Entrada/Salida de archivos

En el capítulo 1 trabajamos con flujos de E/S hacia los dispositivos estándar por defecto definidos en C++ que son el teclado para operaciones de entrada, y el monitor para operaciones de salida. Ahora que tenemos una mejor perspectiva acerca de qué son las clases, capítulo 3, es conveniente dar una repasada al capítulo 1 y en éste artículo trataremos los principales aspectos de las operaciones de E/S en archivos.

Operaciones de escritura en archivos

El archivo de cabecera **fstream.h** define las clases **ifstream**, **ostream** y **fstream** para operaciones de *lectura*, *escritura* y *lectura/escritura* en archivos respectivamente. Para trabajar con archivos debemos crear objetos de éstas clases de acuerdo a las operaciones que deseamos efectuar. Empezamos con las operaciones de escritura, para lo cual básicamente declaramos un *objeto* de la clase *ofstream*, después utilizamos la función miembro **open** para abrir el archivo, escribimos en el archivo los datos que sean necesarios utilizando el operador de inserción y por último cerramos el archivo por medio de la función miembro **close**, éste proceso está ilustrado en nuestro primer programa, *archiv01.cpp*.

```
//***********************************************
//  archiv01.cpp
//  Demuestra la escritura básica en archivo
//  ©1999, Jaime Virgilio Gómez Negrete
//***********************************************

#include <fstream.h>

int main()
{
    ofstream archivo;  // objeto de la clase ofstream

    archivo.open("datos.txt");

    archivo << "Primera línea de texto" << endl;
    archivo << "Segunda línea de texto" << endl;
    archivo << "Última línea de texto" << endl;

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

En el programa se ha creado un objeto de la clase *ofstream* llamado *archivo*, posteriormente se utiliza la función miembro *open* para abrir el arcivo especificado en la cadena de texto que se encuentra dentro del paréntesis de la función. Por lo visto en el capítulo 3 sabemos que podemos invocar a la función constructora de clase de tal manera que el archivo también se puede abrir utilizando la siguiente instrucción:

```
ofstream archivo("datos.txt");  // constructora de ofstream
```

Naturalmente, al utilizar la función constructora ya no es necesario utilizar la función miembro *open*, ésta forma de abrir un archivo es preferida porque el código es más compacto y fácil de leer. De la misma forma que se utilizan manipuladores de

salida para modificar la presentación en pantalla de los datos del programa, igual es posible utilizar éstos manipuladores al escribir datos en un archivo como lo demuestra el programa *archiv02.cpp*, observe que se utiliza un constructor para crear y abrir el archivo llamado *Datos.txt*:

```
//*********************************************
//  archiv02.cpp
//  Demuestra el uso de manipuladores
//  ©1999, Jaime Virgilio Gómez Negrete
//*********************************************

#include <iostream.h>
#include <fstream.h>
#include <iomanip.h>

int main()
{
    ofstream archivo("Datos.txt");  // constructor de ofstream
    int numero;

    cout << "Introduzca un numero:" << endl;
    cin >> numero;
    archivo << "El valor introducido en base 10 es: "
      << numero << endl;

    archivo << resetiosflags(ios::dec);
    archivo << setiosflags(ios::oct);
    archivo << "en base octal es: " << numero << endl;

    archivo << resetiosflags(ios::oct);
    archivo << setiosflags(ios::hex);
    archivo << "y en base hexadecimal es: " << numero << endl;
    archivo << setiosflags(ios::uppercase|ios::showbase);
    archivo << "utilizando los manipuladores uppercase y showbase"
      " el valor es: " << numero << endl;

    archivo << resetiosflags(ios::uppercase|ios::showbase);
    archivo << resetiosflags(ios::hex);
    archivo << setiosflags(ios::showpos|ios::showpoint|ios::fixed);
    archivo << "Utilizando los manipuladores showpos,"
      " showpoint y fixed: " << (float)numero << endl;

    archivo << resetiosflags(ios::showpos|ios::showpoint|
      ios::fixed);
    archivo << "Finalmente el valor es " << numero << endl;

    archivo.close();

