Programación Orientada a Objetos

Unidad 4: Sobrecarga de Operadores

Teoría 05 - 22/09/2011 - Pablo Novara

¿Qué es sobrecarga de operadores?

– C++ permite especificar el comportamiento de sus operadores standard al aplicarlos a tipos de datos no fundamentales (clases o estructuras).

```
int a,b,c;
a=b+c;
```

operadores + e = aplicados a un tipo fundamental (int)

```
class Complejo { … };
Complejo a,b,c;
a=b+c;
```

operadores + e = aplicados a una clase

Sobrecarga de operadores en C++

- Los operadores que se pueden sobrecargar son:

  - * / = < > += -= *= /=

  << >> <<= >>= == != <= >=

  ++ -- % & ^ ! | ~ &= ^=

  |= && || %= [] () new delete

- El único operador presente por defecto es el de asignación (que funciona de forma idéntica al constructor de copia por defecto)

Sobrecarga de operadores en C++

Hay dos formas de sobrecargar operadores:

- Como miembro de una clase (la del primer o único operando):

```
class Complejo { ...
    Complejo operator+ (Complejo c2);
... };
```

- Como función global:

```
Complejo operator+ (Complejo c1, Complejo c2);
```

Sobrecarga de operadores en C++

– Clase Complejo:

```
class Complejo {
    float preal, pimag;
public:
    Complejo():preal(0),pimag(0){}
    Complejo(float pr, float pi):preal(pr),pimag(pi){}
    float VerParteReal() { return preal; }
    float VerParteImag() { return pimag; }
    void CargarParteReal(float pr) { preal=pr; }
    void CargarParteImag(float pi) { pimag=pi; }
};
```

Sobrecarga de operadores en C++

– Clase Complejo:

```
int main() {
    Complejo c1(1,3), c2(4,5),suma;
    suma.CargarParteReal(
        c1.VerParteReal()+c2.VerParteReal());
    suma.CargarParteImag(
        c1.VerParteImag()+c2.VerParteImag());
    cout<<suma.VerParteReal()<<"+";
    cout<<suma.VerParteImag()<<"i";
    return 0;
}
```

Sobrecarga de operadores en C++

– Sobrecarga mediante función miembro:

```
class Complejo {
    float preal,pimag;
  public:
    …
    Complejo operator+ (Complejo c2) {
        Complejo suma(
                this->preal+c2.preal ,
                this->pimag+c2.pimag  );
        return suma;
    }
};
```

Sobrecarga de operadores en C++

– Sobrecarga mediante función miembro:

```
int main() {
    Complejo c1(1,3), c2(4,5),suma;
    suma = c1+c2;

    cout<<suma.VerParteReal()<<"+";
    cout<<suma.VerParteImag()<<"i";=c1+2c2
    return 0;
}
```

Es equivalente a:
suma=c1.operator+(c2);

Sobrecarga de operadores en C++

– Sobrecarga mediante función global:

```
class Complejo { … };

Complejo operator+ (Complejo c1, Complejo c2) {
        Complejo suma(
                c1.VerParteReal()+c2.VerParteReal() ,
                c1.VerParteImag()+c2.VerParteImag() );
        return suma;
    }
};
```

No se puede acceder
a los atributos privados,
a menos que se declare
amistad

Sobrecarga de operadores en C++

– Sobrecarga mediante función global:

```
int main() {
    Complejo c1(1,3), c2(4,5),suma;
    suma = c1+c2;

    cout<<suma.VerParteReal()<<"+";
    cout<<suma.VerParteImag()<<"i";
    return 0;
}
```

Es equivalente a:
suma=operator+(c1,c2);

¿Cuando utilizar función global y cuando miembro?

– Función miembro:

- siempre que se pueda (clases propias)

– Función global:

- clases ajenas o tipos fundamentales
- Ejemplo:

```
Complejo c1;
cout<<c1;
```

Es equivalente a:
cout.operator<<(c1);
ó
operator<<(cout,c1);

Sobrecarga de << y >>

- cout/cin son instancias de las clases ostream/istream
- Estas clases ya tienen sobrecargas para los tipos de datos fundamentales

```
ostream &operator<< (ostream &o,Complejo &c) {
    o<<c.VerParteReal()<<"+"
      <<c.VerParteImag()<<"i";
   return o;
}

istream &operator>> (istream &i,Complejo &c) {
    float a; i>>a; c.CargarParteReal(a);
    float b; i>>b; c.CargarParteReal(b);
    return i;
}
```

Uso del puntero this

– Los operadores que modifican al primer operando, suelen retornar al mismo objeto (por referencia) para permitir la aplicación en cadena:

Ejemplo:

```
class Complejo { …
    Complejo &operator=(Complejo &c) {
        this->preal=c.preal;
        this->pimag=c.pimag;
        return *this;
}

Complejo a,b,c,d(1,1);
a=b=c=d;
```

Es equivalente a:
a.operator=(b.operator=(c.operator=(d)));

Pre y Post Incremento

- Algunos operadores unarios pueden aplicarse de dos formas.
- Ejemplo: pre-incremento y post-incremento:

```
int c;
++c; // pre-incremento
c++; // post-incremento
```

- Para diferenciarlos se usa un argumento ficticio de tipo int:

```
class Complejo { …
    Complejo &operator++(); // pre-incremento
    Complejo operator++(int); // post-incremento
};
```

Sobrecarga de operadores y Cast

- Se puede sobrecargar la conversión a otros tipos de datos (cast)
- Ejemplo:

```
class Fraccion { … } ;

...

Fraccion f;
double d;

d=(double)f;

f=(Fraccion)d;

…
cout<<f;
```

a través de la sobrecarga:
class Fracción { …
operator double() {...}
…};

a través de un constructor
de Fracción que reciba un double
Equivale a: **f = Fracción(d);**

si no hay sobrecarga de <<
se realiza una conversión
implícita a double

Sobrecarga de operadores y Cast

```
class Fraccion {
    int num,den;
public: ...
    Fraccion(double d) {
        den=1;
        while (d!=int(d)) { den*=10; d*=10; }
        num=int(d);
    }
    operator double() {
        return double(num)/den;
    }
    …
};
```

para convertir un double en una Fraccion

para convertir una Fraccion en un double