

CHƯƠNG 4

TOÁN TỬ TẢI BỘI

(Operator Overloading)

Chương 4 sẽ tìm hiểu cách thức để các toán tử của C++ được mở rộng nhằm thực hiện các chức năng khác nhau – gọi là quá tải toán tử.

Các toán tử khi được quá tải sẽ được gọi là toán tử tải bội.

Chương 4 trình bày các vấn đề sau: Định nghĩa toán tử tải bội, quá tải các toán tử xuất nhập dữ liệu, ++, --, [], ().

4.1. Định nghĩa toán tử tải bội

Các toán tử cùng tên thực hiện nhiều chức năng khác nhau được gọi là toán tử tải bội. Dạng định nghĩa tổng quát của toán tử tải bội như sau:

Kiểu_trả_về **operator op**(các tham số)
{*//thân toán tử*}

Trong đó: Kiểu_trả_về là kiểu kết quả thực hiện của toán tử.

op là tên toán tử tải bội (có trong C++)

operator op(các tham số) gọi là *hàm toán tử* tải bội, nó có thể là hàm thành phần hoặc là hàm bạn.

các tham số được khai báo tương tự khai báo biến nhưng phải tuân theo những quy định sau:

- Nếu toán tử tải bội là **hàm thành phần** thì:
 - **Không** có tham số cho toán tử một ngôi và
 - **Một** tham số cho toán tử hai ngôi.
- Nếu toán tử tải bội là **hàm bạn** thì:
 - **Một** tham số cho toán tử một ngôi và
 - **Hai** tham số cho toán tử hai ngôi.

Quá trình xây dựng toán tử tải bội được thực hiện như sau:

- Định nghĩa lớp để xác định kiểu dữ liệu sẽ được sử dụng trong các toán tử tải bội
- Khai báo hàm toán tử tải bội trong vùng **public** của lớp
- Định nghĩa nội dung cần thực hiện

Chú ý:

- Chỉ có thể định nghĩa tải bội cho các toán tử có sẵn của C++.
- Các toán tử tải bội được gọi thực hiện thư thông thường

4.2. Một số lưu ý khi xây dựng toán tử tải bội

1. Trong C++ ta có thể đưa ra nhiều định nghĩa mới cho hầu hết các toán tử trong C++, ngoại trừ những toán tử sau đây: '.', '::', 'sizeof', '?:'
2. Mặc dầu ngữ nghĩa của toán tử được mở rộng nhưng cú pháp, các quy tắc văn phạm như số toán hạng, quyền ưu tiên và thứ tự kết hợp thực hiện của các toán tử vẫn không có gì thay đổi.
3. Không thể thay đổi ý nghĩa cơ bản của các toán tử đã định nghĩa trước, ví dụ không thể định nghĩa lại các phép toán +, - đối với các số kiểu int, float.
4. Các toán tử =, (), [], -> yêu cầu hàm toán tử phải là hàm thành phần của lớp, không thể dùng hàm bạn để định nghĩa toán tử tải bội.

Ví dụ 1: Lớp phân số

Ví dụ 2: Lớp số phức

```

#include <iostream>
using namespace std;
class phanso
{
    int t,m;
public:
    void nhap();
    void xuat();
    phanso operator -();    //lay doi cua phan so
    phanso operator +(phanso p2);    //cong 2 phan so
    phanso operator -(phanso p2);    //tru 2 phan so
    phanso operator *(phanso p2);    //nhan 2 phan so
};
void phanso::nhap()
{
    cout<<"\nNhap tu so:";cin>>t;
    do
    {
        cout<<"\nNhap mau so:";cin>>m;
    } while(m==0);
}

void phanso::xuat()
{
    cout<<t<<"/"<<m<<endl;
}

```



```

phanso phanso::operator -()
{
    phanso p;
    p.t = -t;
    p.m = m;
    return p;
}

```

```

phanso phanso::operator +(phanso p2)
{
    phanso p;    //t/m + p2.t/p2.m
    p.t = t*p2.m + m*p2.t;
    p.m = m*p2.m;
    return p;
}

```

```

phanso phanso::operator -(phanso p2)
{
    phanso p;    //t/m - p2.t/p2.m
    p.t = t*p2.m - m*p2.t;
    p.m = m*p2.m;
    return p;
}

```

```

phanso phanso::operator *(phanso p2)
{
    phanso p;    //t/m * p2.t/p2.m
    p.t = t*p2.t;
    p.m = m*p2.m;
    return p;
}

```

```

int main()
{
    phanso p1,p2,p3,p4,p5;
    cout<<"\n Nhap phan so p1: ";p1.nhap();
    cout<<"\n Nhap phan so p2: ";p2.nhap();
    cout<<"\n p1: ";p1.xuat();
    cout<<"\n p2: ";p2.xuat();
    p3 = p1+p2; //goi cac toan tu + nhu cach dung thong thuong
    cout<<"\n p3 = p1 + p2 = "; p3.xuat();
    p4 = p1 + p2 * p3;
    cout<<"\n p4 = p1 + p2 * p3 = "; p4.xuat();
    p5 = -p1;
    cout<<"\n Doi cua p1 la: "; p5.xuat();
}

```

Thảo luận:

1. Chuyển các phương thức (-, +, -, *) trong ví dụ trên thành hàm bạn.
2. Viết chương trình cho Ví dụ 2. (các phép toán tải bội: +, -, *, / trên lớp Sophuc).
3. Xét lớp phanso. Hãy quá tải các toán tử thích hợp để thực hiện được biểu thức sau:

$$p3 = k + p1 * p2 + m;$$

Trong đó $p1, p2, p3$ là các đối tượng phân số, k và m là các số nguyên.