

**Estructuras de Archivos**

**Héctor Andrey Hernández Alonso**

**MANUAL DE PROGRAMADOR**

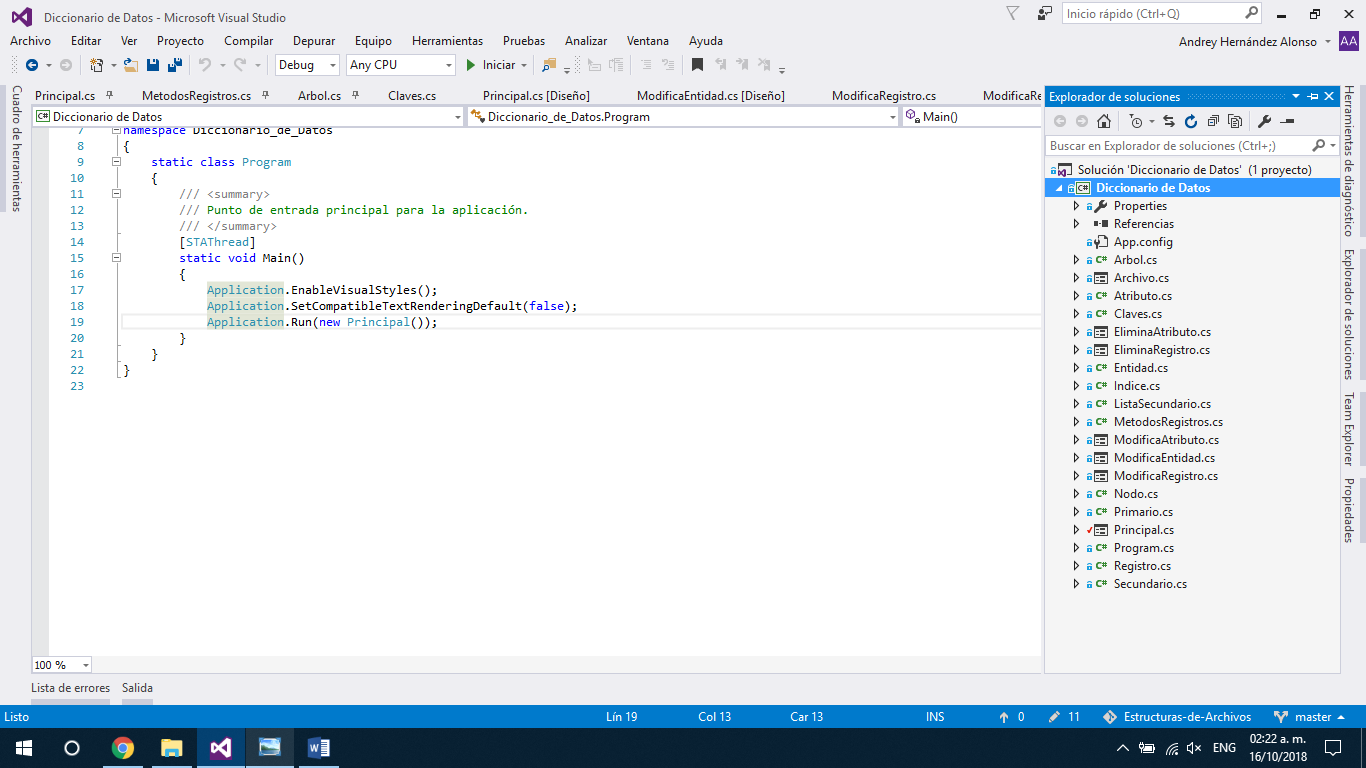


INDICE

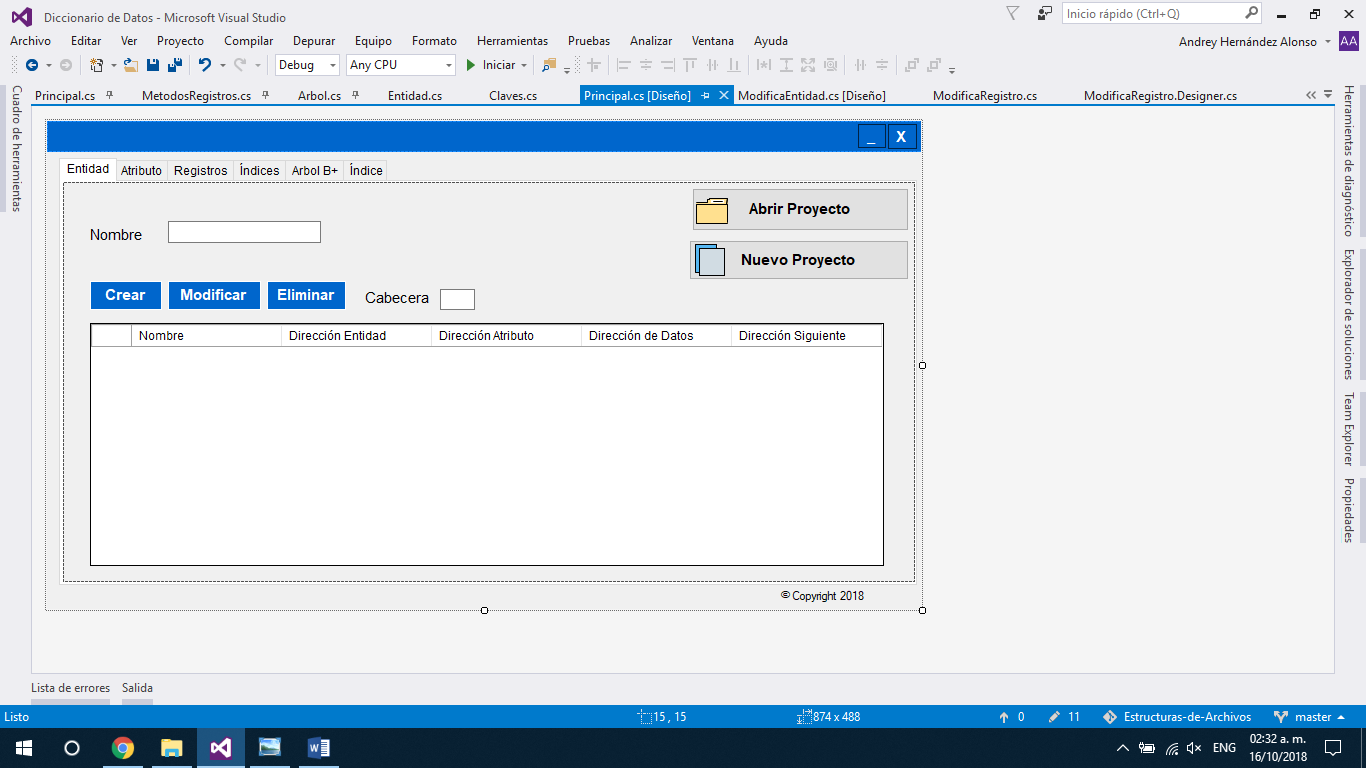
|  |  |
| --- | --- |
| ESTRUCTURA DEL PROYECTO | ------------------------------------------------------------2 |
| Interfaz Ventana Principal | ------------------------------------------------------------3 |
| Lectura y Escritura de Archivos Binario | ------------------------------------------------------------4 |
| Metadatos Entidad | ------------------------------------------------------------5 |
| Inserción Entidad | ------------------------------------------------------------7 |
| Eliminación Entidad | -----------------------------------------------------------10 |
| Modificación Entidad | -----------------------------------------------------------12 |
|  |  |

**ESTRUCTURA DEL PROYECTO**

Cada Clase se explicará en su respectivo apartado.



**Interfaz Ventana Principal ( Entidad )**



Para poder controlar todo el proyecto, lo primero es contar con un Archivo que funcione como **diccionario de datos** para poder administrar todas las altas y bajas de Entidades y Atributos.

**Lectura y Escritura de Archivos Binarios**

Para la lectura de archivos se necesita usar objetos de la Clase BinaryReader.

**BinaryReader**

Lee tipos de datos primitivos como valores binarios en una codificación específica.

