Programación 2009

Tarea 8

***Ejercicio 1** *Escribir un programa que haga alocación de un arreglo de 20 apuntadores a funciones, de valor de regreso enteros y de argumentos tres dobles. Rellenarlo con verdaderos funciones de ese tipo, hacer una llamada a la función del elemento 5 del arreglo con los argumentos de su elección y finalmente liberar el arreglo.*

Ejercicio 2 *Explicar el problema (si hay...) del código siguiente :*

```
struct toto {
int i;
};
...
int *arreglo = new toto(10); // Allocate array of 10 totos
for (int k=0;k<10;k++)
   toto[k]->i = 1;
...
delete [] arreglo;
```

Ejercicio 3 *Reescribir el (muy feo) programa siguiente usando cada vez que lo pienses útil un* ***cast explícito a la C++****.*

```
#include <cmath>
#include <iostream>
using namespace std;
struct myStruct {
  double a,b;
  float  *data;
  int doSomething(const int &x) {
    data = (float*)x;
    double b = pow(x,a) + pow(x,b);
    return b;
  }
  void setData(int *const someData) {
    data = (float *) someData;
  }
};
int main() {
```

```
  int a[10];
  myStruct m;
  m.setData(&a[0]);
  return 0;
}
```

Ejercicio 4 *Siguiendo el ejemplo de la clase (Image), escribir una estructura dedicada a Matrices de double. Esa estructura (**sin constructores**) tendrá métodos para:*

- *inicializarse a partir del contenido de un archivo externo (formato especificado abajo),*
- *imprimir la matriz en la salida estándar,*
- *liberar el espacio memoria alocado para la matriz,*
- *añadir otra matriz (que tiene que ser de misma dimensión) a la matriz de que se llama el método, guardando el resultado en la Matriz que llama.*

Los archivos de matriz contendrán una primera linea con el número de lineas y el número de columnas, luego vendrán los datos, linea por linea. Por ejemplo,

```
3 2
1.0 -9.1
11.0 27.9
-4.9 78.0
```

***Ejercicio 5** *Inspirándose del ejemplo de la clase (estructura imagen), proponer e implementar una estructura "a la C++" para almacenar números complejos (a partir de dobles), y para soportar las operaciones siguientes sobre complejos, a través métodos :*

```
void print(); // Imprime el complejo en la salida estándar
const Complex &copy(const Complex &c); // Copy desde otro complejo
const Complex &add(const Complex &c);  // Adición de otro complejo a este
const Complex &mul(Complex &c);   // Multiplicación de otro complejo a este
const Complex &mul(double &d);    // Multiplicación por un escalar
Complex conjugado();              // Complejo conjugado
const double &real();             // Real part
const double &imaginary();        // Imaginary part
double modulo();                  // Modulo
double fase();                    // Fase
```

*Como se ve, unas de esas funciones regresan **referencias** a la estructura Complejo que las llamó. Se crearán dos archivos Complejo.h y Complejo.cpp, el primero contendrá las **declaraciones** relacionadas a la estructura Complex, el segundo las **implementaciones de cada método**.*