Entrada/Salida de Información

Unidad 7-Cap 8.

» Ing. Ventre, Luis O.

Entrada/Salida de Información

- Introducción:

- En todos los programas utilizados durante el curso, los datos procesados por lo mismos **se pierden al cerrar el programa** y/o apagar la computadora. Cuando los volúmenes de datos son importantes es necesaria encontrar una forma de **almacenamiento permanente** de esta información.

- *Los datos almacenados juntos bajo un nombre común en un medio de almacenamiento distinto a la memoria principal se llaman* ***archivos de datos.***

- **Esto permite un almacenamiento permanente, y poder compartir entre programas los datos.**

- **Una importante tarea es abrir y comunicarse correctamente con estos archivos antes de comenzar a operar.**

- **Objetos y Métodos en el Flujo de archivos:**

- Para almacenar y recuperar datos fuera de un programa en C++ es necesario:

- **Un archivo.**

- **Un objeto de flujo de archivos.**

- *Archivos:*
- **Colección de datos almacenados juntos bajo un nombre común**, en el disco duro, cd, etc.
- Todo archivo tiene un NOMBRE único identificatorio, llamado **nombre externo.**
- Cada sistema operativo tiene sus restricciones para el nombre externo de los archivos. VER TABLA 8.1 Bibliograf.

Entrada/Salida de Información

- Para no estar fuera de ninguna norma de sistema operativo se utilizara la mas restrictiva, DOS. Máximo 8 caracteres, punto decimal con extensión de 3 caracteres.

datos.txt - registro - precios.dat - exper1.dat

- Existen dos tipos de archivos básicos:

- **Archivos de texto**: conocidos como archivos basados en caracteres.
- Archivos binarios.

- *En este capitulo utilizaremos archivos de texto, por su facilidad en la interpretación y procesamiento por editores de texto y entornos de desarrollo.*

Entrada/Salida de Información

- Objetos de flujo de archivos:

- **“Un flujo de archivos es una ruta de transmisión unidireccional utilizada para conectar un archivo almacenado en un dispositivo físico con un programa”.**

- Cada flujo de archivos tiene su propio método, lo que **determina el sentido** del envío de la información, programa archivo, o viceversa.

- UN FLUJO DE ARCHIVOS QUE LEE O RECIBE DATOS DE UN ARCHIVO ES LLAMADO **FLUJO DE ARCHIVO DE ENTRADA**

- UN FLUJO DE ARCHIVOS QUE ESCRIBE O ENVIA DATOS A UN ARCHIVO ES LLAMADO **FLUJO DE ARCHIVO DE SALIDA**

Entrada/Salida de Información

- **PARA CADA ARCHIVO** QUE USE SU PROGRAMA SIN IMPORTAR EL TIPO DE ARCHIVO, **DEBE CREARSE UN OBJETO DE FLUJO DE ARCHIVOS DISTINTO.**
- SI DESEA QUE SU PROGRAMA LEA Y ESCRIBA EL ARCHIVO SE DEBERA CREAR UN OBJETO DE FLUJO DE ARCHIVO DE ENTRADA Y UNO DE SALIDA.

- Anteriormente con #include<iostream> podíamos manejar flujos entrantes y salientes de consola.

- *Ahora es necesario incluir #include<**fstream>** la cual contiene la información necesaria para llevar a cabo las operaciones de procesamiento de archivos.*

- **Los flujos de archivos de entrada se declaran como IFSTREAM**
- **Los flujos de archivos de salida se declaran como OFSTREAM**

Entrada/Salida de Información

- Ej:

```
ifstream archivo_entr;

ofstream archivo_sal;
```

- La primera instrucción declara que archivo_entr es un **objeto de la clase de flujo de archivos** de entrada ifstream.

- La segunda instrucción declara que archivo_sal es un **objeto de la clase de flujo de archivos** de salida ofstream.

- EN EL PROGRAMA SE TIENE ACCESO A UN FLUJO DE ARCHIVOS POR SU **NOMBRE DE OBJETO** DE FLUJO APROPIADO. UNO PARA LEER Y OTRO PARA ESCRIBIR.
- Es necesario abrir un archivo antes de su procesamiento.

Entrada/Salida de Información

- **Métodos de flujos de archivos:**

- Cada objeto de flujo de archivos tiene acceso a los métodos definidos por su clase. Estos incluyen:

  - Conectar un nombre de objeto con un nombre de archivo externo (llamado abrir un archivo)
  - Determinar si la conexión fue exitosa.
  - Cerrar una conexión (llamado cerrar un archivo)
  - Obtener el siguiente elemento de datos para el programa
  - Colocar un elemento de datos nuevo del programa en un flujo de salida.
  - Detectar cuando se ha alcanzado el final de un archivo.