**BinaryReader** br = new **BinaryReade**r( File.Open( filename, FileMode.Open ) **)**

Filename 🡪 Nombre del Archivo

FileMode 🡪 Modo en que se manejara el archivo (Apertura, Escritura, Crear, etc.)

Referencia:

<https://docs.microsoft.com/en-us/dotnet/api/system.io.binaryreader?view=netframework-4.7.2>

**BinaryWriter**

Escribe tipos primitivos en binario en una secuencia y admite escribir cadenas en una codificación específica.

**BinaryWriter** bw = new **BinaryWriter**( File.Open( filename, FileMode.Open ) **)**

Referencia:

<https://docs.microsoft.com/en-us/dotnet/api/system.io.binarywriter?view=netframework-4.7.2>

**Apertura de Archivos**

Se considera que las Entidades y los Atributos están en un mismo archivo.

Los archivos son binarios.

Para la lectura de archivos binarios se usa un objeto de la clase BinaryReader

**METADATOS DE LA ENTIDAD**

**Nombre**

Identificador propio del objeto que se requiere representar

**Dirección de la entidad**

Representa la dirección de archivo en donde se almacena la entidad

**Dirección de los atributos**

Representa la dirección de archivo en donde se encuentra el primer atributo.

**Dirección de los datos**

Representa la dirección de archivo en donde se encuentra el primer registro de datos.

