Entrada y Salida con ficheros en C++

2 de marzo de 2004

Para efectuar la salida de C++ en ficheros es necesario incluir

```
#include <fstream>
```

al principio del fichero fuente .Este header proporciona dos nuevos tipos `ifstream` para ficheros de entrada y `ofstream` para ficheros de salida. La forma de utilizarlos es la siguiente: Supongamos que deseamos efectuar la salida del programa en el fichero `sal.txt`. Creamos una nueva unidad de salida análoga a `cout` que denominamos `prf` por ejemplo. Para ello incluimos la línea

```
ofstream prf("sal.txt");
```

en nuestro programa.

Análogamente si deseamos leer desde un fichero llamado `input.dat`, creamos una unidad de entrada que denominamos `fin` por ejemplo, analoga a `cin`.

```
ifstream fin("input.dat")
```

Con estas unidades de entrada y salida podemos leer y escribir desde fichero, de forma análoga a como lo hacemos con *cin* y *cout* en la entrada y salida standard.

Por ejemplo, si tenemos una variable *r* y queremos calcular una variable `V=4*pi*r*r*r`, incluiremos las sentencias

```
cout<<"entrar radio"<<endl;
fin>>r;
V=4*pi*r*r*r;
prf<<"volumen="<<V<<endl;
```

A veces es necesario dar el camino completo donde se encuentran los ficheros. Por ejemplo si `input.dat` se encuentra en `C:\Archivos de programas\datos`, habra que darle la ruta completa. Para ello deberemos tener en cuenta que \ es un caracter de escape y para introducir \ hay que hacerlo como \\. Ademas los blancos se interpretan como fin de sentencia, por lo cual deberemos introducir

```
ifstream fin("C:\\""Archivos de Programas""\\datos")
```

Las segundas comillas son necesarias para que las interprete como la apertura de unas segundas comillas y no como el fin de las primeras.

Si tenemos un fichero con datos importantes, deberemos asegurarnos de que no escribimos sobre él por error, destruyendo los datos existentes. esto se consigue definiendo el fichero como de sólo lectura. Esto se consigue con la clase `ios_base`. Por ejemplo si tenemos un fichero `input.dat` y queremos declararlo como de sólo lectura incluiremos la sentencia

```
ifstream fin("input.dat",ios_base::in);
```

Tambien se puede declarar un fichero de solo escritura con `ios_base::out`. Puede ser interesante en determinadas ocasiones el añadir las salidas de sucesivas ejecuciones de un programa a un fichero. Esto se consigue abriéndolo en modo añadir,

```
ofstream fout("salida.dat", ios_base::app);
```

Las diferentes opcionmes se pueden combinar por medio del operador `|`. Por ejemplo,

```
ofstream fout("salida.txt",ios_base::app|ios_base::out);
```

declara `fout` como de sólo escritura y en modo añadir.

Un fichero puede declararse simultáneamente de lectura y escritura mediante el tipo `fstream`. Por ejemplo,

```
fstream iofi("temp.dat",ios_base::in|ios_base::out);
```

declara al fichero `temp.dat` como de escritura y lectura simultáneamente. Por supuesto, ésto rara vez es necesario, salvo en programas que generan una gran cantidad de datos que son utilizados por la siguiente ejecución del programa.

Los ficheros utilizados deben ser cerrados antes de acabar el programa. Esto se consigue mediante las sentencias

```
fout.close();
fin.close();
iofi.close();
```