



Preguntas detonadoras



- ❑ ¿Por qué una aplicación que almacena los datos en memoria los pierde cuando se termina o cuando se apaga el equipo de cómputo?
- ❑ ¿Qué ventajas y desventajas ofrece una aplicación que solamente almacena datos en la memoria principal?
- ❑ ¿Cómo se puede almacenar datos de manera permanente?
- ❑ ¿Qué son los archivos y cómo se clasifican?
- ❑ ¿Un archivo procesa datos?
- ❑ ¿Se le puede definir cualquier nombre y extensión a un archivo?

3

Preguntas detonadoras



- ❑ Si se define la extensión de un archivo como jpg, ¿entonces el archivo almacena de manera automática una imagen o fotografía que pueda ser visualizada mediante software comercial con este propósito?
- ❑ ¿Cuáles con las operaciones internas que se pueden realizar con archivos?
- ❑ ¿Se puede eliminar internamente un dato almacenado en el archivo?
- ❑ ¿Cuáles con las operaciones externas que se pueden realizar con archivos?
- ❑ ¿Qué es y para qué sirve un flujo?
- ❑ ¿Qué se requiere agregar a una aplicación para que administre datos en archivos?
- ❑ ¿Cuáles son las clases necesarias para administrar archivos en una aplicación?
- ❑ ¿Se pueden almacenar objetos en un archivo? ¿Cómo?

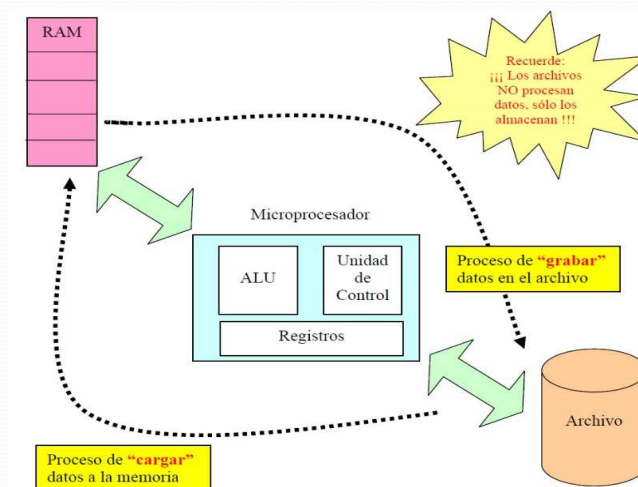
4

¿Por qué usar archivos?

- Algunos programas solamente manejan datos volátiles almacenados en la memoria principal.
- Necesidad de almacenar permanentemente los datos.
- Conservar los datos al salir de la aplicación y/o apagar la computadora.

5

Relación entre la memoria y los dispositivos de almacenamiento secundario



6

Archivos (flujos)

- Algunos autores los identifican también con el nombre de flujos
- Sirven como contenedores de datos en un dispositivo de almacenamiento secundario

7

Definiciones relacionadas con archivos



8

Representación de un archivo



9

Tipos de archivos

Tipos de
archivos
de acuerdo
a su contenido

- De texto
- Binarios

10

Tipos de archivos

Tipos de
archivos
de acuerdo
al modo de
acceso

- Secuenciales
- Relativos (de acceso directo)

11

Archivos secuenciales



12

Archivos relativos (de acceso directo)




13

Archivo vs. archivero

Operación o acción	Archivero	Archivo computacional
Identificar la localización de la información 	Localizando el archivero en particular que contiene las carpetas con la información que se solicita, ya que una oficina puede tener varios archiveros debidamente clasificados e identificados	Identificando la base de datos correspondiente a la información que se solicita. Una base de datos es una colección de archivos relacionados. P. Ejem. Profesores, estudiantes y materias están correlacionados.

14

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
Identificar el lugar exacto donde se encuentra la información 	Regularmente un archivero contiene varios cajones, cada uno con información debidamente clasificada y ordenada.	Se recomienda que los archivos contengan datos relacionados con un objeto de interés en particular y no de varios. P. Ejem. Sólo datos de estudiantes.

15

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
Operaciones 	En un archivero se pueden agregar, extraer o cambiar documentos de las carpetas.	Básicamente un archivo solo tiene 2 operaciones para el manejo de sus registros: <ul style="list-style-type: none"> • Lectura • Escritura Las demás operaciones se realizan como consecuencia de éstas.

