

Bài giảng trên lớp

Bài giảng trên lớp là tài liệu gợi ý và hỗ trợ giảng viên trong quá trình lên lớp, tuy nhiên giảng viên cần có sự chuẩn bị của riêng mình cho phù hợp với từng lớp học.

Nếu giảng viên thiết kế bài giảng tốt hơn tài liệu đã cung cấp, xin hãy chủ động làm và gửi lại cho chúng tôi.

Nếu giảng viên cần thay đổi tài liệu đã cung cấp, những thay đổi có liên quan đến cấu trúc, nội dung kiến thức và có tính đổi mới, xin hãy chủ động làm và gửi lại cho chúng tôi.

Mọi ý kiến xin gửi cho Phòng NC-PTCT để được xem xét và ban hành chính thức và góp phần ngày càng hoàn thiện hơn học liệu của trường.

Trân trọng cảm ơn.

Liên hệ: Phòng NC-PTCT – FPT Polytechnic

Những ý tưởng đổi mới sẽ được xem xét và gửi qua Dự án CNGD để có sự hỗ trợ giảng viên làm nghiên cứu khoa học, hoặc hướng dẫn viết bài báo đăng trên tạp chí CNGD.

Lập trình C++

Bài 7 Toán tử nạp chồng

Hệ thống bài cũ

- Tổng quan về Lớp
- Hàm thành viên nội tuyến (inline)
- Hàm xây dựng, hàm hủy, hàm bạn, lớp bạn
- Đối số mặc định, toán tử phạm vi
- Danh sách khởi tạo thành viên
- Thành viên hằng, tĩnh tham chiếu
- Thành viên là đối tượng của 1 lớp
- Mảng các đối tượng
- Cấu trúc (structure) và hợp (union)

MỤC TIÊU

- Hiểu về toán tử nạp chồng và cách sử dụng nó

Nội dung

- Tổng quan về toán tử nạp chồng
- Cú pháp khai báo toán tử
- Phân loại hàm toán tử
- Toán tử chuyển kiểu
- Con trỏ this

Định nghĩa nạp chồng toán tử

- Nạp chồng toán tử là khả năng định nghĩa các hàm thành phần của lớp dưới dạng các hàm toán tử
- Các toán tử **cùng tên** thực hiện được nhiều chức năng khác nhau được gọi là **nạp chồng toán tử**
- Ví dụ:

```
Date d1(12,3,1989);
```

```
Date d2;
```

```
d2.add_days(d1,45);
```

- Có thể được viết:

```
d2=d1+45;
```

Khai báo hàm toán tử

Khai báo tổng quát

<return – type> operator #(ds đối số)

{

//định nghĩa hàm toán tử

};

Trong đó:

return-type: kiểu trả lại kết quả của hàm toán tử

#:tên toán tử cần định nghĩa (+,-,*,/,...)

Hàm toán tử được định nghĩa trong vị trí **public** của phần khai báo lớp

Phân loại hàm toán tử

Toán tử độc lập:

- Không thuộc lớp nào.
- Ngôi của toán tử là số tham số truyền vào.

```
Phanso operator +(const Phanso &p1, const Phanso &p2);
```

```
bool operator +(const Phanso &p1, const Phanso &p2);
```

Toán tử thuộc lớp:

- Là phương thức của lớp
- Ngôi của toán tử: đối tượng của lớp + số tham số

```
Phanso Phanso:: operator +(const Phanso &p);
```

```
bool Phanso:: operator +(const Phanso &p);
```

Cách sử dụng 2 loại là như nhau.

Phân loại hàm toán tử

Hàm thành phần: hàm này không có đối số cho toán tử 1 ngôi hay có một đối số cho toán tử 2 ngôi \Leftrightarrow tương tự hàm thành phần thông thường

Hàm bạn: hàm có một đối số cho toán tử một ngôi và hai đối số cho toán tử 2 ngôi. Hàm được khai báo thêm từ khóa **friend** trước tên hàm \Leftrightarrow tương tự hàm bạn

Quy tắc sử dụng toán tử

Loại	Khai báo	Sử dụng
Thành phần	type operator #();	a.operator #(); hoặc #a;
	type operator #(type b);	a.operator #(b); hoặc a#b;
Hàm bạn	friend type operator #(type a);	#a
	friend type operator #(type a, type b);	a#b

Trong đó: # là toán tử cần định nghĩa

Ví dụ sử dụng toán tử

Loại	Khai báo	Sử dụng
Thành phần	PS operator -();	kq= a.operator -(); hoặc kq=-a;
	PS operator -(PS b);	kq=a.operator -(b); hoặc kq=a-b;
Hàm bạn	friend PS operator -(PS a);	kq=-a
	friend PS operator -(PS a, PS b);	kq=a-b

Ưu điểm – Hạn chế

Ưu điểm: thực hiện toán tử trên kiểu dữ liệu tự định nghĩa.