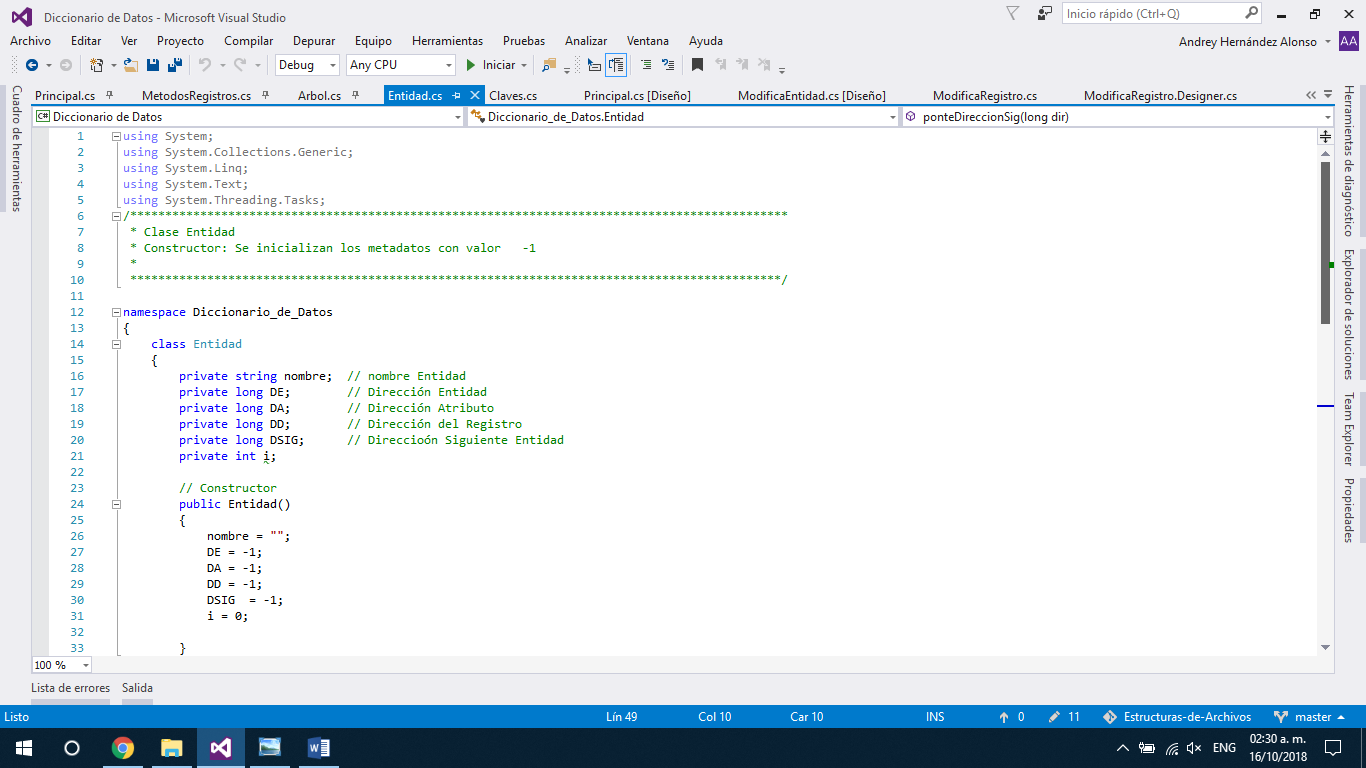
**Dirección de la siguiente entidad**

Representa la dirección de la siguiente entidad

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | DE | DA | DD | DSIG |

|  |  |  |
| --- | --- | --- |
| NOMBRE | SIMBOLOGIA | METODO |
| Nombre | Nombre | br.ReadString() |
| Dirección Entidad | DE | br.ReadInt64() |
| Dirección Atributo | DA | br.ReadInt64() |
| Dirección Registro | DD | br.ReadInt64() |
| Dirección Siguiente | DSIG | br.ReadInt64() |

Para poder controlar las Entidades, se utiliza una **Clase Entidad**



La Clase Entidad cuenta solo con los métodos get’s y set’s de sus metadatos.

Los algoritmos que se presentan en el manejo de Entidades son:

