

PROGRAMACIÓN ORIENTADA A OBJETOS 2o. PARCIAL

Practica #5 Archivos en C#

Daniel Vázquez de la Rosa
d_vazquez@outlook.com

Instituto Politécnico Nacional



Escuela Superior de Cómputo

Septiembre 2018

Datos Personales

Crear un Documento en Word que incluya una caratula que contenga tus datos. Dicho documento será guardado como: Grupo_ApellidoPaterno_NombreAlumno_Practica3_POO y deberá contener toda la información de la practica creada.

Objetivos de la Práctica

- Conocer la forma de transferencia de información en C#.
- Aprender las operaciones de lectura y escritura de archivos.
- Desarrollar aplicaciones utilizando almacenamiento permanente.

Introducción

En las aplicaciones que se han desarrollado hasta el momento, los datos únicamente los manejamos en tiempo de ejecución, es decir, una vez cerrada la aplicación los datos que fueron cargados en memoria se pierden.

Es necesario tener un respaldo de toda la información que se maneja, por eso ahora se incluye el tema de archivos. Un archivo se conoce como un elemento para almacenar información de manera permanente; dichos archivos pueden ser .txt, .doc, entre otros.

La manera de almacenar y recuperar información que perdure en el tiempo se basa en el uso de “memoria secundaria”, compuesta esencialmente unidades externas como CD, DVD, Memorias USB, SD, entre otros tipos de almacenamiento.

Streams

La lectura y escritura a un archivo son hechas usando un concepto genérico llamado **stream**.

La idea detrás del **stream** existe hace tiempo, cuando los datos son pensados como una transferencia de un punto a otro, es decir, como un flujo de datos. En el ambiente de Visual Studio, se encuentran varias clases que representan este concepto que trabaja con archivos o con datos de memoria.

Un **stream** es como se denomina a un objeto utilizado para transferir datos. Estos datos pueden ser transferidos en dos direcciones posibles:

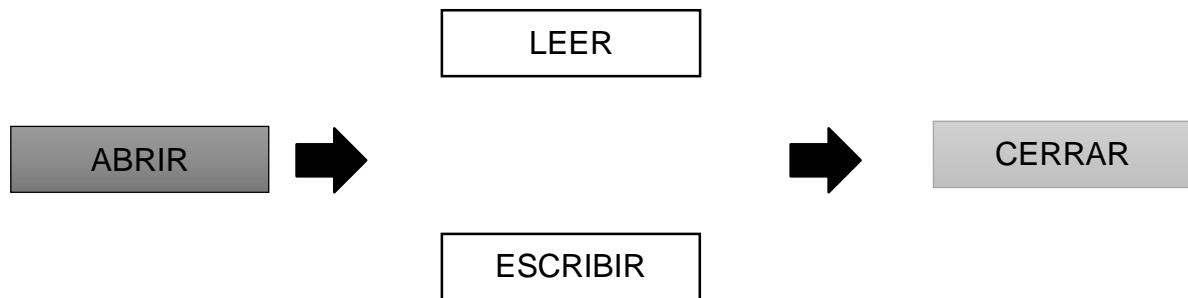
- Si los datos son transferidos desde una fuente externa al programa, entonces se habla de **leer desde el stream**.
- Si los datos son transferidos desde el programa a una fuente externa, entonces se habla de **escribir el stream**.

Frecuentemente, la fuente externa será un archivo, pero eso no necesariamente es el caso, por lo que el concepto es utilizado ampliamente con fuentes de información externas de diversos tipos.

Algunas otras posibilidades fuera de los archivos incluyen:

- Leer o escribir datos a una red utilizando algún protocolo de red, donde la intención es que estos datos sean recibidos o enviados por otro ordenador.
- Lectura o escritura a un área de memoria.
- La consola.
- La impresora.
- Otros.
- Algunas clases que C# provee para resolver este acceso a fuentes diversas, se incluyen las clases de tipo **Reader y Writer**.

Existen tres tipos de operaciones:



Métodos

ReadLine()

Al igual que el conocido `Console.ReadLine()`, este método lee una línea completa de un archivo de texto hasta el cambio de línea más próximo. Al igual que su equivalente de consola, `StreamReader()` no incluye en el string el carácter de cambio de línea.