    return 0;
}
```

Modos de apertura de archivo

Al especificar la apertura de un archivo como se ha mostrado en los programas anteriores, el programa sobreescribe cualquier archivo existente llamado *Datos.txt* en el directorio de trabajo del programa. Dependiendo del propósito del programa es posible que sea necesario agregar datos a los ya existentes en el archivo, o quizá sea necesario que las operaciones del programa no se lleven a cabo en caso de que el archivo especificado exista en el disco, para éstos casos podemos especificar el modo de apertura del archivo incluyendo un parámetro adicional en el constructor, cualquiera de los siguientes:

- **ios::app** Operaciones de añadidura.
- **ios::ate** Coloca el apuntador del archivo al final del mismo.
- **ios::in** Operaciones de lectura. Esta es la opción por defecto para objetos de la clase *ifstream*.
- **ios::out** Operaciones de escritura. Esta es la opción por defecto para objetos de la clase *ofstream*.
- **ios::nocreate** Si el archivo no existe se suspende la operación.
- **ios::noreplace** Crea un archivo, si existe uno con el mismo nombre la operación se suspende.

- **ios::trunc** Crea un archivo, si existe uno con el mismo nombre lo borra.
- **ios::binary** Operaciones binarias.

De esta manera, podemos modificar el modo de apertura del programa *archiv02.cpp* para que los datos del programa se concatenen en el archivo *Datos.txt* simplemente escribiendo el constructor así: `ofstream archivo("Datos.txt", ios::app);`. Si deseamos que el programa no sobreescriba un archivo existente especificamos el constructor de ésta manera: `ofstream archivo("Datos.txt", ios::noreplace);`. Utilizando los especificadores de modo de apertura se puede conseguir un mayor control en las operaciones de E/S en archivos.

Volver al principio

Operaciones de lectura de archivos

Para abrir un archivo y realizar operaciones de lectura se crea un objeto de la clase *ifstream* y se procede prácticamente de la misma forma que lo expuesto en el apartado anterior. Después de abrir el archivo se puede leer su contenido utilizando las funciones miembro de la clase *ifstream* o bién el operador de extracción. Cuando se lee un archivo, por lo general se empieza al principio del mismo y se leerá su contenido hasta que se encuentre el final del archivo. Para determinar si se ha llegado al final del archivo se puede utilizar la función miembro *eof* como condición de un bucle *while*. Además se puede utilizar la función miembro *fail* para detectar un error al abrir el archivo, esto se demuestra en el siguiente programa, *archiv03.cpp*:

```
//***********************************************
//  archiv03.cpp
//  Demuestra operaciones de lectura de archivos
//  ©1999, Jaime Virgilio Gómez Negrete
//***********************************************

#include <fstream.h>

int main()
{
    ifstream archivo("Besos.txt", ios::noreplace);
    char linea[128];
    long contador = 0L;

