Programación 2009

Tarea 12

Las preguntas precedidas por un asterísco son para los alumnos de maestría. En licenciatura, dan puntos extras.

1 Operadores

Ejercicio 1 *Determinar con un ejemplo sencillo cual es el valor entero pasado como segundo operando en el caso de la versión post-fija de los operadores ++ y −−.*

Ejercicio 2 *Tomar la clase Complex de las tareas precedente y escribirle los operadores +, -, *, / y =.*

***Ejercicio 3** Siguiendo con el ejemplo (workStruct/ImageProc) de la clase, añadir a la clase ImageProc:*

1. *un método independent() que permita a los objetos instanciados "independizarse" de los datos compartidos : se hará un clone de la estructura workStruct, si se necesita hacerlo;*
2. *un método copy() que servirá a la vez para el constructor por copia y el operador de asignación;*
3. *unos operadores binarios == y != que verifican si los dos operandos comparten o no el mismo workStruct.*

Ilustrar el uso de esos métodos con un ejemplo.

Ejercicio 4 *¿Qué se puede añadir al código siguiente para que compile ?*

```
class Scale2D {
  int sx,sy;
public:
  Scale2D(int cx=1,int cy=1) {sx=cx;sy=cy;}
};
class Image {
  int width;
  int height;
public:
  Image(int w=0,int h=0) : width(w),height(h) {}
```

```
};
int main() {
  Image img1(200,200),img2(240,250);
  img1 *= Scale2D(3,3);
  img2 /= Scale2D(2,4);
  Image img3 = img1 * Scale2D(10,10);
  if (img3<img2) {
        img3 *= 2;
  }
}
```

Ejercicio 5 *Usando la clase Complex de las tareas precedentes, implementar una clase ComplexPtr, que manejará, internamente, un arreglo dinámico hacia Complex y servirá de "Smart Pointer". Se implementarán en el ComplexPtr :*

- constructores adecuados,
- operadores de incrementación y decrementación (prefija y postfija), operadores de dereferencia *, operadores $->$ y []

```
class Complex {
  ...
};
class ComplexPtr {
  Complex *arreglo;
  int size;
  int index;
  ...
};
```

Un ejemplo será :

```
int main() {
  int N=100;
  ComplexPtr a(N);
  ComplexPtr c(1);
  c = a;
  for (int j=0;j<N;j++) c[j] = Complex(j,j);
  do {
    c->print();
  } while (c++);
  do {
    c->print();
  } while (c--);
}
```