Nạp chồng toán tử - Operator Overloading

Giới thiệu

- C/C++ đã làm sẵn các toán tử cho các kiểu cài sẵn (int, float…)
- Đối với các kiểu dữ liệu người dùng: C++ cho phép định nghĩa các toán tử trên các kiểu dữ liệu người dùng → overload
- Các toán tử cho phép ta sử dụng cú pháp toán học đối với các kiểu dữ liệu của C++ thay vì gọi hàm (tuy bản chất vẫn là gọi hàm).
  - Ví dụ thay **a.set(b.cong(c));** bằng **a = b + c;**
  - Gần với kiểu trình bày mà con người dùng quen
  - Đơn giản hóa mã chương trình

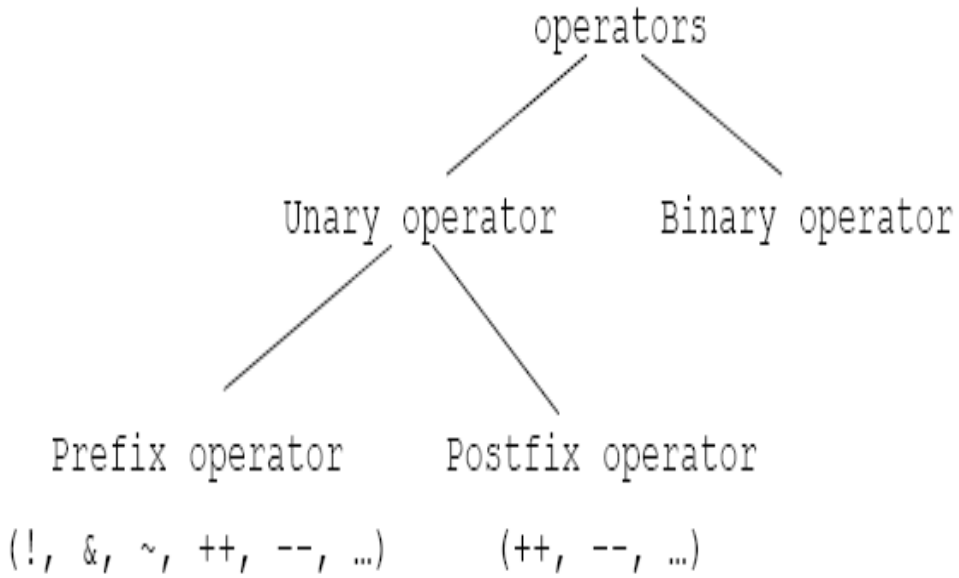
Operator Overload

- Một toán tử có thể dùng cho nhiều kiểu dữ liệu.
- Như vậy, ta có thể tạo các kiểu dữ liệu đóng gói hoàn chỉnh (fully encapsulated) để kết hợp với ngôn ngữ như các kiểu dữ liệu cài sẵn.
- Ví dụ:

```
Phanso z(1,3), z1(2,4), z2(5,4);
z = z1 + z2;
z = z1 + z2*z1 + Phanso(3,1);
```

Các toán tử của C++

- Các toán tử được chia thành hai loại theo số toán hạng nó chấp nhận
  - **Toán tử đơn** nhận một toán hạng
  - **Toán tử đôi** nhận hai toán hạng
- Các toán tử đơn lại được chia thành hai loại
  - **Toán tử trước** đặt trước toán hạng
  - **Toán tử sau** đặt sau toán hạng



Các toán tử của C++ (tt)

- Một số toán tử đơn có thể được dùng làm cả toán tử trước và toán tử sau: ++,--
- Một số toán tử có thể được dùng làm cả toán tử đơn và toán tử đôi: *
- Toán tử chỉ mục ("[…]") là toán tử đôi, mặc dù một trong hai toán hạng nằm trong ngoặc: arg1[arg2]
- Các từ khoá "new" và "delete" cũng được coi là toán tử và có thể được định nghĩa lại

Các toán tử overload được

+	-	*	/	%
^ &	|	!	=	<
>	+=	-=	*=	/=
~=	%=	^=	&=	|=
>>=	<<=	==	!=	<=
>=	&&	||	++	--
,	->	->*	()	[]
new	delete	new[]	delete[]	

>> <<

Các toán tử không overload được

`.`	`.*`
`::`	`:?`
`typeid`	`sizeof`
`const_cast`	`dynamic_cast`
`reinterpret_cast`	`static_cast`

Cú pháp của Operator Overloading

- Khai báo và định nghĩa toán tử thực chất không khác với việc khai báo và định nghĩa một loại hàm bất kỳ nào khác
- Sử dụng tên hàm là "operator@" cho toán tử "@“. Ví dụ: operator+
- Số lượng tham số tại khai báo phụ thuộc hai yếu tố:
  - Toán tử là toán tử đơn hay đôi
  - Toán tử được khai báo là hàm toàn cục hay phương thức của lớp