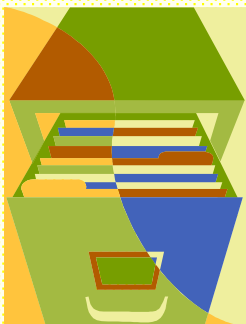
16

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
Apertura 	<p>Obviamente cuando se requiere agregar o consultar carpetas del cajón de un archivero, es necesario primero abrirlo.</p>	<p>Para acceder los datos de un archivo es necesario abrirlo. Existen varios modos de apertura de los archivos dependiendo de las operaciones que se deseen realizar en él.</p>


17

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
Clasificación de los datos 	<p>Los cajones de los archiveros tienen separadores o pequeñas pestañas para identificar las carpetas. Estas facilitan el acceso, ya sea la inserción o la extracción de un carpeta en particular.</p>	<p>Los datos pueden ser almacenados de muchas formas diferentes en los archivos y de esto depende la facilidad (o dificultad) que el archivo muestre para ciertas operaciones de acceso. A estas formas de almacenamiento se les conoce como "organización del archivo".</p>


18

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
<p>Cierre</p> 	<p>Cuando ya no se desea utilizar un cajón de un archivero es necesario cerrarlo, ya que de no hacerlo, se corre el riesgo de dañar o perder la información.</p>	<p>Cuando se termina de utilizar un archivo es necesario cerrarlo. De esa forma se vacía la memoria caché y se asegura almacenar y proteger los datos.</p>

19

Archivo vs. archivero

<i>Operación o acción</i>	<i>Archivero</i>	<i>Archivo computacional</i>
<p>Seguridad</p> 	<p>Cuando ya no se desea utilizar un cajón de un archivero es necesario cerrarlo, ya que de no hacerlo, se corre el riesgo de dañar o perder la información.</p>	<p>Cuando se termina de utilizar un archivo es necesario cerrarlo. De esa forma se vacía la memoria caché y se asegura almacenar y proteger los datos.</p>

20

Definiendo el nombre del archivo

- Declarar una variable de tipo `string` para almacenar el nombre del archivo (incluyendo la ruta de acceso)
- Usar **dobles diagonales** para separar las carpetas de la ruta
- Usar una cadena *verbatim* (identificada por el símbolo `@`)

21

Ejemplos de nombres de archivos

- Mediante una cadena normal:
 - `string` NombreArchivo =
 `"c:\\Datos\\NoSirven\\Archivo.txt";`
- Mediante una cadena *verbatim*:
 - `public string` NombreArchivo2 =
 `@ "c:\\MisDatos\\Ejemplo.txt";`

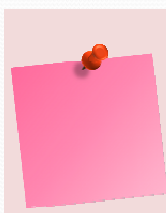
22

Algunas extensiones de nombres de archivos

Extensión	Descripción
.exe	Archivo binario con una aplicación ejecutable.
.bat	Archivo de texto de procesamiento por lotes con comandos ejecutables por el sistema operativo.
.jpg, .bmp, .gif, .tif	Archivos binarios con imágenes o fotografías
.mpg, .avi	Archivos binarios con video
.mp3, .wav	Archivos binarios con audio
.htm, .html	Archivos de texto con páginas web
.doc	Archivo binario con un documento de Microsoft Word
.xls	Archivo binario con un documento de Microsoft Excel
.ppt	Archivo binario con un documento de Microsoft Power Point
.txt	Archivo de texto sin formato

23

Extensiones de nombres de archivos



Aunque el programador puede asignar cualquier nombre y extensión a un archivo, se recomienda que les coloque nombres relevantes de acuerdo a su contenido y el hecho de asignarle una extensión particular, no convierte al archivo al formato de la aplicación por defecto para dicha extensión.