Entrada/Salida de Información

- El método para apertura de un archivo: conecta el **nombre externo del archivo** en la pc con el **nombre del objeto de flujo usado en forma interna por el programa.**

- **Este método se llama open(). Y es proporcionado por las clases ifstream y ofstream.**

- **Ej:**

```
archivo_entr.open("precios.dat");
```

- **Requiere un solo argumento el nombre externo del archivo a vincular.**
- **La instrucción anterior conecta el objeto de flujo de archivos interno llamado archivo_entr con el archivo precios.dat.**

Entrada/Salida de Información

- Cuando un archivo existente se conecta a un flujo de archivos de entrada los datos quedan disponibles para entrada.

- Del mismo modo un flujo de archivo de salida conectado a un archivo, crea un archivo nuevo, si ya existía se pierden los datos.

- Es buena practica de programación, verificar la conexión con el archivo externo, para esto puede utilizarse el **método fail();** el cual devuelve VERDADERO si FALLO la apertura del archivo.

```
ifstream entrada;

entrada.open("precios.dat");

if (entrada.fail())
{cout<<"El archivo no se abrio con éxito";
exit(1);
}
```

Finaliza ejecución.

Incluir <cstdlib>!!!

Entrada/Salida de Información

```
#include <iostream>
#include <fstream>
#include <cstdlib>    // necesario para usar exit()
using namespace std;

int main()
{
  ifstream archivo_entr;

  // abrir el archivo con el nombre externo precios.dat
  archivo_entr.open("precios.dat");

  if (archivo_entr.fail())  // comprueba una apertura exitosa
  {
    cout << "\nEl archivo no fue abierto con exito"
         << "\nCompruebe que el archivo existe realmente."
         << endl;
    cin.ignore();
    exit(1);
  }

  cout << "\nEl archivo se ha abierto c
       << endl;

  // las instrucciones para leer los dat
  // de un archivo se colocarán aqui

  system("PAUSE");
  return 0;
}
```

Declaro un objeto de flujo de archivo de entrada

Uso el método open para vincular el objeto con el archivo externo

Uso el método fail para verificar la conexión con el archivo

```
El archivo se ha abierto con exito para lectura.
Presione una tecla para continuar . . .
```

```
El archivo no fue abierto con exito
Compruebe que el archivo existe realmente.
```

Entrada/Salida de Información

- En el caso de abrir un archivo para escritura, es necesario recordar que si el archivo existe **se sobrescribe**, por lo que podría implementarse primero una lectura para verificar la existencia del archivo y luego abrirlo para escritura. Ver PRGRAMA 8.2 Pág. 451 se vera a continuación.

- Es posible combinar la declaración de un objeto ifstream u ofstream y su instrucción de apertura en una sola instrucción:

```
ifstream archivo_entr;
archivo_entr.open("precios.dat")
```

```
ifstream archivo_entr("precios.dat")
```

Entrada/Salida de Información

```
#include <iostream>
#include <fstream>
#include <cstdlib>   // necesario para usar exit()
using namespace std;

int main()
{
  ifstream archivo_entr;
  ofstream archivo_sal;

  // intenta abrir el archivo externo precios.dat para lectura
  archivo_entr.open("precios.dat");

  char respuesta;

  if (!archivo_entr.fail())  // si no falla el archivo existe
  {
   cout << "Existe un archivo con el nombre precios.dat.\n"
        << "Desea continuar y sobreescribirlo\n"
        << " con los datos nuevos (s o n): ";
   cin >> respuesta;
   if (respuesta == 'n')
   {
     cout << "EL archivo existente no ser sobreescrito." << endl;
      exit(1);  //termina la ejecución del programa
   }
  }
```

Objetos de flujo de entrada y salida

Intento abrir para comprobar si existe el archivo

Solicito respuesta para sobrescribir

Entrada/Salida de Información

```
  // ahora abre el achivo para escritura
  archivo_sal.open("precios.dat");

  if (archivo_sal.fail())  // comprueba una apertura con exito
  {
    cout << "\nEl archivo no fue exitosamente abierto"
         << endl;
    exit(1);
  }

  cout << "El archivo se ha abierto exitosamente para la salida."
       << endl;

  // las instrucciones para escribir en el archivo se colocarán aquí

  system("PAUSE");
  return 0;
}
```

