Ejemplo de diseño modular en C++

Presentamos un ejemplo de diseño modular completo que hace uso de las clases `stack` y `list`. Recordemos que la metodología del diseño modular que usaremos en la asignatura consta de cinco fases:

1. Detectar las clases de datos implicadas en nuestro programa, a partir del enunciado
2. Obtener un esquema preliminar para el programa principal
3. Especificar las clases anteriormente detectadas
4. Escribir el programa principal detallado, usando objetos e instrucciones reales
5. Implementar las clases

Las fases 3 y 5 pueden repetirse durante la implementación, si aparecen nuevas clases auxiliares que no eran evidentes al principio del diseño. También puede ser necesario realizar algunos ajustes en la ubicación o en la especificación de las operaciones, si las decisiones originales no son del todo correctas.

En el ejemplo que vamos a ver, os proporcionamos una parte del diseño para que lo completéis en varias fases. Al final de cada fase, diseñaréis juegos de pruebas y probaréis el programa resultante en situaciones especiales.

Por otra parte, presentaremos la herramienta de documentación `doxygen`. La hemos usado para generar las especificaciones de las clases del ejemplo. En documento aparte, os mostraremos una pequeña introducción a su uso.

7.1. Proyecto: Gestión de una lavadora

Consideremos una lavadora y una cubeta de ropa para lavar. Su operativa normal permite depositar una prenda de ropa tanto en la lavadora como en la cubeta. También existe la posibilidad de completar la lavadora con ropa de la cubeta.

Una lavadora puede estar inicializada o no. Todas las operaciones, salvo la inicialización, se pueden aplicar solamente sobre lavadoras inicializadas. Los datos relevantes para inicializar una lavadora son el peso máximo que se pretende cargar en ella y si va a ser de ropa blanca o de color. Las prendas de ropa también tienen como atributos su peso y su color. Todos los pesos

serán números naturales y la información del color puede representarse con un booleano (por ejemplo, el blanco mediante el valor `false` y el color con el valor `true`).

Cuando se desea completar una lavadora, se extrae de la cubeta la mayor cantidad posible de ropa del tipo correspondiente (blanco o color) que no se pase del peso máximo de la lavadora y sacando primero las prendas de ropa introducidas en último lugar.

Por último, se dispone de una operación que simula el lavado de las prendas que se encuentren en la lavadora en un momento dado. Se podrá aplicar incluso si la lavadora no está llena. Su resultado es que la lavadora queda no operativa y lista para inicializarse con nuevos datos.

El programa principal creará (o, mejor dicho, instalará) la cubeta y la lavadora y se encargará de aplicar las operaciones descritas, ofreciendo un menú de opciones:

```
Esquema de programa principal:

  instalar lavadora
  instalar cubeta
  leer opción
  while (opcion != final) {
    leer datos opción
    aplicar opción
    leer opción
  }
```

Según lo expuesto anteriormente, habrá cinco opciones básicas: inicializar una lavadora, depositar una prenda en la lavadora, depositar una prenda en la cubeta, completar la lavadora y realizar un lavado. Además, durante el proceso se podrán aplicar operaciones de escritura del contenido de la lavadora o la cubeta para supervisar el funcionamiento del programa.

7.2. Ejercicio: programa principal

Implementad el programa principal suponiendo que están disponibles las clases `Lavadora`, `Cubeta` y `Prenda`. Los ficheros `.hh` están en la carpeta de esta sesión (incluyen comentarios especiales para que `doxygen` pueda generar la documentación, ver sección 7.4). Los ficheros `.o` se encuentran en la carpeta `OBJECTES`[1]. La documentación de las clases se encuentra en formato `doxygen` en la subcarpeta `DOC`. Utilizad como plantilla el fichero `pro2_especif.cc` que viene preparado para integrarse en la documentación. Utilizad el fichero `Makefile` (que incluye su propia documentación) para compilar y enlazar este programa.

Probadlo con las entradas que queráis, incluyendo el fichero `datos.txt`, a partir del cual tendréis que deducir la manera de organizar la lectura de los datos. Los resultados correctos para éste se encuentran en `salida.txt`, a partir del cual tendréis que deducir la manera de organizar las escrituras. Los elementos de formato de salida que no se desprendan de las operaciones de

[1] Si estáis haciendo la sesión en un sistema que genera ficheros objeto no compatibles con los del Linux de la FIB, probad el programa principal *después* de implementar las clases, tal como se plantea en un ejercicio posterior

escritura de las clases han de incorporarse en el programa principal. Respecto a la entrada, se han tomado los siguientes valores negativos como códigos de las diversas operaciones:

- -1: inicializar lavadora (datos: peso máximo y color)
- -2: añadir una prenda a la lavadora (datos: peso y color de la prenda)
- -3: añadir una prenda a la cubeta (datos: peso y color de la prenda)
- -4: completar la lavadora
- -5: realizar un lavado
- -6: escribir el contenido de la cubeta
- -7: escribir el contenido de la lavadora
- -8: fin del proceso

Por último, podéis suponer que los datos de la entrada son correctos. Si no lo fueran, habría que aplicar las protecciones correspondientes a las operaciones antes de utilizarlas, para garantizar que se cumplen sus precondiciones.

7.3. Ejercicio: juegos de pruebas especiales

Escribid y probad una serie de ficheros de datos que exploren diversas situaciones límite de la operación `completar_lavadora`, por ejemplo:

- que no se pueda sacar ninguna prenda de la cubeta, aunque en ella haya prendas del color correspondiente, porque la primera prenda posible hace que se pase del peso máximo (la lavadora no se modifica)
- que se saquen de la cubeta todas las prendas del color correspondiente
- que la lavadora quede llena (se alcance el peso máximo exacto)
- que la primera prenda que no se pueda sacar de la cubeta haga que se alcance el peso máximo más 1.

Este tipo de pruebas, basadas solamente en las especificaciones de un programa, se denominan pruebas de *caja negra* (en inglés “black box testing”).

7.4. Ejercicio: Documentación del programa principal

Leed el contenido del fichero `doxygen.pdf` y realizad la primera actividad propuesta “Añadir un programa principal a la documentación de un proyecto”.

7.5. Ejercicio: Implementación de clases. Ejercicio del *Jutge* X64677 de la lista *Pràctica*

Implementad vuestra propia versión de la clase `Cubeta` de modo que el programa principal siga funcionando si en vez de enlazarlo con el `Cubeta.o` de `OBJECTES` lo enlazáis con vuestra versión. Usad el fichero `Cubeta.hh`, añadidle el invariante de la representación de la clase y generad vuestro propio `Cubeta.cc`. Eso obligará a modificar el fichero `Makefile` para que siga siendo útil en la nueva situación. Se han de añadir una regla de compilación para el nuevo fichero `.cc` y se ha de revisar la regla de enlazado.

Para la operación `completar_lavadora` tenemos una operación auxiliar `completar_lavadora_pila` que extrae las prendas de una pila de prendas y las pase a la lavadora. La operación original simplemente elegirá sobre qué pila ha de aplicarse la auxiliar, en función del color de la lavadora. Proponemos dos versiones de dicha auxiliar, una recursiva y otra iterativa, ambas definidas como `static`. Implementad y probad las dos versiones.

Aplicamos una descomposición similar para la operación `escribir`, que requiere una operación auxiliar `escribir_pila_prenda`, también `static`. Es este caso, implementad solo la versión iterativa.

Realizad las pruebas con los mismos juegos empleados para probar el principal y después con otros nuevos, que tendréis que crear para probar situaciones especiales de la implementación que aún no hayan sido probadas. Este otro tipo de pruebas, que ya tienen en cuenta la implementación de un programa, se denominan pruebas de *caja blanca* (en inglés “white box testing”).

Repetid el proceso con la implementación de las clases `Lavadora` y `Prenda`. El fichero `Makefile` también se tendrá que modificar.

7.6. Un ejercicio alternativo

Modificad la operación `completar_lavadora` para que no pare de pasar ropa de la cubeta a la lavadora cuando encuentre una prenda que no quepa en ésta, sino que siga probando las prendas situadas por debajo de dicha prenda, pasando a la lavadora cada prenda encontrada que quepa en ella. Las prendas que queden en la cubeta han de mantener el orden en el que estaban.

Ejemplo: si en la lavadora faltan por llenar 7 unidades de peso de ropa blanca y en la cubeta tenemos las siguientes prendas blancas

```
1
2
2
4
1
3
2
```

al terminar han de quedar en la cubeta sólo las prendas de pesos

```
4
3
2
```

Diseñad una nueva serie de juegos de pruebas, con criterios similares a los de la anterior, para esta versión alternativa.

7.7. Ejercicio: Documentación de un clase

Teniendo en cuenta la modificación realizada en la operación `completar_lavadora`, realizad la segunda actividad propuesta en el fichero `doxygen.pdf` “Actualizar la documentación de una clase”.