Manejo de archivos

El manejo de archivos que vamos a necesitar para nuestros códigos es muy elemental. El primer paso consiste en declarar una variable de tipo stream, que apunta a un flujo de información. Éste puede estar asociado a un archivo del sistema, o a algún dispositivo (como en el caso de `cout` para la pantalla, y `cin` para el teclado).

Escritura en archivos

Para escribir información en un archivo, primero debemos declarar una variable de tipo `ofstream` (output file stream) y asociarla con un nombre de archivo específico (si no existe el archivo, será creado). Una vez hecha esta declaración, podemos mandar información a nuestra nueva variable de tipo `ofstream` de la misma forma en que lo hacíamos con `cout`.

En el siguiente ejemplo, declaramos una variable de tipo `ofstream` de nombre `archivosalida` que apunta a un archivo de nombre "archivo.dat". Luego, escribimos números del 1 al 9 en el archivo (cada uno en una línea distinta), y al final cerramos el archivo (es buena costumbre hacerlo antes de terminar el programa).

```
#include<iostream>
#include<fstream>

using namespace std;

int main() {
        ofstream archivosalida( "archivo.dat" );

        for(int i=1; i<=9 ; i++) {
                archivosalida << i << endl;
        }

        archivosalida.close();
        return 0;
}
```

Lectura de archivos

Para leer información de un archivo, primero debemos declarar una variable de tipo `ifstream` (input file stream) y asociarla con un nombre de archivo específico (el archivo debe existir!). Ahora podemos obtener información de nuestra nueva variable de tipo `ifstream` de la misma forma en que lo hacíamos con `cin`.

En el siguiente ejemplo, declaramos una variable de tipo `ifstream` de nombre `archivoentrada` que apunta a un archivo de nombre "archivo.dat". Luego, leemos 9 veces un entero del archivo (cada vez, depositamos lo que leimos en la variable `num`), y escribimos lo que vamos leyendo en la pantalla. Al final cerramos el archivo (es buena costumbre hacerlo antes de terminar el programa).

La línea `archivoentrada >> num` es la que lee un entero del archivo y lo pone en `num`. El programa sabe que debe leer un entero porque ese es el tipo de variable de `num`.

```
#include<iostream>
#include<fstream>

using namespace std;

int main() {
        ifstream archivoentrada( "archivo.dat" );

        for(int i=1; i<=9 ; i++) {
                int num;
                archivoentrada >> num;
                cout << num << endl;
        }

        archivoentrada.close();
        return 0;
}
```

string filename

A veces vamos a querer trabajar con un archivo cuyo nombre está guardado en una variable de tipo `string`. En el siguiente ejemplo declaramos una variable de este tipo, en la que guardamos el nombre de un archivo. Luego abrimos el archivo para escritura, y lo cerramos.

```
#include<iostream>
#include<fstream>

using namespace std;

int main() {
        string nombrearchivo;
        nombrearchivo = "archivo.dat" ;

        ofstream archivoentrada(nombrearchivo.c_str());
        archivoentrada.close();
        return 0;
}
```

Código mágico: `nombrearchivo.c_str()` transforma a la `string nombrearchivo` a un formato entendible para la definición del `ofstream archivoentrada`.

End of file

Si queremos leer un archivo completo, sin conocer su tamaño, podemos usar la propiedad booleana `eof` (end of file) de las variables `ofstream`. El siguiente código lee números reales (`double`) de un archivo de nombre "archivo.dat" hasta que llega a su final, y los va escribiendo de a uno en pantalla.

```
#include<iostream>
#include<fstream>

using namespace std;

int main() {
        ifstream archivoentrada( "archivo.dat" );

        while (archivoentrada.eof() == false) {
                double r_num;
                archivoentrada >> r_num;
                cout << r_num << endl;
        }

        archivoentrada.close();
        return 0;
}
```