Guion de prácticas

Práctica Final

Mayo de 2016

DECSAI

Metodología de la Programación

Curso 2015/2016

Índice

1. Definición del problema

En el guion anterior se utilizó un array dinámico para representar internamente la clase *Imagen*, evitando los problemas de desperdicio o falta de memoria producido por el uso de arrays automáticos.
Si bien la clase se puede usar, aún es necesario incorporar un conjunto de métodos para considerarla completa.

En esta práctica se completará la funcionalidad de la clase incorporando el constructor de copia, el destructor y la sobrecarga del operador de asignación. Posteriormente, se repetirá el ejercicio, cambiando la representación interna de la Imagen a una matriz dinámica. Finalmente, se sobrecargará el operador '+' en cada una de las representaciones anteriores de la clase *Imagen*.

También, se completará la clase *Lista* creada en la práctica anterior implementando el destructor, el constructor de copia y la sobrecarga del operador de asignación. Finalmente, se sobrecargará el operador '+' en la clase *Lista* .

En todos los casos, se deberá considerar si es necesario reimplementar los posibles métodos afectados dentro de cada una de las clases.

Regla de tres

Cuando nos veamos obligados a definir alguno de los siguientes métodos:

- destructor,
- constructor de copia,
- operador de asignación,

probablemente tengamos que definir los tres. Si no los definimos, el compilador generará estos métodos por defecto. Si alguno de estos métodos por defecto, no cumple adecuadamente su función, probablemente tampoco lo hagan el resto.

2. Objetivos

El desarrollo de esta práctica pretende servir a los siguientes objetivos:

- Continuar trabajando con memoria dinámica.
- Completar la implementaciones del las clases *Imagen* y *Lista* utilizadas en la práctica 5.
- Implementar la clase *Imagen* con una nueva representación interna, dándose cuenta de que, al no cambiar el interfaz, no es necesario modificar los programas o funciones que usan la clase.
- Realizar programas que lean parámetros desde la línea de órdenes.

3. Tareas a realizar

A partir del trabajo especificado en la práctica 5 se deberán realizar las siguientes tareas.

3.1. Completar y extender la clase *Lista*

1. Completar clase *Lista* creada en la práctica anterior implementando el destructor, el constructor de copia y la sobrecarga del operador de asignación.
2. Extender la clase sobrecargando el operador '+' para poder realizar operaciones de concatenación de listas ($L = L + S$, siendo L una *Lista* y S un *String*).

3.2. Completar la clase *Imagen*

1. Cree una carpeta denominada `imagen`. Luego, dentro de la carpeta `imagen`, cree otra carpeta denominada `vector` y mueva todos los ficheros que tenga actualmente de la práctica 5, junto con la actual implementación de la clase *Lista*, dentro de esta nueva carpeta.
2. Complete la clase *Imagen* que utiliza un array dinámico desarrollada en la práctica 5, implementando el destructor, el constructor de copia y la sobrecarga del operador de asignación. Repase los apuntes de teoría. Considere si cambia la implementación de otros métodos cuando la clase está completa.
3. Pruebe la nueva implementación sobre los programas *testimagen* y *arteASCII2* de la práctica 5. Para el segundo de ellos, cambie su interfaz de modo que reciba los parámetros desde la línea de órdenes:

   ```
   ./arteASCII2 imagen.pgm grises.txt
   ```

   Recuerde asimismo eliminar la llamada al método *destruir* dentro de cada *main*. Por último, utilice el programa `Valgrind` para comprobar el uso correcto de la memoria dinámica.
4. Se pretende agregar a la clase, una funcionalidad que permita "concatenar" imágenes.

   Supongamos que B y C son imágenes de tamaño 100×80 y 256×256, respectivamente. La concatenación de B con C será una nueva imagen de tamaño 256×336 obtenida "pegando" la imagen C a la derecha de la imagen B, quedando la parte no cubierta por ninguna imagen en negro. Tome como referencia el siguiente ejemplo:

   Para ello sobrecargará el operador '+' para poder realizar operaciones como $A = B + C$. Es importante notar que para que funcione correctamente, debe tener antes implementada la sobrecarga del operador de asignación.

B	B	B
B	B	B
B	B	B

Cuadro 1: Imagen B

C	C	C	C
C	C	C	C

Cuadro 2: Imagen C

B	B	B	C	C	C	C
B	B	B	C	C	C	C
B	B	B				

Cuadro 3: Imagen A = B + C.