* Inserción de Entidad
* Eliminación de Entidad
* Modificación de Entidad

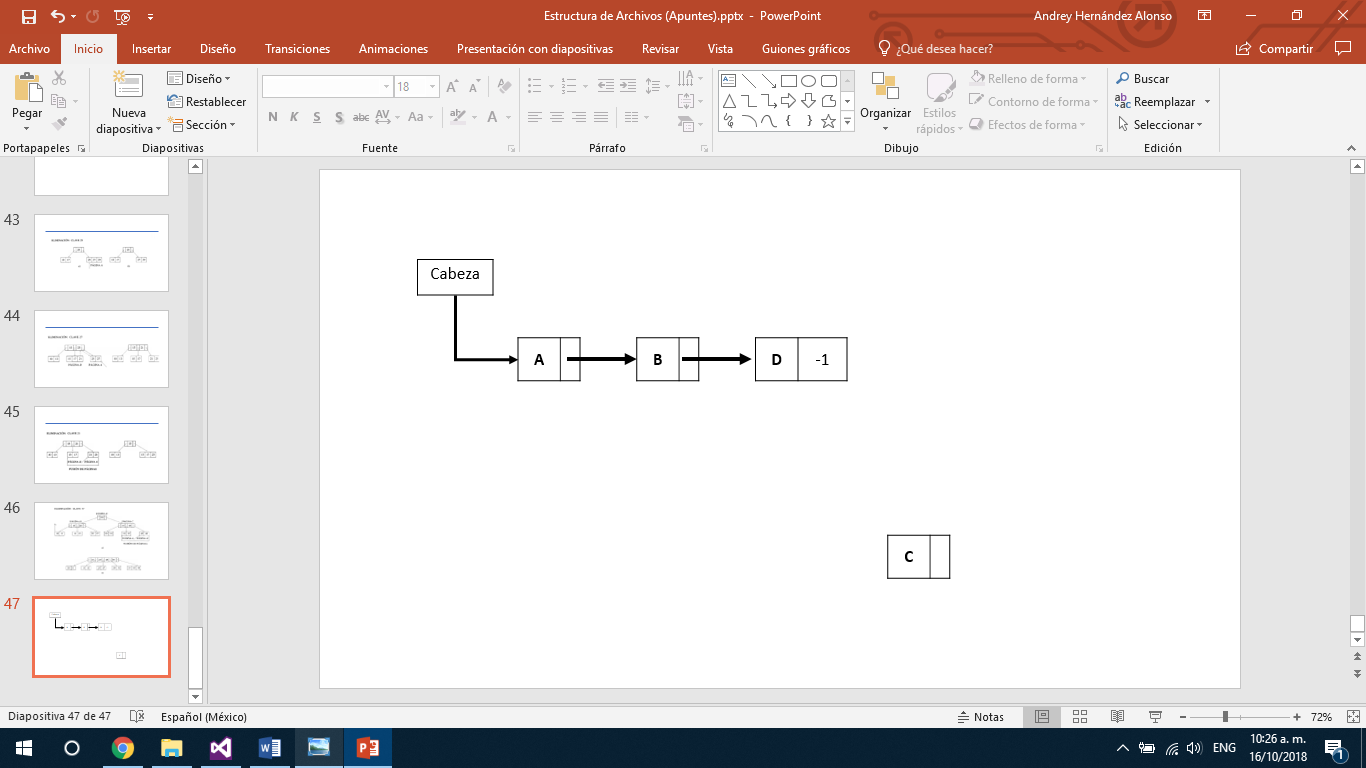
**Algoritmo: Inserción de Entidad**

Consideraciones al insertar:

* Inserción Ordenada
* El primer campo de 8 bytes del archivo