Cú pháp của Operator Overloading

aa@bb ➔ aa.operator@(bb) hoặc operator@(aa,bb)

@aa ➔ aa.operator@() hoặc operator@(aa)

aa@ ➔ aa.operator@(int) hoặc operator@(aa,int)

là phương thức của lớp

là hàm toàn cục

Các bài toán nạp chồng toán tử

- Thư viện nhập xuất iostream.h
- Toán tử gán (operator =)
- Toán tử số học (operator +, …)
- Toán tử so sánh (operator >, …)

Thư viện nhập xuất iostream.h

- << : toán tử ra
- >> : toán tử vào
- ***Ví dụ***: nhập vào giá trị cho một biến số nguyên:

```
int a;
printf("nhập a: ");
scanf("%d", &a);
```

```
int a;
cout << "nhập a: ";
cin >> a ;
```

Đặt vấn đề

```
CSoPhuc x;
x.nhap( );
x.xuat( );
```

```
CSoPhuc x;
cin >> x;
cout << x;
```

➔để giải quyết vấn đề trên phải định nghĩa ***toán tử vào*** và ***toán tử ra*** cho lớp CSoPhuc

Số phức

- Số ảo i là số thoả: $i^2 = -1$.
- Số phức z có dạng: $z = a + bi$ trong đó a được gọi là phần thực, b gọi là phần ảo.

Cho 2 số phức $z_1 = a_1 + b_1 i, z_2 = a_2 + b_2 i$. Khi đó:

- $$z_1 = z_2 \Leftrightarrow \begin{cases} a_1 = a_2 \\ b_1 = b_2 \end{cases}$$
- $$z_1 \pm z_2 = (a_1 \pm a_2) + (b_1 \pm b_2)i$$
- $$z_1 z_2 = (a_1 + b_1 i)(a_2 + b_2 i) = a_1 a_2 - b_1 b_2 + (a_1 b_2 + a_2 b_1)i$$
- $$\frac{z_1}{z_2} = \frac{a_1 + b_1 i}{a_2 + b_2 i} = \frac{(a_1 + b_1 i)(a_2 - b_2 i)}{(a_2 + b_2 i)(a_2 - b_2 i)} = \frac{a_1 a_2 + b_1 b_2 + (b_1 a_2 - b_2 a_1)i}{a_2^2 + b_2^2}$$

Khai báo lớp CSoPhuc

```
class CSoPhuc
{
    private:
        float thuc;
        float ao;
    public:
        friend istream& operator>> (istream& is, CSoPhuc& x);
        friend ostream& operator<< (ostream& os, CSoPhuc& x);
        …..
};
```

→ ***friend:*** hàm bạn của một lớp thì được phép truy xuất đến mọi thành phần thuộc tính, phương thức của lớp

Định nghĩa toán tử vào

```
istream& operator>> (istream& is, CSoPhuc& x)
{
        cout << “Nhap phan thuc: “;
        is >> x.thuc;
        cout << “Nhap phan ao: “;
        is >> x.ao;
        return is;
}
```

Định nghĩa toán tử ra

```
ostream& operator<< (ostream& os, CSoPhuc& x)
{
        os <<  “(“ <<x.thuc << “+” << x.ao << “i” << “)”;
        return os;
}
```

Sử dụng toán tử vào và toán tử ra

```
CSoPhuc a;
cin >> a;
cout << a;

CSoPhuc a, b, c, d;
cin >> a >> b >> c >> d;
cout << a << b << c << d;
```

Bài tập 1

- Hãy khai báo và định nghĩa *toán tử vào* và *toán tử ra* cho lớp CPhanSo.

```
class  CPhanSo
{
   private:
      int tu;
      int mau;
   public:
      void rutgon();
      friend istream& operator>> (istream& is, CPhanSo&  x);
      friend ostream& operator<< (ostream& os, CPhanSo&  x);
      CPhanSo operator = (CPhanSo x);
      CPhanSo operator + (CPhanSo x);
      CPhanSo operator - (CPhanSo x);
      CPhanSo operator * (CPhanSo x);
      CPhanSo operator / (CPhanSo x);
      int operator > (CPhanSo x);
      int operator < (CPhanSo x);
      int operator >= (CPhanSo x);
      int operator <= (CPhanSo x);
      int operator == (CPhanSo x);
      int operator != (CPhanSo x);
};
```

```
istream& operator>> (istream& is, CPhanSo& x)
{
    cout << "Nhap tu: ";
    is >> x.tu;
    do {
        cout << "Nhap mau: ";
        is >> x.mau;
    }while (x.mau==0);

    return is;
}

ostream& operator<< (ostream& os, CPhanSo& x)
{
    os <<  x.tu << '/' << x.mau;
    return os;
}
```

Toán tử gán (operator=)

- Khái niệm: Toán tử gán trong ngôn ngữ C được sử dụng để gán giá trị của biến này cho biến khác.

