4.3. QUÁ TẢI CÁC TOÁN TỬ << , >>

Có thể quá tải các toán tử <<, >> **để xuất, nhập dữ liệu cho đối tượng.**

Hai hàm toán tử << và >> được khai báo là hàm bạn của lớp.

1. Toán tử << được quá tải có nguyên mẫu như sau:

```
friend ostream & operator
    <<(ostream & stream, ClassName & Object);
```

Hàm toán tử << trả về tham chiếu chỉ đến dòng xuất **ostream**. Tham số thứ nhất **`stream`** của hàm toán tử << là một tham chiếu chỉ đến dòng xuất `ostream`, tham số thứ hai **`Object`** là đối tượng được chèn vào dòng.

Khi sử dụng, dòng xuất đóng vai trò toán hạng bên trái và đối tượng xuất dữ liệu là toán hạng bên phải.

2. Toán tử >> được quá tải có nguyên mẫu như sau:

```
friend istream & operator
>>(istream & stream, ClassName & Object);
```

Hàm toán tử >> trả về tham chiếu chỉ đến dòng nhập istream.

Tham số thứ nhất **`stream`** của hàm toán tử >> là một tham chiếu chỉ đến dòng nhập `istream`, tham số thứ hai **`Object`** là đối tượng của lớp đang xét mà chúng ta muốn tạo dựng nhờ vào dữ liệu lấy từ dòng nhập.

Khi sử dụng, dòng nhập đóng vai toán hạng bên trái, đối tượng nhận dữ liệu là toán hạng bên phải.

Ví dụ 4.3. Chương trình sau minh họa việc quá tải hai toán tử << và >> trên lớp Diem.

```
//qua tai toan tu <<, >> de xuat nhap du lieu
#include <iostream>
using namespace std;
class Diem
{
  private:
              int x, y;
  public:
              Diem (int x = 0, int y = 0);  //Ham tao
              friend istream & operator >>(istream &Input, Diem &p);
              friend ostream & operator <<(ostream &Output,Diem &p);
};


Diem::Diem(int x1, int y1)
  { x = x1;
    y = y1;
  }
```

```
istream &  operator >>(istream &In,  Diem &p)
  {
    cout<<"Nhap x: "; In>>p.x;
    cout<<"Nhap y: "; In>>p.y;
    return In;
  }
ostream &  operator <<(ostream &Out,  Diem  &p)
  {
        Out<<"x = "<<p.x<<" ";
        Out<<"y = "<<p.y<<"\n";
        return Out;
  }
```

```
int main()
{
	Diem  d1, d2; //goi ham tao Diem va gan du lieu ban dau cua
d1, d2 là (0,0)
	cout<<d1;  //goi ham tai boi << de in ra x = 0 y = 0
	cout<<d2;  //goi ham tai boi << de in ra x = 0 y = 0
	//nhap du lieu moi cho d1 va d2
	cout<<"Nhap du lieu cho d1: \n";
	cin>>d1;  //goi ham tai boi >> de nhap du lieu cua d1
	cout<<"d1 :"<<d1; //goi ham tai boi << de in ra du lieu cua d1
	cout<<"Nhap du lieu cho d2: \n";
	cin>>d2;
	cout<<"d2 :"<<d2;
}
```

Thảo luận:

1. Hãy quá tải toán tử <<, >> để xuất nhập dữ liệu cho các đối tượng của lớp phân số.
2. Hãy quá tải toán tử <<, >> để xuất nhập dữ liệu cho các đối tượng của lớp mảng.

4.4. Quá tải một số toán tử đặc biệt

4.4.1. Quá tải các toán tử ++ , --

Ta có thể định nghĩa chồng cho các toán tử ++/-- theo quy định sau:

- Toán tử ++/-- dạng tiền tố trả về một đối tượng thuộc lớp.
- Toán tử ++/-- dạng hậu tố trả về một đối tượng thuộc lớp, khi định nghĩa ta thêm vào một tham số int.

Nhắc lại:

- Toán tử tăng/giảm ++/-- để tăng/giảm giá trị của biến 1 đơn vị.
- Việc dùng toán tử này trong các biểu thức ở dạng tiền tố (viết trước) hay hậu tố (viết sau) biến là có ý nghĩa khác nhau:

+) Ở dạng tiền tố thì giá trị của biến tăng/giảm 1 rồi mới sử dụng.

+) Ở dạng hậu tố thì sử dụng giá trị của biến rồi mới tang/giảm 1 cho biến.