es la Cabeza de la lista, este indica la primera Entidad de forma ordenada
* La dirección de cada Entidad siempre será el tamaño del Archivo en ese momento

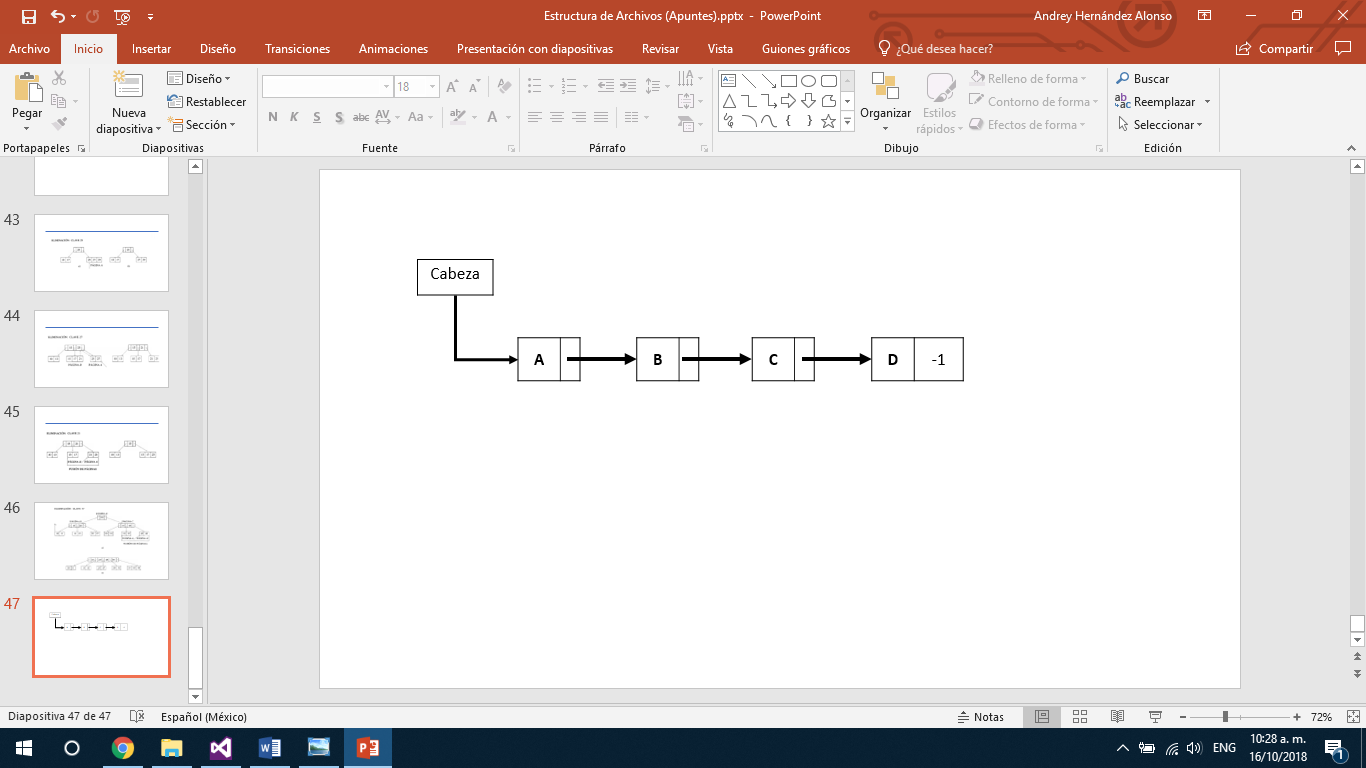
**Pasos para la inserción de Entidades**