String Linea = SR.ReadLine()

ReadToEnd()

Este método, por su parte, se encarga de acumular la información que hay desde la lectura anterior (que pudo haberse hecho con `ReadLine`, por ejemplo) hasta el final del archivo, todo con el mismo string.

String Linea = SR.ReadToEnd()

Read()

Finalmente, el método simple Read() se encarga de leer un carácter a la vez, lo que permite procesar símbolo por símbolo el contenido del archivo. Convenientemente, este método reconoce el cambio de línea y se lo salta como si no existiese. Cuando se encuentra con el fin del archivo, retorna un valor (-1), considerando que su retorno es siempre un int (y no un char).

```
int Carácter = SR.Read()
```

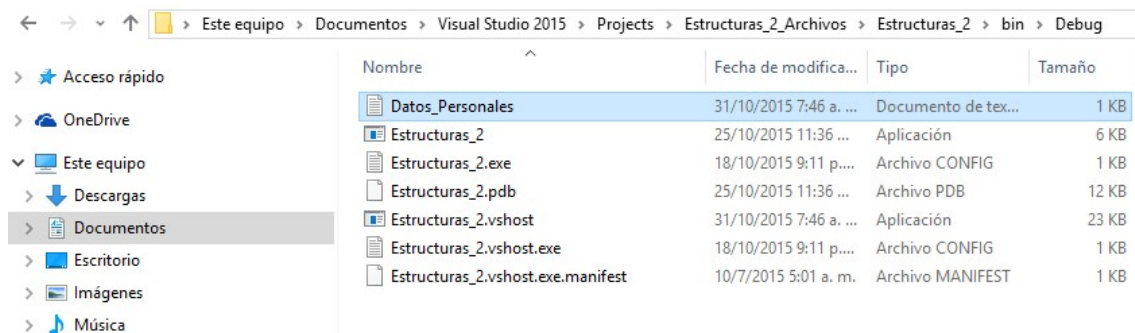
Este mismo método ofrece una declaración alternativa (sobrecarga), donde es posible leer una cantidad específica de caracteres y almacenarlos en un arreglo de enteros.

Lógica de los Archivos:

Leer	Escribir
Abrir flujo desde el archivo	Abrir flujo hacia el archivo
Mientras haya información leer información	Mientras haya información escribir información
Cerrar flujo	Cerrar flujo

Desarrollo

NOTA: Al momento de crear el archivo se puede especificar la ruta en donde se quiere almacenar, de lo contrario se guarda en la carpeta de nuestro proyecto; es decir, en la siguiente ubicación:



Nombre	Fecha de modifica...	Tipo	Tamaño
Datos_Personales	31/10/2015 7:46 a. ...	Documento de tex...	1 KB
Estructuras_2	25/10/2015 11:36 ...	Aplicación	6 KB
Estructuras_2.exe	18/10/2015 9:11 p....	Archivo CONFIG	1 KB
Estructuras_2.pdb	25/10/2015 11:36 ...	Archivo PDB	12 KB
Estructuras_2.vshost	31/10/2015 7:46 a. ...	Aplicación	23 KB
Estructuras_2.vshost.exe	18/10/2015 9:11 p....	Archivo CONFIG	1 KB
Estructuras_2.vshost.exe.manifest	10/7/2015 5:01 a. m.	Archivo MANIFEST	1 KB

Donde el archivo que se encuentra marcado es el que tenemos en uso.

Ejemplo 1

Vamos a desarrollar una sencilla aplicación para conocer cómo crear, escribir y leer archivos de texto.

```
1  using System;
2  //Se incluya la librería de entrada/salida
3  //Para poder utilizar las clases de lectura/escritura
4  using System.IO;
5
6  namespace Ejemplo 1
7  {
8      class Program
9      {
10         static StreamReader Leer;
11         static StreamWriter Escribir;
12
13         static void Main(string[] args)
14         {
15             int Op;
16             Console.WriteLine("\tDigite 1 para crear y escribir sobre un archivo.");
17             Console.WriteLine("\tDigite 2 para leer un archivo.");
18             Op = int.Parse(Console.ReadLine());
19             if(Op == 1)
20             {
21                 Escribir = new StreamWriter("Archivo.txt",true);
22                 Console.WriteLine("\tMensaje de prueba.");
23                 String Cadena = Console.ReadLine();
24                 Escribir.WriteLine(Cadena);
25                 Console.WriteLine("\tEscritura exitosa...");
26                 Escribir.Close();
27             }
28
29             if(Op == 2)
30             {
31                 Leer = new StreamReader("Archivo.txt", true);
32                 String Cadena = Leer.ReadLine();
33                 Console.WriteLine("\tLo leído del archivo es: "+Cadena);
34                 Leer.Close();
35             }
36             Console.ReadKey();
37         }
38     }
39 }
```