  → Mở rộng cho C++, toán tử gán được sử dụng để gán thành phần dữ liệu của đối tượng này cho đối tượng khác.

Đặt vấn đề

```
class CHocSinh
{
    private:
        char hoten[30];
        int toan;
        int van;
        float dtb;
    .......
};
```

```
CHocSinh a,b
a.nhap();
....
b = a;
```

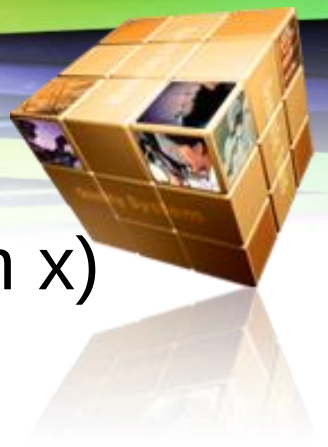
➔Để giải quyết vấn đề trên ta phải định nghĩa toán tử gán cho lớp CHocSinh

b=a → đối tượng b gọi thực hiện phương thức ***operator=*** với tham số a

Khai báo

```
class CHocSinh
{
        private:
                char hoten[30];
                int toan;
                int van;
                float dtb;
        public:
                CHocSinh operator=(CHocSinh x);
                …
};
```

Định nghĩa toán tử gán



```
CHocSinh CHocSinh::operator=(CHocSinh x)
{
        strcpy (hoten, x.hoten);
        toan = x.toan;
        van = x.van;
        dtb = x.dtb;
        return *this;
}
```

→ ***this*** là con trỏ giữ địa chỉ của đối tượng đang gọi thực hiện phương thức

→ ***this** nội dung của đối tượng mà this trỏ tới

Bài tập 2

- Hãy định nghĩa *toán tử gán* cho lớp **CSoPhuc.**
- Hãy định nghĩa *toán tử gán* cho lớp **CPhanSo.**

ĐN nạp chồng toán tử gán cho lớp CSoPhuc

```
CSoPhuc  CSoPhuc::operator = (CSoPhuc x)
{
    thuc = x.thuc;
    ao = x.ao;
    return *this;
}
```

ĐN nạp chồng toán tử gán cho lớp CPhanSo

```
CPhanSo  CPhanSo::operator=(CPhanSo x)
{
    tu = x.tu;
    mau = x.mau;
    return *this;
}
```

Toán tử số học

- toán tử cộng (operator +)
- toán tử trừ (operator -)
- toán tử nhân (operator *)
- toán tử chia (operator /)
- toán tử mod (operator %)

Toán tử số học

- toán tử cộng bằng (operator +=)
- toán tử trừ bằng (operator -=)
- toán tử nhân bằng (operator *=)
- toán tử chia bằng (operator /=)
- toán tử mod bằng (operator %=)
- toán tử tăng một (operator ++)
- toán tử trừ một (operator --)

Đặt vấn đề

```
CSoPhuc  a, b, kq;
a.nhap();
b.nhap();
kq = a.tong(b);
```

→ kq = a + b; ???

➔Phải định nghĩa toán tử gán và toán tử cộng cho lớp CSoPhuc

Khai báo

```
class CSoPhuc
{
    private:
        float thuc;
        float ao;
    public:
        CSoPhuc operator = (CSoPhuc x);
        CSoPhuc operator + (CSoPhuc x);
        CSoPhuc operator - (CSoPhuc x);
        CSoPhuc operator * (CSoPhuc x);
        …..
};
```

```
CSoPhuc  CSoPhuc::operator+(CSoPhuc x)
{
    CSoPhuc  temp;
    temp.thuc = thuc + x.thuc;
    temp.ao = ao + x.ao;
    return  temp;
}

CSoPhuc  CSoPhuc::operator - (CSoPhuc x)
{
    CSoPhuc  temp;
    temp.thuc = thuc - x.thuc;
    temp.ao = ao - x.ao;
    return  temp;
}

CSoPhuc  CSoPhuc::operator * (CSoPhuc x)
{
    CSoPhuc  temp;
    temp.thuc = thuc * x.thuc - ao * x.ao;
    temp.ao = thuc * x.ao + ao * x.thuc;
    return  temp;
}
```

Bài tập 3

- Hãy định nghĩa các toán tử cộng, toán tử trừ, toán tử nhân, toán tử chia cho lớp **CPhanSo**

ĐN nạp chồng toán tử cộng

```
CPhanSo  CPhanSo::operator+(CPhanSo x)
{
    CPhanSo  temp;
    temp.tu = tu * x.mau + mau * x.tu;
    temp.mau = mau * x.mau;
    return  temp;
}
```

