Entrada y Salida vía Archivos

- Lectura de un archivo de entrada
- Escritura en un archivo de salida

Agustín J. González

ELO329

Lectura de archivos de entrada

- Un archivo de entrada es una secuencia de bytes que es leída desde un archivo de disco
- la clase ifstream define objetos a partir de archivos de entrada.
  - Esta clase es derivada de la clase istream

```
#include <fstream>
```

- Los archivos pueden contener datos binarios o de texto

Apertura de un archivo de entrada

Dos métodos :

- Método #1: Pasar el nombre del archivo al constructor de ifstream
  - Este debe ser un arreglo de caracteres, no un objeto string
- Método #2: Llamar al miembro open() usando un objeto ifstream ya existente

Ejemplos…

Apertura de un archivo de entrada

```
// method #1:
ifstream payFile("payroll.txt");

// method #2:
ifstream empFile;
empFile.open("employee.txt");
```

La apertura lleva el puntero interno de al comienzo del archivo.

Apertura de un archivo de entrada

- Mejor estilo: Declarar un string para mantener el nombre del archivo
- Se debe llamar a c_str() en el string para retornar su arreglo de caracteres.

```
const string PayFilename("payroll.txt");

ifstream payFile( PayFilename.c_str() );
```

Chequeo de errores

- La función is_open() de la clase ifstream retorna verdadero cuando un archivo está abierto

```
const string PayFilename("payroll.txt");

ifstream payFile( PayFilename.c_str());
if( !payFile.is_open() )
{
  cerr << "Cannot open input file: "
       << PayFilename << endl;
  return;
}
```

Chequeo de errores

El objeto ifstream puede ser usado como una expresión booleana, la cual es falsa cuando el archivo está es un estado de falla:

```
ifstream payFile( PayFilename.c_str());
if( !payFile )
{
  cerr << "Cannot open input file: "
       << PayFilename << endl;
  return;
}
```

Lectura de un archivo

- Basta llamar al operador de extracción (>>). Éste salta espacios en blanco e ingresa el archivo de entrada en variables:

```
infile >> age >> salary;
```

- Llamamos a getline(infile, string) para leer una línea entera :

```
string buffer;
getline(infile, buffer);
```

Lectura de archivos

Usamos un lazo y llamanos la función miembro eof() para detectar una condición de fin de archivo:

```
while( !infile.eof() )
{
  infile >> salary;
  infile.ignore(10,'\n');
  getline(infile, buffer);
}
```

Cerrado de Archivos

- Los archivos se cierran automáticamente cuando sus variables pierden su alcance.
- Una variable pierde su alcance al final de su bloque de definición.

```
void OpenFile()
{
  ifstream payFile(PayFilename.c_str());
  //...
  //...
}     // payFile goes out of scope here
```

Cierre de Archivos

- Se puede solicitar cerrar un archivo llamando la función close()

```
void OpenFile()
{
  ifstream payFile(PayFilename.c_str());
  //...
  payFile.close();
  payFile.open( PayFilename.c_str());
  //...
}
```

Función OpenFile

```
bool OpenFile( const string & filename,
               ifstrean & infile )
// PURPOSE:  Opens an input file stream
// RECEIVES: filename - a string
//           infile - input file stream
// RETURNS:  true if the file could be
//           opened, false otherwise.
{
  infile.open( filename.c_str() );

// continued...
```

Funcion OpenFile

```
//...

  if( !infile.is_open() )
  {
    cerr << "Cannot open input file: "
         << filename << endl;
    return false;
  }
  return true;
}
```

Llamado a OpenFile

```
const string PayFilename("payroll.txt");

ifstream payFile;

if( !OpenFile(PayFilename, payFile) )
  return;
```

Escritura a un Archivo de Salida

Apertura de un Archivo de salida

- Usamos las clase ofstream para definir objetos para archivos de salida
  - La clase ofstream se deriva de ostream

```
#include <fstream>
```

- Se crea un nuevo archivo si no existe ya.
- Si se abre un archivo para salida, cualquier contenido previo del archivo es borrado, a menos que indicamos lo contrario.

Apertura de un archivo de salida

Segmento de código con chequeo de errores:

```
const string PayFilename("payroll.txt");

ofstream payFile( PayFilename.c_str());
if( !payFile.is_open() )
{
  cerr << "Cannot create output file: "
       << PayFilename << endl;
  return;
}
```

Agregando contenidos a un archivo

El parámetro ios::app le indica al constructor de la clase ofstream el deseo de agregar contenido a un archivo:

```
ofstream payFile( "payfile.txt", ios::app );
```

Escritura a un archivo de salida

- Usamos el operador de insercion (<<) para escribir hacia un archivo de salida.
- Es posible definir formatos para la salida...

```
outfile << employeeID << '\n'
  << firstName << '\n'
  << lastName << '\n'
  << age << " " << yearsOfService
  << " " << salary << endl;
```

Escritura a un archivo de salida

- Es posible usar otros tipos de formato:

```
outfile
  << "Employee ID: " << employeeID << '\n'
  << "Name: " << firstName << " "
  << lastName << '\n' << "Age:  " << age
  << '\n' << "Years of Service: "
  << yearsOfService << '\n'
  << "Salary: " << salary << endl;
```

Fin