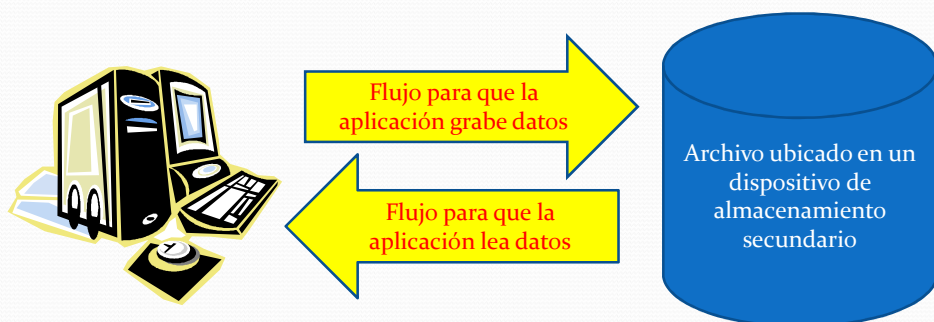
24

Flujos

- Para almacenar o consultar datos en un archivo es necesario establecer un canal de comunicación.
- Este canal se conoce como **flujo** (*stream* en inglés).
- Este canal es un mensajero entre la aplicación y el archivo ubicado en un dispositivo de almacenamiento secundario (disco duro, memoria USB, CD, etc.)

25

Flujo (*stream*)



26

Operaciones con archivos

Operaciones
con
archivos

- Lectura
- Escritura

27

Apertura de archivos



28

Actividades para grabar datos en un archivo

Orden	Actividad	Acciones
1	Abrir el archivo	Abrir un flujo en modo de escritura hacia el archivo. Si el archivo no existe, entonces debe crearse.
2	Escribir	Grabar el dato en el archivo (por medio del flujo)
3	Cerrar	Cerrar el archivo (por medio del flujo)

29

Actividades para leer datos de un archivo

Orden	Actividad	Acciones
1	Abrir el archivo	Abrir un flujo en modo de lectura hacia el archivo. Asegurarse que el archivo exista.
2	Leer	Leer datos hasta llegar al final del archivo.
3	Cerrar	Cerrar el archivo (por medio del flujo)

30

Espacio de nombres requerido

```
using System.IO;  
// Para el uso de archivos
```

31

Algunas clases de System.IO

Clase	Descripción
FileStream	Permite crear objetos para leer o escribir datos en archivos. Para ello es necesario definir el nombre completo del archivo (incluyendo la ruta de ubicación de sus carpetas)
StreamWriter	Permite crear objetos que implementan un sistema de escritura de datos basado en secuencias de caracteres.
StreamReader	Permite crear objetos que implementan un sistema de lectura de datos basado en secuencias de caracteres.
File	Esta clase contiene métodos estáticos para manipular archivos como su creación, copiarlos, eliminarlos, moverlos o detectar su existencia.

32

Estableciendo el flujo de bytes a través de un objeto de la clase *FileStream*

- Para escribir o leer datos en un archivo, es necesario abrirlo estableciendo un flujo al crear un objeto de la clase **FileStream**
- Es necesario definir el nombre del archivo (incluyendo la ruta de sus carpetas).

33

Constructores de la clase *FileStream*

- `FileStream(string NombreArchivo, FileMode ModoDeApertura);`
- `FileStream(string NombreArchivo, FileMode ModoDeApertura, FileAccess ModoDeAcceso);`

34

Modos de apertura de archivos (FileMode)

FileMode	Uso
CreateNew	Crea un nuevo archivo. Si el archivo existe dispara una IOException
Truncate	Abrir un archivo existente. Una vez abierto, el archivo será truncado a cero bytes de longitud.
Create	Crea un nuevo archivo. Si el archivo existe será sobrescrito.
Open	Abrir un archivo existente. Si no existe dispara una FileNotFoundException.
OpenOrCreate	Abrir un archivo existente, si no existe, lo crea.
Append	Abrir un archivo para agregar datos al final en caso de existir; de lo contrario crea un archivo nuevo.

35

Modos de acceso de archivos (FileAccess)

FileAccess	Uso
Read	Acceso al archivo en modo de solo lectura
ReadWrite	Acceso al archivo en modo de lectura y escritura
Write	Acceso al archivo en modo de solo escritura

36

Estableciendo el flujo de escritura a través de un objeto de la clase *StreamWriter*

- Para escribir datos en un archivo, es necesario abrirlo en modo escritura estableciendo un flujo al crear un objeto de la clase **StreamWriter**
- Ejemplo:

```
StreamWriter flujoEscritura= new  
StreamWriter(NombreArchivo);
```

37

Estableciendo el flujo de lectura a través de un objeto de la clase *StreamReader*