5. Crear un nuevo programa *suma.cpp* para probar esta nueva funcionalidad de la clase. *Suma* recibirá desde la línea de órdenes los siguientes parámetros: el nombre de la primera y segunda imagen, el nombre de la imagen de salida y un flag para indicar si la salida se guarda en texto (t) o binario (b). Por ejemplo, para indicar que la concatenación de dos imágenes debe ser guardada en un fichero de texto se ejecutará el siguiente programa:

```
./suma img1.pgm img2.pgm img_out.pgm t
```

3.3. Modificar la representación de la clase *Imagen*

1. Crear la carpeta `matriz` dentro de la carpeta `imagen` previamente creada. Copie el contenido de la carpeta `vector` dentro de esta nueva carpeta, incluyendo todos los ficheros con la implementación completa de dicha clase.

2. Sobre la copia, reimplemente la clase *Imagen* utilizando como estructura interna una matriz dinámica. Utilice la representación que mantiene una estructura de punteros al comienzo de las filas con todas las filas almacenadas como una zona continua de memoria. Dicha estructura se muestra en la Fig. 1.

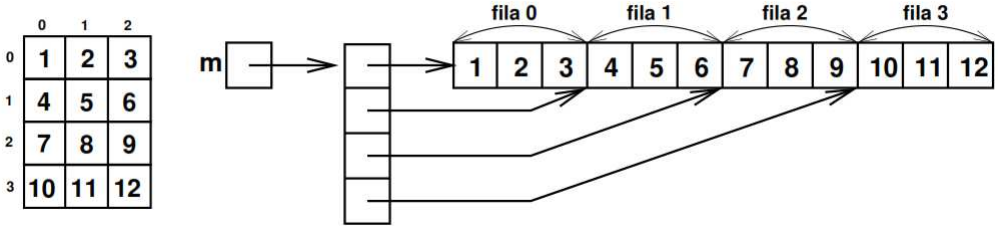

Figura 1: Estructura de la matriz.

Tal como ocurre en la implementación del array dinámico, el constructor de la clase debe reservar memoria y el destructor, liberarla. Adapte las implementaciones del constructor, constructor de copia, destructor, y la sobrecarga del operador de asignación. Repase los

apuntes de teoría para implementar la reserva/liberacion de memoria asociada a la representacion utilizada de la matriz. Tenga especial cuidado con los siguientes aspectos:

- Reimplementar los métodos de acceso/consulta a la **Imagen** (set/get, setPos/getPos).
- Reimplementar las funciones de lectura/escritura de la **Imagen** a ficheros. En esta práctica se considerará la lectura y la escritura de imágenes tanto de texto como binarias.

3. Pruebe esta nueva implementación sobre la nueva versión de los programas *testimagen*, *arteASCII2* y *suma* desarrollados en la tarea anterior. Por último, utilice el programa `Valgrind` para comprobar el uso correcto de la memoria dinámica.

4. Material a entregar

Cuando esté todo listo y probado el alumno empaquetará la estructura de directorios en un archivo con el nombre **practicafinal.zip** y lo entregará en la plataforma **decsai** en el plazo indicado. No deben entregarse archivos objeto (**.o**) ni ejecutables. Para asegurarse de esto último conviene ejecutar **make mrproper** antes de proceder al empaquetado. Deberá asegurarse de que ejecutando las siguientes órdenes se compila y ejecuta correctamente su proyecto:

```
unzip practicafinal
cd practicafinal/imagen/vector
make
bin/testimagen
bin/arteASCII2
bin/suma

cd ../matriz
make
bin/testimagen
bin/arteASCII2
bin/suma
```

Se debe hacer uso de la herramienta valgrind para asegurarse de que no se producen los errores en el uso de la memoria dinámica. Además, recuerde que haciendo **make documentacion** debe poder generarse la documentación doxygen del software.

Finalmente, el alumno deberá escribir un informe donde consten los nombres y DNI de los integrantes del grupo, los problemas que hayan podido surgir durante el desarrollo de la práctica, capturas de pantalla mostrando ejemplos del uso, etc. Este informe, en formato pdf, se guardará en las carpetas doc dentro de las carpetas `vector` y `matriz`.