Hạn chế:

Không thể tạo toán tử mới

Không thể định nghĩa lại toán tử trên kiểu cơ bản

Ngôi của toán tử giữ nguyên

Độ ưu tiên của toán tử không đổi



DEMO

DEMO khai báo nạp
chồng toán tử



Vấn đề

Viết chương trình xây dựng lớp Phân số. Yêu cầu định nghĩa các hàm:

Nhập và xuất phân số

Hàm toán tử + để tính tổng hai phân số

Hàm toán tử - để tính hiệu hai phân số (friend)

Áp dụng nhập vào hai phân số và in ra kết quả



Khai báo lớp phân số

```
class ps{  
    int ts, ms;  
    public:  
    //định nghĩa hàm nhập xuất  
    void nhap();  
    void xuat();  
    //định nghĩa toán tử +, -  
    ps operator+(ps b);  
    friend ps operator-(ps a, ps b);  
};
```

Xây dựng hàm nhập, xuất

```
void ps::xuat( )
{
    if(ms==1) cout<< ts;
    else {
        if(ts==0) cout<<"0";
        else
            cout<<ts<<"/"<<ms;
    }
}
```

```
void ps::nhap( )
{
    cout<<"Nhap tu so:";
    cin>>ts;

    cout<<"Nhap mau so:";
    cin>>ms;

    if (ms<0){
        ts=-ts;
        ms=-ms;
    }
}
```


Xây dựng hàm nạp chồng toán tử +,-

```
ps ps::operator+(ps b){  
    ps kq;  
    kq.ts = ts*b.ms+b.ts*ms;  
    kq.ms = ms*b.ms;  
    return kq;  
}
```

```
ps operator-(ps a,ps b){  
    ps kq;  
    kq.ts = a.ts*b.ms-b.ts*a.ms;  
    kq.ms = a.ms*b.ms;  
    return kq;  
}
```

Gọi các hàm đã xây dựng

```
void main( ){  
    ps x,y,kq;  
    cout<<"Nhap phan so thu nhat.\n";    x.nhap( );  
    cout<<"\nNhap phan so thu hai.\n";    y.nhap( );  
    kq = x+y;  
    cout<<"\nTong hai phan so: "; kq.xuat();  
    kq = x-y;  
    cout<<"\nHieu hai phan so: "; kq.xuat();  
    system("pause");  
}
```

Mã nguồn và kết quả

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

class ps{
    int ts, ms;
public:
    //định nghĩa hàm nhập xuất
    void nhap();
    void xuat();
    //định nghĩa toán tử +, -
    ps operator+(ps b);
    friend ps operator-(ps a, ps b);
};

void ps::xuat( ){
    if(ms==1) cout<< ts;
    else {
        if(ts==0) cout<<"0";
        else cout<<ts<<"/"<<ms;
    }
}

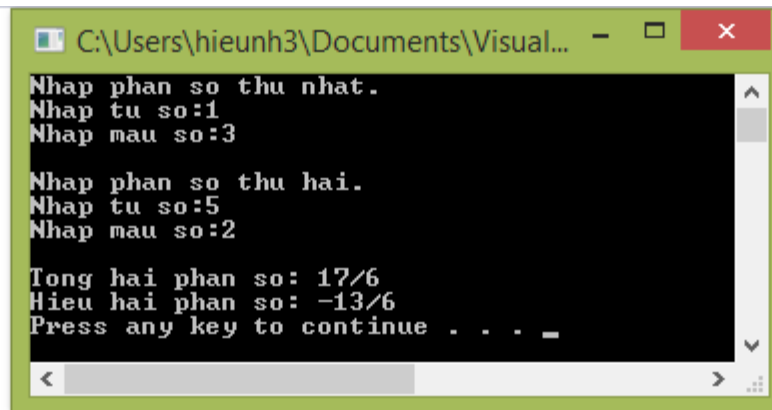
void ps::nhap( ){
    cout<<"Nhập tu số:"; cin>>ts;
    cout<<"Nhập mẫu số:"; cin>>ms;
    if (ms<0){
        ts=-ts;
        ms=-ms;
    }
}
```

```
ps ps::operator+(ps b){
    ps kq;
    kq.ts = ts*b.ms+b.ts*ms;
    kq.ms = ms*b.ms;
    return kq;
}

ps operator-(ps a,ps b){
    ps kq;
    kq.ts = a.ts*b.ms-b.ts*a.ms;
    kq.ms = a.ms*b.ms;
    return kq;
}
```

```
void main( ){
    ps x,y,kq;
    cout<<"Nhập phân số thứ nhất.\n";    x.nhap( );
    cout<<"\nNhập phân số thứ hai.\n";    y.nhap( );

    kq = x+y;
    cout<<"\nTong hai phân số: "; kq.xuat();
    kq = x-y;
    cout<<"\nHieu hai phân số: "; kq.xuat();
    cout<<endl;
    system("pause");
}
```



