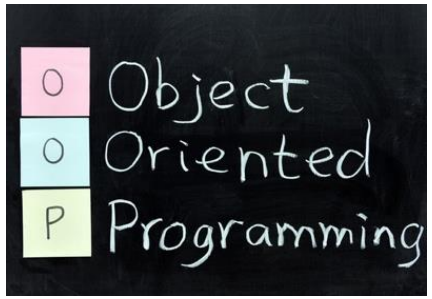



Môn học

Lập trình hướng đối tượng

Đỗ Phúc Hảo

Khoa Công nghệ thông tin




Nội dung

❖Chương 4: Đa năng hoá toán tử

- Toán tử trên lớp
- Hàm toán tử
- Một số ví dụ
- Chồng toán tử nhập và xuất
- Chuyển đổi kiểu




1. Toán tử lớp

❖Trong C++ có thể định nghĩa chồng đối với hầu hết các toán tử (một ngôi hoặc hai ngôi) trên các lớp, nghĩa là một trong các toán hạng tham gia phép toán là đối tượng

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




2. Hàm toán tử

- Hàm toán có thể khai báo là hàm thành phần của lớp hoặc là hàm bạn của lớp.
  - Khi hàm toán tử là hàm thành phần của lớp thì toán hạng thứ nhất trong phép toán phải là đối tượng thuộc lớp đó và số đối số của hàm toán tử bằng số ngôi của phép toán trừ đi một.




2. Hàm toán tử (t)

❖Khi hàm toán tử là hàm bạn của lớp thì toán hạng thứ nhất trong phép toán không nhất thiết phải là đối tượng thuộc lớp đó và số đối số của hàm toán tử bằng số ngôi của phép toán.




2. Hàm toán tử (t)

```
class sp {
        private:
                double a, b;
        public:
                sp(double a = 0, double b = 0);
                void Xuat();
                sp operator+(sp u); // Toán tử cộng hai đối tượng sp
                friend sp operator+(double x, sp u); // cộng số thực với sp
                void operator-(); // Toán tử một ngôi
};
```




2. Hàm toán tử (t)

```
sp::sp(double a, double b){
        this->a = a; this->b = b;
}
sp sp::operator+(sp u){
        sp res(a + u.a, b + u.b); // hàm thiết lập hai đối số.
        return res;
}
```




2. Hàm toán tử (t)

```
void sp::operator-(){
        a = -a; b = -b;
}
void sp::Xuat(){
        cout<<a <<“+” <<b<<“i”<<endl;
}
sp operator+(double x, sp u){ /* hàm friend */
        sp res(x + u.a, u.b);
        return res;
}
```




2. Hàm toán tử (t)

```
int main(){
        sp a(2,3);
        sp b(4,5);
        sp c = a + b;
        c.Xuat();
        sp d = a + 3;
        d.Xuat();
        -d;
        d.Xuat();
}
```




2. Hàm toán tử (t)

❖Thay vì phải khai báo cả hai hàm toán tử

```
sp operator+(sp u);
friend sp operator+(double x, sp u);
```

❖Để chạy được với 3 câu lệnh

```
sp c = a + b; c = a + 3; hoặc d = 4 + b;
```

❖Ta có thể thay bằng chỉ một hàm toán tử

```
friend sp operator +(sp u, sp v);
```




3. Một số ví dụ tiêu biểu

❖Chồng toán tử gán (=)

- Việc khai báo tường minh toán tử gán chỉ cần thiết khi các đối tượng tham gia phép gán có thành phần dữ liệu kiểu con trỏ.
- Khi trong lớp không khai báo một hàm toán tử gán nào thì trình biên dịch sẽ tự động cung cấp cho lớp một hàm toán tử gán mặc định để thực hiện câu lệnh gán hai đối tượng (hàm đó sẽ không giải quyết được vấn đề với phần dữ liệu kiểu con trỏ)




3. Một số ví dụ tiêu biểu (t)

❖Ví dụ:

```
class mang{
        private:
                int n, *a;
        public:
                mang(int t):n(t){};
                mang& operator=(mang &u);
};
```

❖Hàm thiết lập sao chép [mang(mang &u);]