ĐN nạp chồng toán tử trừ

```
CPhanSo  CPhanSo::operator-(CPhanSo x)
{
    CPhanSo  temp;
    temp.tu = tu * x.mau - mau * x.tu;
    temp.mau = mau * x.mau;
    return  temp;
}
```

ĐN nạp chồng toán tử nhân

```
CPhanSo  CPhanSo::operator*(CPhanSo x)
{
    CPhanSo  temp;
    temp.tu = tu * x.tu;
    temp.mau = mau * x.mau;
    return  temp;
}
```

ĐN nạp chồng toán tử chia

```
CPhanSo  CPhanSo::operator/(CPhanSo x)
{
    CPhanSo  temp;
    temp.tu = tu * x.mau;
    temp.mau = mau * x.tu;
    return  temp;
}
```

Toán tử so sánh

- toán tử so sánh lớn hơn (operator >)
- toán tử so sánh nhỏ hơn (operator <)
- toán tử so sánh lớn hơn hoặc bằng (operator >=)
- toán tử so sánh nhỏ hơn hoặc bằng (operator <=)
- toán tử so sánh bằng (operator ==)
- toán tử so sánh khác (operator !=)

Đặt vấn đề

```
int x, y;
....
if (x > y)
{
    .....
}
```

```
CPhanSo a, b;
....
if (a > b)
{
    .....
}
```

→ Định nghĩa ***toán tử so sánh lớn hơn*** cho lớp CPhanSo

Khai báo lớp

```
Class CPhanSo
{
    private:
        int tu;
        int  mau;
    public:
        CPhanSo operator – (CPhanSo x);
        int operator > (CPhanSo x);
        int operator < (CPhanSo x);
        int operator >= (CPhanSo x);
        int operator <= (CPhanSo x);
        int operator == (CPhanSo x);
        int operator != (CPhanSo x);
        ……
};
```

Bài tập 4

- Hãy định nghĩa các toán tử so sánh cho lớp *CPhanSo*

ĐN toán tử so sánh lớn hơn cho lớp CPhanSo

```
int   CPhanSo::operator>(CPhanSo x)
{
        CPhanSo  temp = *this-x;
        if (temp.tu * temp.mau > 0 )
            return  1;
        else
            return  0;
}
```

ĐN toán tử so sánh nhỏ hơn cho lớp CPhanSo

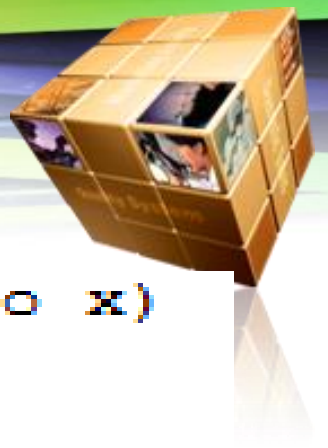
```
int  CPhanSo::operator<(CPhanSo x)
{
      CPhanSo  temp = *this-x;
      if (temp.tu * temp.mau < 0 )
         return  1;
      else
         return  0;
}
```

ĐN toán tử so sánh >= và <= cho lớp CPhanSo

```
int   CPhanSo::operator>=(CPhanSo x)
{
        CPhanSo  temp = *this->x;
        if (temp.tu * temp.mau >= 0 )
            return  1;
        else
            return  0;
}

int   CPhanSo::operator<=(CPhanSo x)
{
        CPhanSo  temp = *this->x;
        if (temp.tu * temp.mau <= 0 )
            return  1;
        else
            return  0;
}
```

ĐN toán tử so sánh == và != cho lớp CPhanSo



```
int   CPhanSo::operator==(CPhanSo x)
{
        CPhanSo  temp = *this-x;
        if (temp.tu * temp.mau == 0 )
            return  1;
        else
            return  0;
}

int   CPhanSo::operator!=(CPhanSo x)
{
        CPhanSo  temp = *this-x;
        if (temp.tu * temp.mau != 0 )
            return  1;
        else
            return  0;
}
```

Test 2

- Thiết kế và khai báo lớp *CPhanSo*
  - Cài đặt các phương thức nhập, xuất cho lớp CPhanSo
  - Định nghĩa toán tử vào và toán tử ra cho lớp CPhanSo
  - Định nghĩa nạp chồng toán tử gán
  - Định nghĩa nạp chồng các toán tử số học (+ , - , * , /) trên lớp CPhanSo
  - Định nghĩa các toán tử so sánh (>,<,>=,<=,==,!=)
- Viết chương trình nhập vào 2 phân số. Cho biết kết quả tính tổng, hiệu, tích, thương của 2 phân số này. Thực hiện so sánh 2 phân số.