Abro el archivo para escritura

Verifico apertura del archivo en modo escritura

Entrada/Salida de Información

- NOMBRES INCRUSTADOS E INTERACTIVOS:
- En el ejemplo, el **nombre del archivo a abrir esta incrustado en el código**, si se cambiara el archivo es necesario **recodificar y recompilar el programa**?. Y si el usuario deseara introducir el nombre del archivo a abrir?.
- Aquí se presentan dos alternativas, o declarar una variable de cadena y darle el nombre al inicio del programa.
- O solicitar el ingreso interactivo mediante cin o getline del nombre de la cadena archivo a abrir.

```
int main()
{
  string nombre_archivo =

  ifstream archivo_entr;

  archivo_entr.open(nombre
```

getline(cin,nombre);

```
int main()
{
  // colocar el nombre de archivo
  string nombre_archivo;
  ifstream archivo_entr;

  cout << "Ingrese el nombre del archivo quue desea abrir: ";
  cin  >> nombre_archivo;

  archivo_entr.open(nombre_archivo.c_str()); // abre el archivo
```

Entrada/Salida de Información

- Una alternativa para no necesitar el método c_str; es definir al nombre del archivo como un arreglo de caracteres con un tamaño fijo.

```
// colocar el nombre de archivo
char nombre_archivo[21];
string descrip;
double precio;

ifstream archivo_entr;

cout<<"Ingrese el nombre del archivo a procesar:"<<endl;
cin>>nombre_archivo;


archivo_entr.open(nombre_archivo)
```

```
string nombre_archivo;
string descrip;
double precio;

ifstream archivo_entr;

cout<<"Ingrese el nombre del archivo a procesar:"<<endl;
getline(cin,nombre_archivo);


archivo_entr.open(nombre_archivo.c_str());
```

Entrada/Salida de Información

- Un archivo se cierra cuando se utiliza el **método close().** Este método rompe la conexión entre el objeto de flujo de archivo interno y el nombre externo del archivo.

- Este método no requiere argumento alguno.

```
archivo_entr.close();
```

- Este método tiene un importante sentido debido a las limitaciones de cantidades de archivos abiertos simultáneamente.

- Los archivos abierto al final de la ejecución normal del programa serán **cerrados de manera automática por el sistema operativo.**

Entrada/Salida de Información

- Al usar objetos ifstream y ofstream **el modo** de entrada o salida esta **implicado por el objeto**. Otro medio para crea flujos de archivos es usar objetos **fstream.** Estos pueden utilizarse para entrada o salida, por lo tanto requiere una designación explicita.

- Se declaran igual:

```
fstream  objeto;
```

- Pero al usar el método open obviamente se necesita un parámetro adicional que podrá ser:

```
objeto.open(nombre_archivo, ios::modo);

ios::in (entrada)          ios::out(salida)
ios:app (anexar)           ios::ate(va al final del archivo abierto)
ios::binary (binario)
ios::trunc (elimina el contenido si existe)
ios::nocreate (si existe falla para entrada)
ios::noreplace (si existe falla para salida)
```

- **LECTURA Y ESCRITURA DE ARCHIVOS BASADO EN CARACTERES**

- Leer o escribir archivos basados en caracteres es una **operación casi idéntica a imprimir en pantalla** o aceptar datos desde el teclado, solo debe cambiarse el objeto cout por el nombre del objeto de flujo de archivos.

  Si archivo_sal es un objeto de tipo ofstream son ejemplos validos

- **archivo_sal<< ‘a’;**

- **archivo_sal<< “Hola Mundo”;**

- **archivo_sal<< descrip << ‘ ‘ << precio;**

Entrada/Salida de Información

```
#include <iostream>
#include <fstream>
#include <cstdlib>  // necesario para usa
#include <string>
#include <iomanip>  // necesario para for
using namespace std;

int main()
{
  // colocar el nombre de archivo
  string nombre_archivo = "precios.dat";
  ofstream archivo_sal;

  archivo_sal.open(nombre_archivo.c_str());

  if (archivo_sal.fail())
  {
     cout << "\nEl archivo no fue abierto con exito" << endl;
     exit(1);
  }

  // establece el formato del flujo de salida
  archivo_sal << setiosflags(ios::fixed)
              << setiosflags(ios::showpoint)
              << setprecision(2);
  // envía datos al archivo
  archivo_sal << "Baterias "  << 39.95 << endl
              << "Focos     " <<  3.22 << endl
              << "Fusibles  " <<  1.08 << endl;

  archivo_sal.close();
  cout << "El archivo " << nombre_archivo
       << " fue escrito con exito." << endl;

  system("PAUSE");
  return 0;
}
```

```
El archivo precios.dat fue escrito con exito.
Presione una tecla para continuar . . .
```

Entrada/Salida de Información

- Si el programa **no encuentra el archivo debido a que esta en otro directorio** y no el de compilación. Es factible indicarle el path o camino para llegar al mismo de la siguiente manera.