3. Một số ví dụ tiêu biểu (t)

```
mang& mang::operator=(mang &u){
        cout << “dinh nghia phep gan”;
        if(a != NULL) delete []a;
        n = u.n;
        a = new int [n];
        for(int i=0;i<n;i++) a[i] = u.a[i];
        return *this;
}
```

```
mang::mang(mang &u){
        cout << "Ham sao chep " << endl;
        n = u.n;
        a = new int [n];
        for (int i = 0; i < n;i++){
                a[i] = u.a[i];
        }
}
```




3. Một số ví dụ tiêu biểu (t)

```
int main(){
        mang a(3);
        a.Xuat();
        mang b = a; // gọi hàm thiết lập sao chép
        b = a; // Biến b đã được tạo → hàm định nghĩa phép =
        b.Xuat();
        return 0;
}
```




4. Chồng toán tử nhập & xuất

❖Chồng toán tử nhập (>>) và xuất (<<)

- Nhằm cho phép các đối tượng đứng bên phải chúng khi thực hiện thao tác nhập xuất
- Khai báo như là hàm bạn (vì các toán hạng đứng bên trái chúng là các đối tượng thuộc lớp ostream và istream)




4. Chồng toán tử nhập & xuất (t)

Khai báo:

```
friend istream& operator>>(istream &is, sp &u);
friend ostream& operator<<(ostream &os, sp u);
```

Định nghĩa

```
istream& operator>>(istream &is, sp &u){
    is>>u.a>>u.b;
    return is;
}
```




4. Chồng toán tử nhập & xuất (t)

Định nghĩa

```
ostream& operator<<(ostream &os, sp u){
        os<<u.a<<" "<<u.b;
        return os;
}
int main(){
        sp a; cin>>a; cout<<a;
}
```




4. Chồng toán tử nhập & xuất (t)

- Lưu ý: Giá trị trả về của hàm toán tử nhập/xuất được chọn là trả về tham chiếu đến đối tượng cin/cout nhằm cho phép thực hiện nhập xuất liên tiếp nhiều đối tượng.




5. Chuyển đổi kiểu (ngầm định)

- Được định nghĩa thông qua một hàm thiết lập chuyển kiểu cho lớp, với đối số có kiểu cần phải chuyển thành một đối tượng thuộc lớp đó.
- Không thể sử dụng để chuyển các đối tượng của lớp mình sang kiểu khác và chúng chỉ có thể được sử dụng trong phép gán và phép khởi tạo giá trị.




5. Chuyển đổi kiểu (ngầm định) (t)

```
class sp {
        private:
                double a, b;
        public:
                sp(double a = 0, double b = 0);
                void xuat();
                friend sp operator+(sp u, sp v);
};
```




5. Chuyển đổi kiểu (ngầm định) (t)

```
int main(){
        sp p(1,2);
        sp p1(2,-3);
        sp res = p + p1; // Thiết lập chuyển kiểu ngầm định
        res.xuat();
        return 0;
}
```




5. Chuyển đổi kiểu (ép buộc)

- Phép chuyển kiểu tường minh được xác định thông qua toán tử chuyển kiểu, toán tử này có thể được dùng để chuyển các đối tượng sang kiểu khác.
- C++ quy ước hàm toán tử chuyển kiểu phải là hàm thành phần của lớp liên quan và không có đối số.




5. Chuyển đổi kiểu (ép buộc) (t)

Dạng tổng quát của hàm toán tử chuyển kiểu:

```
operator <type> () {
    return <value>;
}
```

- <type> là tên của kiểu dữ liệu mà một đối tượng sẽ được chuyển sang.
- <value> là giá trị của đối tượng sau khi được chuyển.




5. Chuyển đổi kiểu (ép buộc) (t)

```
class sp {
        private:
                double a, b;
        public:
                sp(double a = 0, double b = 0);
                void xuat();
                friend sp operator+(sp u, sp v);
                // Hàm toán tử chuyển kiểu (ép buộc)
                operator double();
};
```




5. Chuyển đổi kiểu (ép buộc) (t)

```
int main(){
        sp p(1,2);
        sp p1(2,-3);
        sp res = p + p1;
        res.xuat();

        double n = p;
        cout << n;
        return 0;
}
```

```
sp::operator double(){
        return a;
}
```




Thank you!

Any questions?