1. Leer la Cabeza del Archivo y posicionar el apuntador. Si la Cabeza del Archivo indica -1, entonces no hay ninguna Entidad en el Archivo.
2. Buscar la nueva posición de la Entidad en el archivo.
3. Agregar la entidad con sus campos y actualizar su apuntador al siguiente
4. Si la entidad nueva, es la primera alfabéticamente, entonces la Cabeza del archivo apunta a la nueva Entidad



Insertar ( C )

|  |  |
| --- | --- |
| **C** |  |



Para comparar las cadenas se usa el método CompareTo()

**CompareTo()**

Compara dos objetos y devuelve un valor entero con 3 posibles resultados donde:

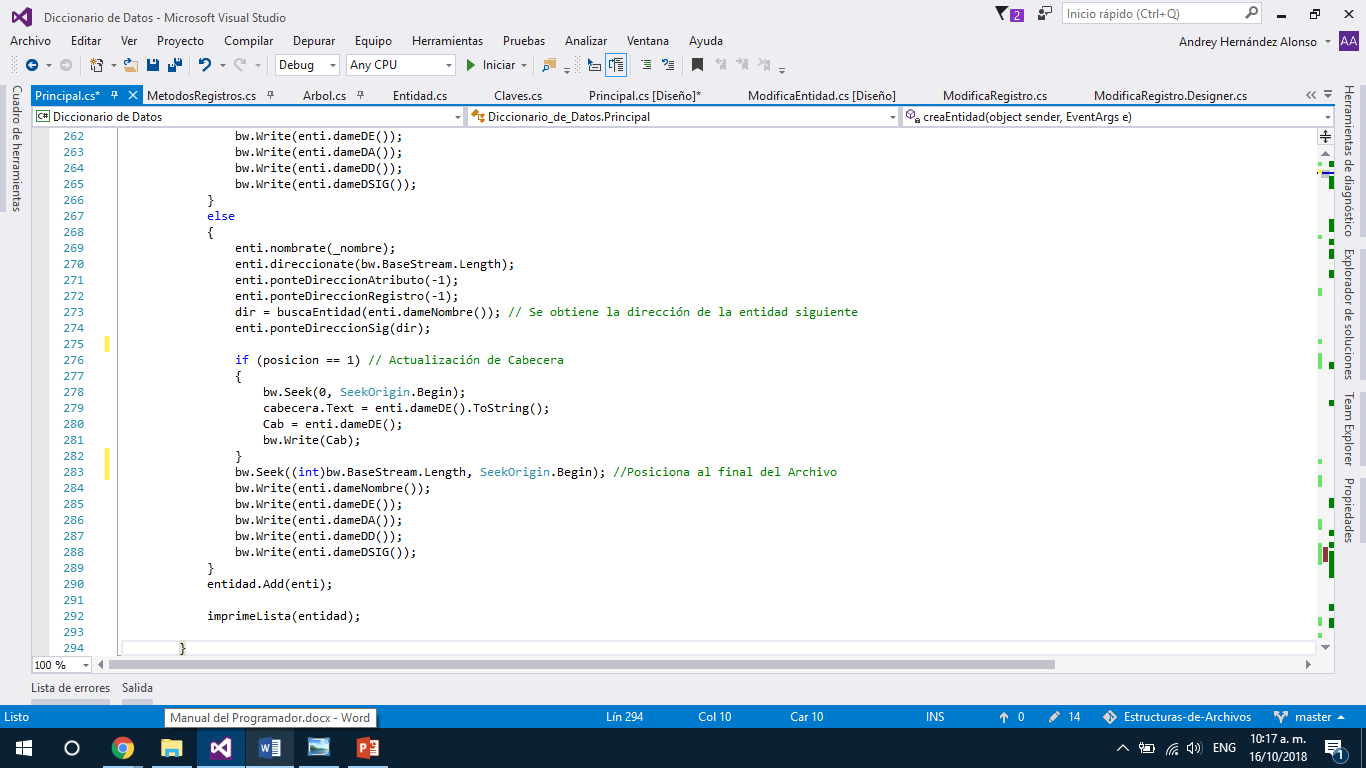
* Si el valor es -1, La entidad 1 es antes que la nueva entidad
* Si el valor es 0, La entidad 1 y la nueva son las mismas
* Si el valor es 1, La entidad 1 va después de la nueva entidad.