- Suponga que el archivo esta en disco duro C, en el directorio Trabajo, 2009,

- Debería colocarse:

```
// colocar el nombre de archivo
string nombre_archivo = "c:\\Trabajo\\2009\\precios.dat";
ofstream archivo_sal;
ifstream entrada;

archivo_sal.open(nombre_archivo.c_str());
```

- Porque se coloca doble barra \\?

- **LECTURA Y ESCRITURA DE ARCHIVOS BASADO EN CARACTERES**

- Leer archivos basados en caracteres es una operación **casi idéntica** a aceptar datos desde el teclado, solo debe cambiarse el objeto cin por el nombre del objeto de flujo de archivos.

  Si archivo_ent es un objeto de tipo ifstream son ejemplos válidos

- archivo_ent >> vble;

  se leerá el valor en el archivo y se almacenara en vble

- archivo_ent >> descrip >> precio;

  se leerán dos valores y se almacenan en las variables descrip y precio.

Entrada/Salida de Información

- En la tabla 8.3 de la pagina 463 pueden observarse otros métodos para el ingreso de datos al programa desde el archivo Se verá un resumen mas adelante.

- **A continuación veremos un ejemplo de cómo leer el archivo precios.dat. Este programa utiliza el método good() para detectar el EOF. (Ver tabla 8.2 Métodos de estado del archivo).**

- **La expresión archivo_entr.good() evalúa a verdadero siempre y cuando el valor leído del archivo NO SEA el carácter de END OF FILE.**

- **Por lo tanto al colocarlo en la instrucción: while(archivo_entr.good()) genera un bucle que se ejecuta hasta que se lea del archivo el EOF y no se incluye.**

Entrada/Salida de Información

```
#include <iostream>
#include <fstream>
#include <cstdlib>  // necesario para usar exit()
#include <string>
using namespace std;

int main()
{
  // colocar el nombre de archivo
  string nombre_archivo = "precios.dat";
  string descrip;
  double precio;

  ifstream archivo_entr;

  archivo_entr.open(nombre_archivo.c_str());

  if (archivo_entr.fail()) // comprueba una apertura exitosa
  {
    cout << "\nEl archivo no fue abierto con exito"
         << "\nCompruebe que el archivo existe realmente."
         << endl;
    exit(1);
  }
```

Variables para almacenar el valor recibido del archivo

Objeto de flujo de datos de entrada

Verifico apertura del archivo - conexion

Entrada/Salida de Información

```
// lee y despliega los contenidos del archivo
archivo_entr >> descrip >> precio;
while (archivo_entr.good()) // comprueba el siguiente caracter
{
   cout << descrip << ' ' << precio << endl;
   archivo_entr >> descrip >> precio;
}

archivo_entr.close();

system("PAUSE");
return 0;
}
```

Lee dos datos del archivo

Mientras el dato leído sea distinto a EOF

Imprime los datos leídos

Lee dos datos del archivo

Cierra el archivo

Otra forma de implementar el análisis de final de archivo seria:

while(!archivo_entr.eof()).

El método eof devuelve un valor verdadero solo cuando se ha leído o trans--mitido el marcador EOF.

Por ultimo otra forma de implementar lo mismo seria:

while(archivo_entr>> descrip>> precio)

Esta expresión contiene el operador de extracción >>, el cual extrae datos del archivo y devuelve un valor booleano para indicar si fue exitosa la operación, cuando la extracción evalúa a falso estamos en presencia de EOF

Entrada/Salida de Información

- Otra alternativa para el ingreso de datos desde el archivo es el uso de getline. Este método tiene la siguiente sintaxis:

getline(objetoArchivo, strObj, carácter-terminacion)

- **ObjetoArchivo**, es el nombre del objeto de flujo de archivo.
- **StrObj** es un objeto de clase STRING.
- Carácter de terminación puede ser una constante o vble. por omisión es \n carácter de línea nueva.

```
// lee y despliega los contenidos del archivo
while (getline(archivo_entr,linea))
  cout << linea << endl;
```

- Obviamente “linea” debe ser un string! Ver programa 8.6 Pág.. 466

Resumen tabla 8.3

.**get()** D*evuelve el siguiente carácter extraído de flujo de entrada como int.*

.**get(charVar)** E*xtrae el siguiente carácter y lo asigna a variable CharVar (contrapartida .**put()** coloca de a un carácter en el flujo de arch. de salida)*
.**putback(expresionChar)** *Devuelve un carácter en el flujo de entrada y no altera el archivo.*

.**getline(objArchivo, strObj, caracter_terminacion)** *Extrae caracteres del objeto flujo de entrada hasta que se encuentra con el carácter de terminación y los asigna a strObj. Devuelve un booleano.*

.peek() *devuelve el siguiente carácter sin extraerlo.*

.**ignore(int n)** *Salta los siguientes n caracteres. Si se omite n se saltea el siguiente carácter.*