    if(archivo.fail())
    cerr << "Error al abrir el archivo Besos.txt" << endl;
    else
    while(!archivo.eof())
    {
        archivo.getline(linea, sizeof(linea));
        cout << linea << endl;
        if((++contador % 24)==0)
        {
            cout << "CONTINUA...";
            cin.get();
        }
    }
    archivo.close();
    return 0;
}
```

El programa crea un objeto de la clase ifstream para abrir el archivo llamado *Besos.txt* utilizando el constructor de clase y especificando la bandera *ios::noreplace* que evita que el archivo sea sobreescrito. Si por algún motivo ocurre un error al abrir el archivo se genera el mensaje de error especificado en la línea 16. En ausencia de errores el programa entra en un bucle *while* el cual está evaluado por efecto de la función miembro *eof( )* de tal manera que el bucle se ejecuta hasta encontrar el final del archivo. Utlizando la función miembro *getline( )* se obtiene una línea de texto y se exhibe en pantalla, línea 21, luego utilizamos una instrucción condicional *if* con el operador de módulo (**%**) para determinar si se han leído 24 líneas de texto. Cada vez que el contador de líneas dividido entre 24 dé como resultado un resíduo de cero el programa se detiene permitiendo leer las 24 líneas de texto previas. Para continuar

se debe presionar la tecla enter y entonces el programa leerá y mostrará en pantalla las siguientes 24 líneas de texto, líneas 22 a la 26.

Analizando el éxito de E/S de archivos

En el programa *archiv03.cpp* se utilizó la función miembro *fail( )* para determinar el éxito de la operación de apertura del archivo *Besos.txt*. La función miembro *fail( )* produce el valor de 1 si ocurre un error en la operación de archivo. Similarmente es recomendable utilizar otras funciones para verificar no solo la apertura de archivo sino también las operaciones de lectura y escritura, las funciones miembro que nos permiten éstas pruebas son las siguientes:

- **good** Produce un 1 si la operación previa fué exitosa.
- **eof** Produce un 1 si se encuentra el final del archivo.
- **fail** Produce un 1 si ocurre un error.
- **bad** Produce un 1 si se realiza una operación inválida.

Tres de las cuatro funciones enlistadas quedan demostradas en el siguiente programa llamado *archiv04.cpp* el cual copia el contenido del archivo llamado *Besos.txt* en uno llamado *Copia.txt*. En primer lugar se crea un objeto de la clase *ifstream* llamado *origen* que nos sirve para abrir el archivo *Besos.txt* para operaciones de lectura, la función miembro *origen.fail( )* nos indica la existencia de un error, en caso de que éste exista se despliega un mensaje en pantalla y el programa termina. Si la apertura del archivo *Besos.txt* fué exitosa se procede entonces a la siguiente parte del programa donde se crea un objeto de la clase *ofstream* llamado *destino* para operaciones de escritura el cual especifica que el archivo a crear se llamará *Copia.txt* y de acuerdo a la bandera *ios::noreplace*, si existe un documento con el nombre de *Copia.txt* la función constructora fallará, éste proceso es detectado por la función miembro *destino.fail( )* desplegando un mensaje en pantalla y terminando el programa. *archiv04.cpp* es un programa que sirve para copiar un archivo basado en caracteres ASCII.

```
//*********************************************************
//  archiv04.cpp
//  Demuestra éxito en operaciones de E/S de archivos
//  ©1999, Jaime Virgilio Gómez Negrete
//*********************************************************

#include <fstream.h>
#include <stdlib.h>

int main()
{
    ifstream origen("Besos.txt");
    char linea[128];

    if(origen.fail())
    cerr << "Error al abrir el archivo Besos.txt" << endl;
    else
    {
        ofstream destino("Copia.txt", ios::noreplace);
        if(destino.fail())
        cerr << "Error al crear el archivo \"Copia.txt\""
          << endl;
        else
        {
            while(!origen.eof())
            {
                origen.getline(linea, sizeof(linea));
                if(origen.good()) // si lectura ok y
                if(origen.eof())  // si eof, -> termina
                exit(1);          // el programa
                else
                destino << linea << endl;
                if(destino.fail())
                {
                    cerr << "Fallo de escritura en archivo"
                      << endl;
                    exit(1);
                }
            }
```

```
        }
        destino.close();
    }
    origen.close();

    return 0;
}
```

En caso que las operaciones de apertura de los archivos involucrados en el programa *archiv04.cpp* sean exitosas, entonces se inicia un bucle en donde se lee en primer lugar una línea de texto, línea 27 del programa, el éxito de ésta operación es monitoreado por la función miembro *origen.good( )*, si ésta función indica que la lectura del archivo fué exitosa entonces la función miembro *origen.eof( )* verifica si la línea en cuestión es el final del archivo, si no es así, entonces la línea leída se escribe en el archivo *Copia.txt*, entonces le corresponde a la función miembro *destino.fail( )* verificar que el proceso de escritura tuvo éxito, línea 33 del programa. El bucle se repite hasta encontrar el final del archivo, condición que se verifica tanto en la linea 25 como en la 29 del programa.

Volver al principio

Operaciones con archivos binarios

Las operaciones de flujos de archivos se ejecutan en forma predeterminada en modo de texto, sin embargo hay ocasiones en donde se requiere realizar operaciones en archivos binarios, como sucede con archivos de estructuras de datos ó aquellos que contienen datos numéricos de punto flotante. A manera de prueba trate de relaizar una copia de un archivo ejecutable utilizando el programa *archiv04.cpp*, se dará cuenta que si bién, el programa no marca errores, el resultado sencillamente no es el esperado. La prueba definitiva es comparar el tamaño en bytes del archivo original contra el tamaño del archivo copiado.

El programa *archiv05.cpp* ejecuta operaciones de E/S en archivos utilizando el modo binario (*ios::binary*), éste programa puede copiar efectivamente un archivo ejecutable, en el ejemplo *Archiv04.EXE*, generando un nuevo archivo llamado *Copia.EXE*, el nuevo código, basado en *archiv04.cpp* es el siguiente:

