5)
	{
		tg10 = tg5 + 12345;		//05:51:42
	}
	cout << tg1 << endl << tg2 << endl << tg3 << endl << tg4 << endl;
	cout << tg5 << endl << tg6 << endl << tg7 << endl << tg8 << endl;
	cout << tg9 << endl << tg10 << endl;
}
```