Ejemplo_2: Lectura – Escritura

```
1  using System;
2  using System.IO;
3  //Librería para abrir archivos:
4  using System.Diagnostics;
5
6  namespace Ejemplo 2
7  {
8      class Program
9      {
10         struct Persona
11         {
12             public String Nombre;
13             public int Edad;
14             public String Direccion;
15         }
16         static StreamWriter Escribir;
17         static StreamReader Leer;
18         public static void Main(string[] args)
19         {
20             int i = 0, contador = 0;
21             String Opcion, Linea;
22             //Creamos el objeto de tipo estructura:
23             Persona Acceso = new Persona();
24             //Objeto para escribir sobre el archivo
25             do
26             {
27                 Console.Clear();
28                 Escribir = new StreamWriter("Datos_Personales.txt", true);
29                 Console.WriteLine("\tPersona con ID: [" + (i + 1) + "]");
30                 Console.WriteLine("\tNombre: ");
31                 Acceso.Nombre = Console.ReadLine();
32                 Console.WriteLine("\tEdad: ");
33                 Acceso.Edad = int.Parse(Console.ReadLine());
34                 Console.WriteLine("\tDirección: ");
35                 Acceso.Direccion = Console.ReadLine();
36                 //Escribimos sobre el archivo:
37                 Escribir.WriteLine("\tPersona con ID:" + (i + 1));
38                 Escribir.WriteLine("\tNombre: " + Acceso.Nombre);
39                 Escribir.WriteLine("\tEdad: " + Acceso.Edad);
40                 Escribir.WriteLine("\tDirección: " + Acceso.Direccion);
41                 Escribir.WriteLine("\t-----");
42                 i++;
43                 Escribir.Close();
44                 Console.WriteLine("\n\tDesea ingresar otro registro? (S/N)");
45                 Opcion = Console.ReadLine();
46                 if(Opcion == "N" || Opcion == "n")
47                 {
48                     Process.Start("Datos_Personales.txt");
49                 }
50             } while (Opcion == "S" || Opcion == "s");
51
52             //Creamos objeto para leer desde el archivo:
53             Leer = new StreamReader("Datos_Personales.txt");
54             while((Linea = Leer.ReadLine()) != null)
55             {
56                 Console.WriteLine(Linea);
57                 contador++;
58             }
59             Leer.Close();
60             Console.ReadKey();
61         }
62     }
63 }
```

Responda

1. Desarrolle un programa haciendo uso de estructuras que simule una agenda. La aplicación debe solicitar los datos y mantenerlos en tiempo de ejecución (el programa debe estar activo hasta que se escoja la opción salir). Los datos que se deben solicitar son los siguientes: Nombre, DUI, Dirección, Teléfono, Email, Profesión u Oficio. El menú debe tener las opciones: 1. Ingresar Datos. 2. Mostrar Datos. 3. Buscar Persona. 4. Salir. En la opción 3, debe buscar por DUI, ya que es un dato que es único para cada usuario, al encontrar dicha persona el programa debe mostrar los datos encontrados. ***Para esta aplicación se requiere obligatoriamente el uso de archivos.***

Esta Práctica se deberá de entregar a más tardar el día jueves 04/10/2018 antes de las 10:00 PM, toda práctica que no sea enviada dentro de este horario será evaluada con cero.

Al Enviar la Practica, deberás poner en el asunto del correo que envíes la siguiente información: Practica3_Grupo_ApellidoPaterno_ApellidoMaterno_Nombre

Bibliografía

- Deitel, Harvey M. y Paul J. Deitel, Cómo Programar en C#, Segunda Edición, México, 2007.