```
//**********************************************************
//  archiv05.cpp
//  Demuestra operaciones con archivos binarios
//  ©1999, Jaime Virgilio Gómez Negrete
//**********************************************************

#include <fstream.h>
#include <stdlib.h>

int main()
{
    ifstream origen("Archiv04.exe", ios::binary);
    char linea[1];

    if(origen.fail())
    cerr << "Error al abrir el archivo Archiv04.exe" << endl;
    else
    {
        ofstream destino("Copia.exe", ios::binary);
        if(destino.fail())
        cerr << "Error al crear el archivo \"Copia.exe\""
          << endl;
        else
        {
            while(!origen.eof()&&!origen.fail())
            {
                origen.read(linea, sizeof(linea));
                if(origen.good())
                {
                    destino.write(linea, sizeof(linea));
                    if(destino.fail())
                    {
```

```
                        cerr << "Error en el archivo \"Copia.exe\""
                          << endl;
                        exit(1);
                    }
                }
                else if(!origen.eof())
                {
                    cerr << "Error en el archivo \"Archiv04.exe\""
                      << endl;
                    exit(1);
                }
            }
        }
        destino.close();
    }
    origen.close();

    return 0;
}
```

Observará que se utiliza la función miembro *origen.read( )* para leer un byte del archivo especificado por el objeto de la clase *ifstream* llamado *origen*, línea 27. En forma similar, se utiliza la función miembro *destino.write( )* para escribir un byte en el archivo especificado para el objeto de la clase *ofstream* llamado *Copia.EXE*, línea 30 del programa. La comprobación de las operaciones de E/S se realizan como se indicó para el programa *archiv04.cpp*. Si se desea utilizar los operadores de inserción y extracción para operaciones de E/S en archivos binarios se requiere sobrecargar éstos operadores, ésto lo trataremos más adelante cuando se aborde el tema de la sobrecarga de operadores en C++.

Volver al principio

Lectura y escritura en un archivo

Hasta este punto hemos efectuado operaciones, sea de escritura o bién de lectura en un archivo, tanto en formato de texto como en formato binario pero todavía no se han efectuado ámbas operaciones en un mismo archivo. En ciertas aplicaciones es necesario efectuar operaciones de lectura/escritura en un archivo como es el caso de una base de datos, para esto es necesario crear un objeto de la clase **fstream**. Cuando un programa abre un archivo para operaciones de E/S éste mantiene el registro de dos apuntadores de archivo, uno para operaciones de lectura y otro para operaciones de escritura. Como en la mayoría de los casos en que se abre un archivo para operaciones de E/S se efectúa acceso aleatorio, analizaremos ésta condición.

Acceso aleatorio de archivos

En los programas presentados hasta este punto se han realizados operaciones *secuenciales* en el archivo, ya sea para escritura ó lectura de datos, empezando por el principio y continuando hasta el final del mismo. Las operaciones *aleatorias* en un archivo no necesariamente inician por el principio del archivo, en lugar de ésto es posible desplazarse por el contenido del archivo para realizar una operación de E/S. Para mover los apuntadores de archivo a posiciones específicas se utilizan dos funciones: **seekg( )** coloca el apuntador de escritura de archivo en un lugar específico, y **seekp( )** mueve el apuntador de lectura a una posición específica en el archivo, la sintáxis de las funciones es ésta:

- `seekp(desplazamiento, posicion)`
- `seekg(desplazamiento, posicion)`

El parámetro *desplazamiento* especifica la cantidad de bytes que se moverá el apuntador de archivo, puede ser un valor positivo ó negativo. El parámetro *posicion* especifica el lugar del archivo a partir del cual se moverá el apuntador de archivo, de acuerdo a las siguientes banderas

- **`ios::beg`** Desde el principio del archivo
- **`ios::cur`** Desde la posición actual del apuntador

- `ios::end` Desde el fin del archivo

Para demostrar las operaciones aleatorias conviene generar un breve archivo de texto en donde poder efectuar algunas operaciones de E/S, use el siguiente código:

