DỊCH SÁCH C++

CHƯƠNG 18: NẠP CHỒNG

Nạp chồng hàm:

Là các hàm có tên giống nhau, nhưng khác nhau về kiểu dữ liệu của tham số. Khi gọi tên, hàm sẽ được truy cập tới dựa vào tham số truyền vào.

VD:

void tong( int a, int b)

{

return a+b;

}

void tong( float a, float b)

{

return a+b;

}

int main()

{

tong(2,3); // truy cap ham tong( int a, int b)

tong(3.5,6.8); //truy cap ham tong( float a, float b)

Nạp chồng toán tử:

Toán tử có thể nạp chồng:

|  |  |  |
| --- | --- | --- |
| **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 | [ ] |

Tóan tử không thể nạp chồng:

|  |  |
| --- | --- |
| **Nhóm** | **Toán tử** |
| Xác định phạm vi | :: |
| Lựa chọn thành viên | . |
| Lựa chọn thành viên thông qua con trỏ | .\* |
| Kích thước | sizeof |
| Căn chỉnh | alignof |
| Trả về thông tin lúc chạy | typeid |
| Toán tử điều kiện | ? : |
| Chỉ thị tiền xử lý | # |
| Chỉ thị tiền xử lý | ## |

Tên của hàm toán tử:

operator@(); Với @ là toán tử

VD:

class phanso{

private:

int tu; int mau;

puplic:

void operator\*(phanso a,phanso b);

};

void phanso::operator\*(phanso a,phanso b){

phanso t;

tu=a.tu\*b.tu;

mau=a.mau\*b.mau;

return t;

}

int main(){

phanso x,y;

cout<<”Tong”<<x\*y; //goi ham tinh tong

}

Hàm nạp chồng toán tử có ít nhất 1 đối số, đối số đầu tiên bao giờ cũng thuộc lớp do người dùng định nghĩa.

Đối với các phép toán tăng giảm một đối tượng:

Có 2 trường hợp: ++a hoặc a++

Đối với trường hợp ++a ta viết tên hàm : operator++a();

Đối với trường hợp a++ ta viết tên hàm : operator++a(int);

Có những phép toán mà đối với dữ liệu có sẵn, chương trình có thể tự hiểu và chuyển đổi như –a tương tương với a -= 1 hay là a=a-1. Nhưng với nạp chồng khi ta viết opertator +=() không có nghĩa nó là sự kết hợp giữ operator+() và operator=().

Với toán tử gán( =) giá trị được cung cấp cho đối tượng trước khi gán sẽ được xóa.

Một hàm nạp chồng toán tử phải là một thành viên của class và có ít nhất một đối số do người dùng định nghĩa

Có 2 cách để truyền giá trị cho toán tử nạp chồng là truyền bằng trị và truyền bằng tham trị, với các đối tượng không thay đổi trong suốt quá trình thực hiện chương trình thì ta có thể thêm const.

|  |  |
| --- | --- |
| **Truyền bằng trị** | **Truyền tham trị** |
| operator +=( int a); | operator +=( int &a); |

VD: hàm truyền bằng tham trị

ostream& operator << (ostream & os, Phanso &ps){

os<< ps.tu<<"/"<<ps.mau<<endl;

return os;

}

VD: hàm truyền bằng trị

operator <(Phanso ps){

if((float)(tu/mau)<(float)(ps.tu/ps.mau))

return 1;

}

Hàm nạp chồng toán tử là thành viên của lớp, nhưng có những hàm nạp chồng toán tử cùng tên, giống nhau cả về tham số nhưng khác nhau về kết quả trả về. Vì vậy khi gọi tên hàm chương trình sẽ không biết phải sử dụng hàm nào. Do đó namespace được triển khai để giúp chương trình tìm được hàm cần sử dụng. Toán tử nạp chồng khi được gọi ra sẽ được chương trình tìm kiếm trong không gian tên của nó thông qua các tham số.

Thay vì gọi một hàm dài để thực hiện tính toán một bài toán vơi nhiều tham số, ta có thể thay đổi giá trị của tham số đầu tiên để việc gọi hàm trể nên dễ hơn. Bằng cách đó, ta sẽ phải sử dụng các toán tử như +=, -=, \*=, /=,…

VD:

class complex {

double re,im;

public:

complex & operator+=(comphex a); };

complex & operator+=(comphex a, complex b) {

return a+=b;

}

void ( comlex x, complex y, complex z)

{

complex r1{ x+y+z}; //r1=operator+(operator+(x,y),z)

complex r2={x}; //r2=x

complex r2+=y //r2=operator+=(r2,y)

complex r2+=z; //r2= operator +=(r2,z)

}

Ngoài các dạng số phức công số phức còn có dạng số phức cộng với số nguyên. Với a là số phức trường hợp này có dạng a+2 hoặc 2+a. Khi đó ta xây dựng chương trình như sau: