5.2 Formato de ficheros

Anteriormente se ha explicado con detalle cómo mediante listas enlazadas se almacena en memoria toda la información que usa la aplicación. Generalmente un usuario de la aplicación, cuando tenga que realizar alguna operación con la misma, no deseará introducir de nuevo toda la información. Normalmente, un usuario que introduce cierta información en la aplicación o realiza ciertos cambios quiere tener la posibilidad de hacer que esos cambios sean permanentes. Sin embargo toda la información se almacena en memoria, de manera que los datos dejarán de estar disponibles tan pronto como el usuario cierre la sesión de trabajo.

Es por tanto completamente necesario implementar un sistema que permita salvar toda la información necesaria a ficheros de disco para que pueda ser recuperada en cualquier instante. De esta forma se permite además que un usuario almacene en disco la información correspondiente a varias redes viales distintas y que en cada sesión de trabajo pueda seleccionar con cuál de ellas quiere trabajar.

Parece obvio que se debe implementar algún sistema mediante el cual se pueda almacenar ordenadamente en disco toda la información que contiene cada una de las listas. Aunque realmente no se debe almacenar toda la información de cada una de las listas, sino sólo la información necesaria para que el sistema sea capaz de reconstruir cada una de ellas. Es inútil almacenar, por ejemplo, la dirección en memoria de cada uno de los nodos de la lista, porque cuando ésta se reconstruya se deberá solicitar de nuevo memoria dinámica al sistema y es absolutamente improbable que asigne de nuevo las mismas direcciones.

Una vez planteado el problema se va a exponer la solución implementada en la aplicación. En primer lugar, hay que tener en cuenta que no todas las listas necesitan ser salvadas, puesto que algunas de ellas no contienen información transcendente y sólo tienen como fin facilitar la tarea de la programación o el funcionamiento de la aplicación, aunque este tipo de listas no son abundantes. Por lo tanto el primer paso debe ser la selección de las listas que deben ser salvadas a disco y las que no. Por ejemplo, la lista *Lista_tramos_nodo* no necesita ser salvada a disco, pues se crea a partir de los datos de la listas *Lista_tramos* y *Lista_nodos*.

En segundo lugar hay que plantearse la distribución de los ficheros de disco, es decir, cuántos ficheros se van a dedicar para cada red vial o escenario, o para cada lista, etcétera. La forma más cómoda sería almacenar toda la información de un escenario, es decir, cada una de las listas, en un único fichero. Este método tiene la ventaja de que con sólo cargar un fichero se tiene toda la información necesaria para reconstruir un escenario completo. Esto es ventajoso sobre todo si se tienen varios escenarios y se quiere tener localizada la información correspondiente a cada uno de ellos.

Pero esta opción también tiene graves problemas. El primero de ellos es que para escenarios grandes el fichero puede tener un tamaño considerable que puede provocar una ralentización en exceso del acceso al mismo. El segundo es la complejidad que supone. Habría que indicar al sistema cuando acaba la información de una lista y cuando comienza otra de manera que esta “señalización” sea tolerante a errores, es decir, que un fallo en la lectura de una las listas o nodos no implique que no se pueda

acceder al resto de la información. En caso contrario, cualquier error en dicho fichero implicaría la pérdida de toda la información de la base de datos.

Se opta por usar un sistema diametralmente opuesto al anterior, que consiste en almacenar cada una de las listas en un fichero separado. Esto elimina el segundo inconveniente, ya que los ficheros son de menor tamaño y mucho más manejables y cada uno de ellos sólo va a contener la información de una lista. La principal ventaja que ofrecía el sistema anterior, que consistía en la centralización de la información, puede conseguirse con ayuda del usuario: éste sólo tiene que almacenar todos los ficheros de cada uno de los escenarios en directorios separados. De esta manera se elimina el principal inconveniente del método anterior y se pueden obtener las mismas ventajas.

Una vez decidida la información que se va a almacenar en cada fichero hay que definir la forma en que se va almacenar dicha información y el formato del fichero, ya que en los sistemas DOS y Windows los ficheros se pueden abrir tanto en modo texto como binario.

La forma de almacenamiento de la información que se ha elegido es la siguiente: en primer lugar se escribe en el fichero el número de nodos que contiene la lista y a continuación se escribe la información de cada uno de los nodos. En este caso es mucho más útil abrir el fichero en modo binario, ya que de esta manera se puede escribir directamente la información de cada uno de los nodos, sin tener que convertir previamente dicha información a texto. Es decir, se puede escribir en el fichero cada uno de los nodos que forman la lista. Además cuando se almacena cierta cantidad de información en un fichero suele ocupar menos espacio en formato binario que en texto.

Y hay otro detalle mucho más importante: como se verá, las funciones que se encargan de salvar la información a disco se han implementado como métodos de la clase lista genérica *Lista_inf*. Si el archivo se hubiese abierto en modo texto, habría que escribir cada uno de los campos de cada uno de los nodos como texto en el archivo, de manera que la función ya no podría ser genérica sino que tendría que haberse codificado una función para cada una de las listas, ya que los objetos nodos de cada una de ellas serían instancias de clases distintas con datos miembro distintos.

Esto se verá más en detalle cuando a continuación se presente el código encargado de implementar el almacenamiento y la posterior reconstrucción.

Una vez comprendido el proceso de almacenamiento, es sencillo entender cómo la aplicación recuperará los datos de disco. Para cada una de las listas la aplicación abrirá el fichero adecuado, que debe ser indicado por el usuario.

Una vez abierto el fichero, la aplicación creará una lista cuyos nodos sean instancias de la clase adecuada usando la plantilla *Lista_inf*. En este primer momento la lista debe estar vacía, es decir, no debe tener ningún nodo enlazado.

A continuación lee del fichero el número de nodos que contendrá la lista. Este dato lo usará para implementar un bucle que realizará las siguientes operaciones en cada una de sus iteraciones:

- Leer el nodo del fichero, que está almacenado tal cual.

- Crear un nodo idéntico al leído del fichero y enlazarlo a la lista.

De esta manera se reconstruye la lista enlazada.

Implementación en C++

Como se dijo anteriormente, a continuación se expone el código de una de las funciones que salva datos a disco y una de las que carga datos desde disco.

Estas funciones son métodos propios de la clase lista genérica *Lista_inf*, por lo que podrán ser usadas por todas las listas que se hayan creado usando dicha plantilla.

Se expone en primer lugar el código de *Guardar_Información*:

```
template <class Y> void Lista_inf<Y>::Grabar_Informacion(char *nombre)
{
 FILE *F1;
 unsigned int numero_elementos,i;
 Y *inf;
 F1=fopen (nombre,"wb");
 numero_elementos=Numero_elementos();
 inf=(Y*)malloc (sizeof(Y)*numero_elementos);
 if (F1!=NULL)
  {
        fwrite (&numero_elementos,sizeof(unsigned int),1,F1);
        for (i=0;i<numero_elementos;i++)
             inf[i]=Elemento(i);
        // A continuacion pasamos a salvar.
        fwrite (inf,sizeof(Y),numero_elementos,F1);
        free (inf);
  }
fclose (F1);
}
```

La función recibe como parámetro el nombre del fichero donde se va a guardar la información. En primer lugar se definen las variables a usar: un manejador de fichero (*F1*), un par de enteros (concretamente *unsigned int*) y un puntero a un nodo de la lista. Obsérvese cómo es genérico, este código es válido para todas las listas creadas con la plantilla *Lista_inf*, independientemente de la clase a la que pertenezcan cada uno de los nodos.

En primer lugar se abre el fichero adecuado en modo binario y para escritura (“*wb*”). A continuación se reserva memoria para una tabla de objetos de la misma clase que los nodos de la lista y con el mismo número de elementos, porque la información de la lista se almacenará en una tabla. El puntero *inf* contiene ahora la dirección de comienzo de dicha tabla.

A continuación se escribe en el fichero el número de nodos que tiene la lista. Seguidamente se copia cada uno de los nodos de la lista en un elemento de la tabla, con lo que se tiene almacenada la lista en forma de tabla.

Finalmente se escribe la tabla en el fichero y se libera la tabla de la memoria.

La otra función existente para salvar la información a disco es *Grabar_Informacion_cont*. Esta función es exactamente igual a la que se acaba de exponer, la única diferencia es que recibe como parámetro un manejador de fichero en lugar del nombre del fichero que se quiere abrir, es decir, que se debe abrir el fichero antes de llamar a la función y cerrarlo tras regresar.

Una nota importante es que si se hubiese abierto el archivo en modo texto, se tendría que haber escrito cada uno de los campos del objeto nodo en dicho fichero. Al tener distintas listas, los nodos de cada lista tendrán distintos datos miembro. Esto implica que habría que codificar una función distinta para salvar la información a disco para cada una de las listas, y lo mismo ocurriría con la función para cargar información, mientras que abriendo el fichero en binario se ha podido codificar una sola función genérica para salvar y otra para cargar.

A continuación se muestra la función para cargar información desde disco:

```
template <class Y> void Lista_inf<Y>::Cargar_Informacion(char *nombre)
{
 FILE *F1;
 unsigned int numero_elementos,i;
 Y *inf;
 F1=fopen (nombre,"rb");
 if (F1!=NULL)
  {
        fread (&numero_elementos,sizeof(unsigned int),1,F1);
        inf=(Y*)malloc (sizeof(Y)*numero_elementos);
        fread (inf,sizeof(Y),numero_elementos,F1);
        for (i=0;i<numero_elementos;i++)
            Add_elemento(inf[i]);
        free (inf);
  }
fclose (F1);
}
```

Puede observarse como la función es muy similar a la anterior, la única diferencia es el modo de apertura del archivo (“*rb*”, binario para lectura) y de donde se leen los datos. En este caso se crea de nuevo en memoria la tabla donde se almacenan todos los nodos pero la información se lee del fichero y a continuación se añade a la lista cada uno de los nodos de la tabla. La lista debía haber sido creada con anterioridad.

Casos especiales: las listas de nodos y tramos

Se acaba de presentar el método usado para salvar la información a disco y cargarla desde él. Sin embargo no se ha tratado un aspecto muy importante: ¿cómo se introducen

los datos por primera vez?. En efecto, si es la primera vez que se ejecuta la aplicación, o se desea crear un nuevo escenario, no se va a disponer de un archivo en disco del que cargar la información. Así que deben existir métodos para “crear” los datos.

Los métodos habilitados para “crear” los datos dependen del tipo de datos a crear. Hay interfaces que permiten crear datos por primera vez, como es el caso del interfaz de las líneas de bus.

Este interfaz crea la lista con los tramos que recorre cada línea de bus al introducir un número de línea distinto a los ya existentes, y va añadiendo cada uno de los tramos a la lista según se vayan seleccionando los nodos sobre el mapa de la red vial. Así que no habría que preocuparse por estas listas.

Sin embargo, hay otras listas para las que puede resultar más sencillo habilitar una forma alternativa, más rápida de cargar los datos y luego modificar o completar éstos usando el interfaz adecuado.

Este es el caso de las listas de tramos y nodos. Los tramos y nodos son los componentes básicos de la red vial. Cada nodo representa un cruce de la red vial y cada tramo representa cada uno de los sentidos de una vía que une dos nodos. Es decir, una calle con circulación en doble sentido se representaría por dos tramos, uno para cada sentido.

Antes de empezar a trabajar con la aplicación ésta debe cargar una red vial. La información básica de la red vial la constituyen los tramos y los nodos. Por lo tanto, antes de empezar a trabajar con la aplicación ésta debe haber reconstruido las listas de tramos y nodos. Dada la gran cantidad de nodos y tramos que pueden formar la red vial de una ciudad, es deseable disponer de un método alternativo que permita cargar estos datos de forma sencilla.

Antes de exponer el método hay que matizar que hay información sobre los nodos y tramos que no es necesaria para construir la red vial, y dicha información no necesita ser cargada por primera vez en la aplicación para su correcto funcionamiento. La información básica necesaria para cada nodo es:

- Código del nodo: es un número único que identifica unívocamente al nodo. Es la clave primaria para la lista de nodos.
- Posición del nodo: posición del nodo en coordenadas absolutas o globales. Será, por tanto, una pareja de números, una coordenada para el eje x y otra para el eje y.

Toda la información relativa al nodo que no ha aparezca en el listado anterior se considera no básica, y podrá actualizarse o modificarse posteriormente con el interfaz adecuado para ello, pero no es necesario que sea cargada inicialmente.

La información básica necesaria para el tramo es:

- Código del tramo: es un número único que identifica unívocamente al tramo. Es la clave primaria para la lista de tramos.
- Nodos origen y destino del tramo: cada tramo tiene un nodo origen y destino. En realidad, por definición, no puede haber dos tramos que tengan el mismo nodo origen y destino, pero es más cómodo usar un código para diferenciar unos de otros.

Análogamente, toda la información relativa al tramo que no ha aparezca en el listado anterior se considera no básica, y podrá actualizarse o modificarse posteriormente con el interfaz adecuado para ello, pero no es necesario que sea cargada inicialmente.

El método que se ha habilitado para cargar por primera vez un escenario o red vial consiste en que se puede almacenar la información básica necesaria para los nodos y los tramos en sendos ficheros de texto. Estos ficheros de texto tienen un formato muy sencillo, por lo que pueden ser creados fácilmente por el usuario o quizá exportados de otras aplicaciones.

Como ya se ha comentado, estos ficheros contienen únicamente la información estrictamente necesaria para representar la red vial. Cualquier otro tipo de información que se quiera añadir posteriormente a los nodos o tramos para tener una descripción más detallada de ellos puede hacerse fácilmente usando el interfaz adecuado para ello.

Una vez que la aplicación haya leído la información de estos archivos de texto, creará las listas de nodos y tramos necesarias para la representación de la red vial. Cuando el usuario ordene a la aplicación salvar los datos a disco, los datos serán almacenados ya con el formato binario estándar, de manera que la próxima vez que se ejecute la aplicación ya no será necesario cargar de nuevo estos ficheros especiales en formato texto, sino que ya se dispondrá de los ficheros estándares en el formato binario.

Es importante remarcar el hecho de que son necesarios dos ficheros para representar gráficamente la red vial, uno que contenga la información de los nodos y otro la de los tramos, independientemente de que estén en formato binario o en formato texto.

El formato de fichero en modo texto que almacena la información sobre los nodos es el siguiente:

Código_nodo Coordenada_x Coordenada_y

Cada línea almacena la información sobre un nodo, y cada una de las líneas está dividida en estos tres campos. Como ejemplo se adjuntan las primeras líneas de un fichero real :

```
  1    35262.07    43881.89
  2    34587.10    43711.33
  3    34946.36    43758.50
  4    35229.41    43631.49
  5    35545.91    43648.66
  6    34656.05    43061.76
  7    35214.90    43041.80
  8    35715.27    43284.29
  9    34278.65    43254.09
 10    34115.35    42704.32
 11    34474.61    42715.20
 12    35523.35    42617.22
```

```
13    34585.90    42219.50
14    35142.32    42243.45
15    35151.60    41321.60
16    35058.86    42700.69
17    33093.96    43073.63
18    33235.60    42226.50
19    33757.17    42095.91
20    33181.11    41778.54
```

El fichero en modo texto que almacena la información de los tramos es muy similar. También contiene una línea para cada tramo y cada línea está dividida en tres campos:

Código_nodo_origen Código_nodo_destino Código_tramo

A continuación se presentan como ejemplo las primeras líneas de un fichero real:

```
 1     555     1
 1     556     2
 2     222     3
 2     223     4
 2     558     5
 3     557     6
 3     827     7
 4     233     8
 4     828     9
 5     530    10
 5     848    11
 6     227    12
 6     229    13
 6     823    14
 6     825    15
 7     231    16
 7     234    17
 7     253    18
 7     255    19
 8     235    20
 8     237    21
 8     479    22
 9     224    23
 9     225    24
 9     228    25
10     801    26
10     803    27
10     806    28
10     808    29
11     800    30
11     805    31
11     820    32
12     244    33
```

```
12      247     34
12      248     35
12      835     36
13      256     37
13      259     38
13      809     39
13      813     40
14      252     41
14      257     42
14      258     43
15      260     44
15      822     45
15      824     46
15      849     47
16      254     48
16      264     49
17      830     50
17     1157     51
17     1184     52
18      611     53
18      612     54
18      622     55
19      614     56
19      897     57
19      898     58
20      617     59
```

En estos ejemplos puede apreciarse la simplicidad de estos ficheros en modo texto.