```
Nhập phân số thứ nhất.
Nhập tử số:1
Nhập mẫu số:3

Nhập phân số thứ hai.
Nhập tử số:5
Nhập mẫu số:2

Tong hai phân số: 17/6
Hieu hai phân số: -13/6
Press any key to continue . . . _
```

Hàm toán tử

Toán tử có thể định nghĩa lại:

Ngôi	Nhóm	Toán tử
1 Ngôi	Tăng giảm	++, --
	Dấu số học	+, -
	Logic	!
	Con trỏ	*, &
	Ép kiểu	Int,float,double
2 Ngôi	Số học	+, -, *, /, %, +=, -=, *=, /=, %=
	So sánh	>, <, ==, >=, <=, !=
	Logic	&&,
	Nhập xuất	<<, >>
	Gán	=
	Lấy chỉ số mảng	[]

Hàm toán tử

Toán tử không thể định nghĩa lại:

Toán tử	Ý nghĩa
'	Truy xuất phần tử
::	Toán tử ::
?:	Toán tử điều kiện
#	Chỉ thị tiền xử lý
##	Chỉ thị tiền xử lý



DEMO

DEMO nạp chồng
toán tử số học



Nạp chồng toán tử so sánh

Sáu toán tử so sánh `<`, `>`, `<=`, `>=`, `==`, và `!=` có thể được nạp chồng tương tự cách mà các toán tử số học được nạp chồng dưới dạng các hàm **friend**.

Ví dụ:

```
class Ration{  
    friend bool operator==(const Ratio&, const Ratio&);;  
    bool operator==(const Ratio& x, const Ratio& y){  
        return (x.num * y.den == y.num * x.den);;  
    }  
};
```



DEMO

**DEMO nạp chồng
toán tử so sánh**



Nạp chồng toán tử tăng, giảm

Toán tử tăng `++` và toán tử giảm `--` đều có hai dạng: **tiền tố** và **hậu tố**. Cả bốn dạng này đều có thể được nạp chồng.

Ví dụ:

```
class PT{  
    public:  
        PT operator++();  
    private: int num, den;  
};  
  
PT PT::operator++(){  
    num += den;  
    return *this;  
}
```



DEMO

DEMO nạp chồng
toán tử tăng, giảm



Định nghĩa toán tử dòng xuất (<<)

Khai báo:

```
friend ostream & operator << (ostream &out, class-name c);
```

Định nghĩa:

```
ostream & operator << (ostream &out, class-name c)
{
    //định nghĩa thao tác in các giá trị của "c" ra màn hình.
    // Lưu ý dùng dòng out<< để xuất thay vì dùng cout<<
    return out;
}
```

Trong đó

ostream : tham chiếu chỉ đến dòng xuất

out: tên dòng xuất tạm thời sử dụng thay cho cout

class-name: tên của lớp đối tượng cần định nghĩa toán tử xuất

Định nghĩa toán tử dòng xuất (<<)

Ví dụ

```
class PT{  
    friend ostream& operator<<(ostream&, const PT&);  
    public:  
        PT(int n=0, int d=1) : num(n), den(d) { };  
    private: int num, den;  
};  
ostream& operator<<(ostream& ostr, const PT& r){  
    return ostr << r.num << '/' << r.den;  
}
```



DEMO

DEMO nạp chồng
toán tử dòng xuất



Định nghĩa toán tử dòng nhập (>>)

Khai báo

```
friend istream & operator >> (istream &in, class-name &c);
```

Định nghĩa

```
istream & operator >> (istream &in, class-name &c)
{
    //định nghĩa thao tác nhập các giá trị cho đối tượng "c".
    // Lưu ý dùng dòng in >> để nhập thay vì dùng cin>>
    Return in;
};
```

Trong đó

istream : tham chiếu chỉ đến dòng nhập

in: tên dòng nhập tạm thời sử dụng thay cho cin

class-name: tên của lớp đối tượng cần định nghĩa toán tử nhập

Định nghĩa toán tử dòng nhập (>>)

Ví dụ

```
class Ratio {  
    friend ostream& operator >>(ostream&, Ratio&);  
public:  
    Ratio(int n=0, int d=1) : num(n), den(d) { };  
    // các khai báo khác đặt ở đây private:  
private: int num, den;  
};  
ostream& operator >>(ostream& istr, Ratio& r){  
    cout << "\tNumerator: ";    istr >> r.num;  
    cout << "\tDenominator: "; istr >> r.den;  
    return istr;  
}
```



DEMO

DEMO nạp chồng
toán tử dòng nhập



Toán tử chuyển kiểu

Định nghĩa toán tử chuyển kiểu

- Toán tử chuyển đổi kiểu được dùng để chuyển đổi một đối tượng của lớp thành đối tượng lớp khác hoặc thành đối tượng của một kiểu dữ liệu đã có sẵn
- Hàm chuyển đổi kiểu phải là hàm thành viên không tĩnh và không hàm bạn

Cú pháp: **operator** <kiểu_cần_chuyển> ();

Ví dụ:

```
class date {  
    public:  
        operator int();//chuyển kiểu date về kiểu số nguyên  
};
```



DEMO

DEMO nạp chồng
toán tử chuyển kiểu



Định nghĩa toán tử chỉ số dưới

- Thông thường để xuất ra giá trị của 1 phần tử tại vị trí cho trước trong đối tượng.
- Định nghĩa là hàm thành viên.

```
class StrVec {  
    public:  
        StrVec (const int dim);  
        ~StrVec ();  
        char* operator [] (int);  
        int add(char* );  
        // .....  
    private:  
        char **elems; // các phần tử  
        int dim; // kích thước của vecto  
        int used; // vị trí hiện tại  
};
```

```
char* StrVec::operator [] (int i) {  
    if ( i>=0 && i<used) return elems[i];  
    return "";  
}  
  
void main() {  
    StrVec sv1(100);  
    sv1.add("PTPhu"); sv1.add("BTThan");  
    sv1.add("NVLan"); sv1.add("CTHuy");  
    cout<< sv1 [2]<<endl;  
    cout<< sv1 [0];  
}
```



DEMO

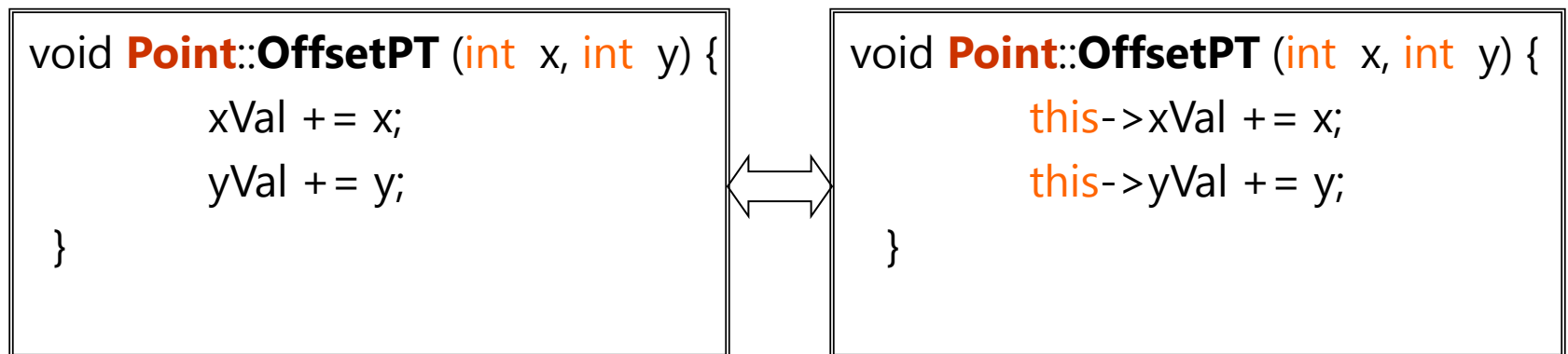
DEMO nạp chồng
toán tử chỉ số dưới



Con trỏ this

Con trỏ *this:

- Là 1 thành viên ẩn, có thuộc tính là **private**.
- Trỏ tới chính bản thân đối tượng.



- Có những trường hợp sử dụng *this là dư thừa (Ví dụ trên)
- Tuy nhiên, có những trường hợp phải sử dụng con trỏ ***this**

Toán tử phạm vi

Toán tử **::** dùng để xác định chính xác hàm (thuộc tính) được truy xuất thuộc lớp nào.

Câu lệnh: `pt.OffsetPT(2,2);`
 `<=> pt.Point::OffsetPt(2,2);`

Cần thiết trong một số trường hợp:

- Cách gọi hàm trong thừa kế.
- Tên thành viên bị che bởi biến cục bộ.

Ví dụ: `Point (int xVal, int yVal)`
 `{`
 `Point::xVal = xVal;`
 `Point::yVal = yVal;`
 `}`

Kết luận

- Tổng quan về toán tử nạp chồng
- Cú pháp khai báo toán tử
- Phân loại hàm toán tử
- Toán tử chuyển kiểu
- Con trỏ this

Chuẩn bị bài sau

Sinh viên đọc sách và slide trước bài học kế tiếp về **kết hợp và kế thừa** gồm:

Kết hợp và kế thừa

Các thành viên protected, private

Hàm virtual và tính đa hình

Hàm hủy



FPT POLYTECHNIC

THANK YOU!

www.poly.edu.vn