Ví dụ: int a =3, b;

b = ++a; //a=4, b=4

//sau khi thuc hien thì b = 4, a=4

int x =5, y;

y = x++; //gán x=5 cho y rồi mới tang x lên 1

//kết quả khi thực hiện y = 5, x=6

Hai lệnh sau đều cho kết quả như nhau: đề tang i lên 1

++i;

Ví dụ 4.5 Quá tải các toán tử ++,-- trên lớp Diem.

```
#include <iostream>
using namespace std;
class Diem
  {
    private:
      int x,y;
    public:
      Diem() {x = y = 0;}
      Diem(int x1, int y1)
      {x = x1;
       y = y1;}
  Diem  operator ++(); //qua tai toan tu ++ tien to
  Diem operator ++(int); //qua tai toan tu ++ hau to
  Diem  operator --(); //qua tai toan tu -- tien to
  Diem operator --(int); //qua tai toan tu -- hau to
```

```
Diem  Diem::operator ++() //dang tien to
 {
   ++x;    ++y;
   return (*this);  //tra ve doi tuong this tro toi
  }

Diem Diem::operator ++(int)  //dang hau to
 {
   Diem temp = *this; //luu giu gia tri doi tuong this tro toi cho temp
   ++*this;  //tang gia tri doi tuong this tro toi len 1
   return temp;
  }
```

```
Diem  Diem::operator --()
 {
   x--;   y--;
   return (*this);
  }
Diem Diem::operator --(int)
 {
   Diem temp = *this;
   --*this;
   return temp;
   }
```

```
int main()
{
  Diem d1(2,3),d2;
  d2 = ++d1;  //tang 1 cho d1 roi gan cho d2
  d2.xuat(); // x = 3  y = 4
  d1.xuat(); // x = 3 y = 4

  Diem d3(5,7),d4;
  d4 = d3++;  // gan d3 cho d4 roi moi tang d3 len 1
  d4.xuat(); // x = 5  y = 7
  d3.xuat(); // x = 6  y = 8
  return 0;
}
```

Ví dụ: Chương trình sau minh họa quá tải toán tử ==, ++, += trên lớp Date.

```
#include <iostream>
using namespace std;
class Date
 {
   private:
      int D,M,Y;
   public:
            int nhuan();
            int ngaythang();
            void  nhap();
      void xuat();
      int operator ==(Date d2);
      Date operator ++();
      Date operator ++(int);
      Date operator +=(int);  //tang doi tuong ngay len k
ngay
};
```

```
int Date::operator ==(Date d2)
  { if (Y==d2.Y && M==d2.M && D ==d2.D)
      return 1;
     else
      return 0;
   //return (Y==d2.Y && M==d2.M && D ==d2.D);
  }

Date Date::operator ++()
 {
   int nday;
   nday = ngaythang();
   if (++D>nday)
    {
      D = 1;
      if (++M>12)
       { M = 1;
                  Y = Y + 1;
                }
    }
  return *this;
}
//Sv bo sung toan tu ++ dang hau to
```

```
Date Date::operator +=(int k)
{
  for (int i = 1; i <= k; i++)
    ++(*this);
  return *this;
}

int main()
{
  Date ob; int k;
  cout<<"\n Doi tuong ob = ";
  ob.nhap();
  ++ob;
  cout<<"\n ob = ob + 1 = ";
  ob.xuat();
  cout<<"\n Nhap so ngay can them vao : ";
  cin>>k;
  ob += k;
  cout<<"\n ob = ob + "<<k<< " = ";
  ob.xuat();
  return 0;
}
```

4.4.2 Quá tải toán tử [ ]

Khi cài đặt các lớp vector hoặc chuỗi ký tự, chúng ta cần phải truy cập đến từng phần tử của chúng, trong ngôn ngữ C/C++ đã có toán tử [ ] để truy cập đến một phần tử của mảng. Đây là toán tử hai ngôi, có dạng a[b] và khi quá tải toán tử này thì hàm toán tử tương ứng phải là thành phần của một lớp.

Ví dụ 4.7. Chương trình minh họa việc quá tải toán tử [ ] để truy cập đến thành phần của một đối tượng vector thuộc lớp Vector.

4.4.3. Quá tải toán tử ( )

Toán tử ( ) được dùng để gọi hàm, toán tử này gồm hai toán hạng: toán hạng đầu tiên là tên hàm, toán hạng thứ hai là danh sách các tham số của hàm.

Toán tử này có dạng giống như toán tử [ ] và khi quá tải toán tử này thì hàm toán tử tương ứng phải là thành phần của một lớp.

Ví dụ 4.8. Xét lớp Vector ở Ví dụ 4.7, quá tải toán tử ( ) để truy cập đến thành phần thứ i của một đối tượng thuộc lớp Vector:

Câu hỏi và Bài tập

Xem Chương 4 của giáo trình!