```
//***********************************************
//  letras.cpp
//  Genera un abecedario
//  ©1999, Jaime Virgilio Gómez Negrete
//***********************************************

#include <fstream.h>

int main()
{
    ofstream archivo("Letras.txt");

    for(char letra='A'; letra <='Z'; letra++)
    archivo << letra;
    archivo.close();

    return 0;
}
```

El código genera un alfabeto en el archivo llamado *Letras.txt*, ahora utilizaremos el programa *archiv06.cpp* para "navegar" por el contenido del archivo y generar una palabra amigable en pantalla:

```
//*********************************************************
//  archiv06.cpp
//  Demuestra lectura y escritura en un archivo
//  ©1999, Jaime Virgilio Gómez Negrete
//*********************************************************

#include <fstream.h>

int main()
{
    char letra;
    fstream letras("Letras.txt", ios::in|ios::out);

    letras.seekg(7, ios::beg);   // obtiene la letra H
    letra=letras.get();
    letras.seekp(0, ios::end);
    letras << letra;  // coloca la letra al final

    letras.seekg(-13, ios::end); // obtiene la letra O
    letra = letras.get();
    letras.seekp(0, ios::end);
    letras << letra;

    letras.seekg(-17, ios::end); // obtiene la letra L
    letra = letras.get();
    letras.seekp(0, ios::end);
    letras << letra;

    letras.seekg(0, ios::beg);   // obtiene la letra A
    letra = letras.get();
    letras.seekp(0, ios::end);
    letras << letra;

    letras.seekg(-4, ios::end);
    while(!letras.eof())
    cout.put((char)letras.get());

    letras.close();

    return 0;
}
```

Observe que para posicionar el apuntador de lectura de archivo a partir del fin del mismo se utiliza un número negativo y además la lectura avanza hacia el final del archivo. Por la naturaleza del programa *archiv06.cpp* solo desplegará el mensaje *HOLA* en pantalla una sola vez, esto es porque las letras que forman la palabra

"HOLA" se leen del archivo y a su vez se escriben al final del mismo, el código toma en cuenta las letras previamente escritas, lineas 19 a la 32. Para un mejor entendimiento del funcionamiento del programa utilice el depurador de errores de su compilador en la modalidad de paso por paso.

Volver al principio

Salida a impresora

De la misma manera en que es posible escribir la salida en un archivo, habrá ocasiones en las que es necesario producir constancia en papel utilizando una impresora. En general es posible tratar a la impresora como uno más de los dispositivos disponibles para la salida de datos, se crea un objeto de la clase *ofstream* especificando como nombre de archivo la palabra **PRN**, tal y como lo demuestra el último programa de éste capítulo, *archiv07.cpp*:

```
//*********************************************
//  archiv07.cpp
//  Demuestra la salida a impresora
//  ©1999, Jaime Virgilio Gómez Negrete
//*********************************************

#include <fstream.h>
#include <stdlib.h>

int main()
{
    char texto[256];
    ofstream impresora("PRN");
    ifstream archivo("Besos.txt");

    if (archivo.fail())
    cerr << "Error al abrir el archivo \"Besos.txt\""
      << endl;
    else
    {
        while (!archivo.eof())
        {
            archivo.getline(texto, sizeof(texto));
            if (archivo.good())
            {
                impresora << texto << endl;
                if (impresora.fail())
                {
                    cerr << "Error de escritura en impresora"
                      << endl;
                    exit(1);
                }
            }
            else if (!impresora.eof())
            {
                cerr << "Error al leer el archivo \"Besos.txt\""
                  << endl;
                exit(1);
            }
        }
        archivo.close();
        impresora.close();
    }

    return 0;
}
```

Volver al principio

Descargas

El código fuente de los programas del presente capítulo están contenidos en el archivo comprimido llamado *archiv.zip* (24.5 Kb.), para descargarlo haga clic aquí.

Volver al principio