- Para leer datos de un archivo, es necesario abrirlo en modo lectura estableciendo un flujo al crear un objeto de la clase **StreamReader**
- Ejemplo:

```
StreamReader flujoLectura= new  
StreamReader(NombreArchivo);
```

38

¿Cómo detectar si existe un archivo?

```
if (File.Exists(NombreArchivo))  
{  
    .....  
}
```

39

```
static void Agregar()  
{  
    string NombreArchivo = "c:\\Datos\\NoSirven\\Archivo.txt"; // Declaración de una variable con el nombre del archivo  
    string _strLinea = ""; // Declaración de la variable con la línea de texto  
  
    Console.WriteLine("\n\nTeclee la línea de texto que desea almacenar: ");  
    _strLinea = Console.ReadLine();  
  
    System.IO.FileStream flujo=null; // Declaración de un flujo mediante la clase FileStream  
    System.IO.StreamWriter flujoEscritura=null; // Declaración de un flujo de solo escritura mediante la clase StreamWriter  
  
    // Intenta establecer un flujo de solo escritura  
    try  
    {  
        if(System.IO.File.Exists(NombreArchivo)) // Verifica si el archivo existe  
        {  
            flujo = new System.IO.FileStream(NombreArchivo, System.IO.FileMode.Append); // Abre el archivo en modo "Agregar"  
            flujoEscritura = new System.IO.StreamWriter(flujo); // Establece el flujo de solo escritura con el archivo abierto  
        }  
        else  
            flujoEscritura= new System.IO.StreamWriter(NombreArchivo); // Crea el archivo  
  
        flujoEscritura.WriteLine(_strLinea); // Graba la línea de texto en el archivo a través de su flujo  
        Console.WriteLine("\nLínea de texto agregada al archivo !!!");  
    }  
    catch(Exception ex)  
    {  
        Console.WriteLine("\nERROR: " + ex.Message);  
    }  
    finally  
    {  
        // Cierra los archivos  
        if (flujoEscritura != null)  
            flujoEscritura.Close();  
        if (flujo != null)  
            flujo.Close();  
        Console.ReadKey();  
    }  
}
```



```
static void Listar()
{
    string NombreArchivo = "c:\\Datos\\NoSirven\\Archivo.txt"; // Declaración del nombre del archivo
    string _strLinea = ""; // Declaración de la variable con la línea de texto

    System.IO.StreamReader flujoLectura=null; // Declaración de un flujo de solo lectura con la clase StreamReader

    // Intenta leer las líneas de texto almacenadas en el archivo
    try
    {
        flujoLectura = new System.IO.StreamReader(NombreArchivo); // Intenta abrir el archivo en modo solo lectura
        while (true)
        {
            _strLinea=flujoLectura.ReadLine(); // Lee una línea de texto del archivo a través de su flujo
            if (_strLinea == null)
                break;
            Console.WriteLine(_strLinea); // Despliega en pantalla la línea de texto leída del archivo
        }
    }
    catch(Exception ex)
    {
        Console.WriteLine("ERROR: "+ex.Message);
    }
    finally
    {
        // Cierra el archivo
        if (flujoLectura != null)
            flujoLectura.Close();
        Console.WriteLine("\nFIN DEL ARCHIVO");
        Console.ReadKey();
    }
}
```

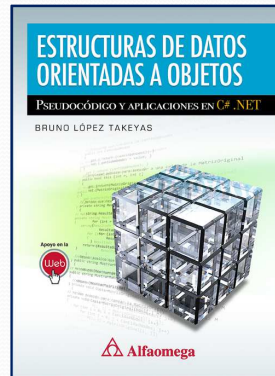
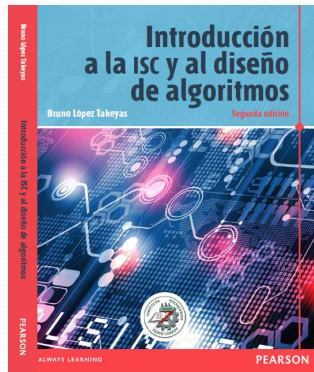
Cerrar el archivo

- Una vez usado el archivo, entonces debe cerrarse mediante:

```
if (flujo!= null)
    flujo.Close();
```

Otros títulos del autor

<http://www.itnuevolaredo.edu.mx/Takeyas/Libro>



takeyas@itnuevolaredo.edu.mx



Bruno López Takeyas