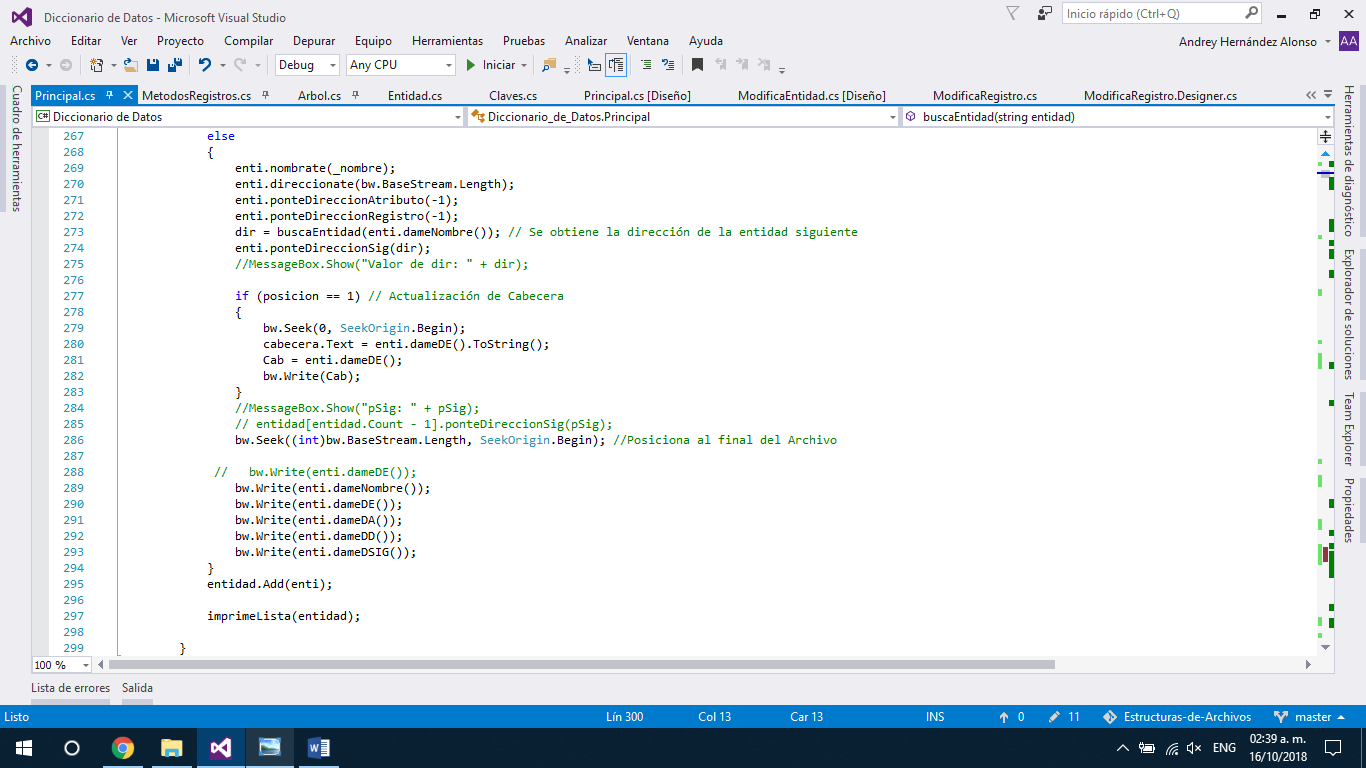
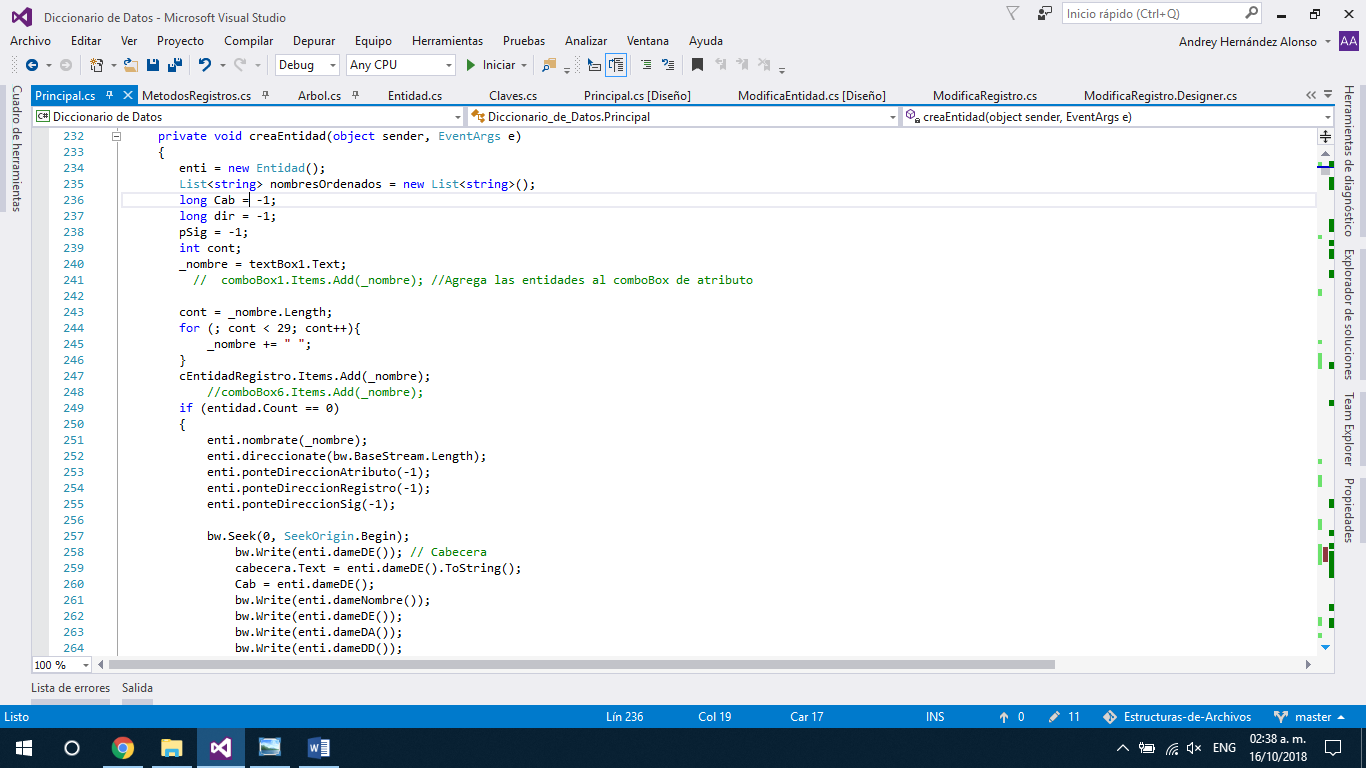
**int** compara;

compara = entidad1.nombre.**CompareTo(**nuevaEntidad.nombre**)**;

Referencia

<https://docs.microsoft.com/en-us/dotnet/api/system.int32.compareto?view=netframework-4.7.2>





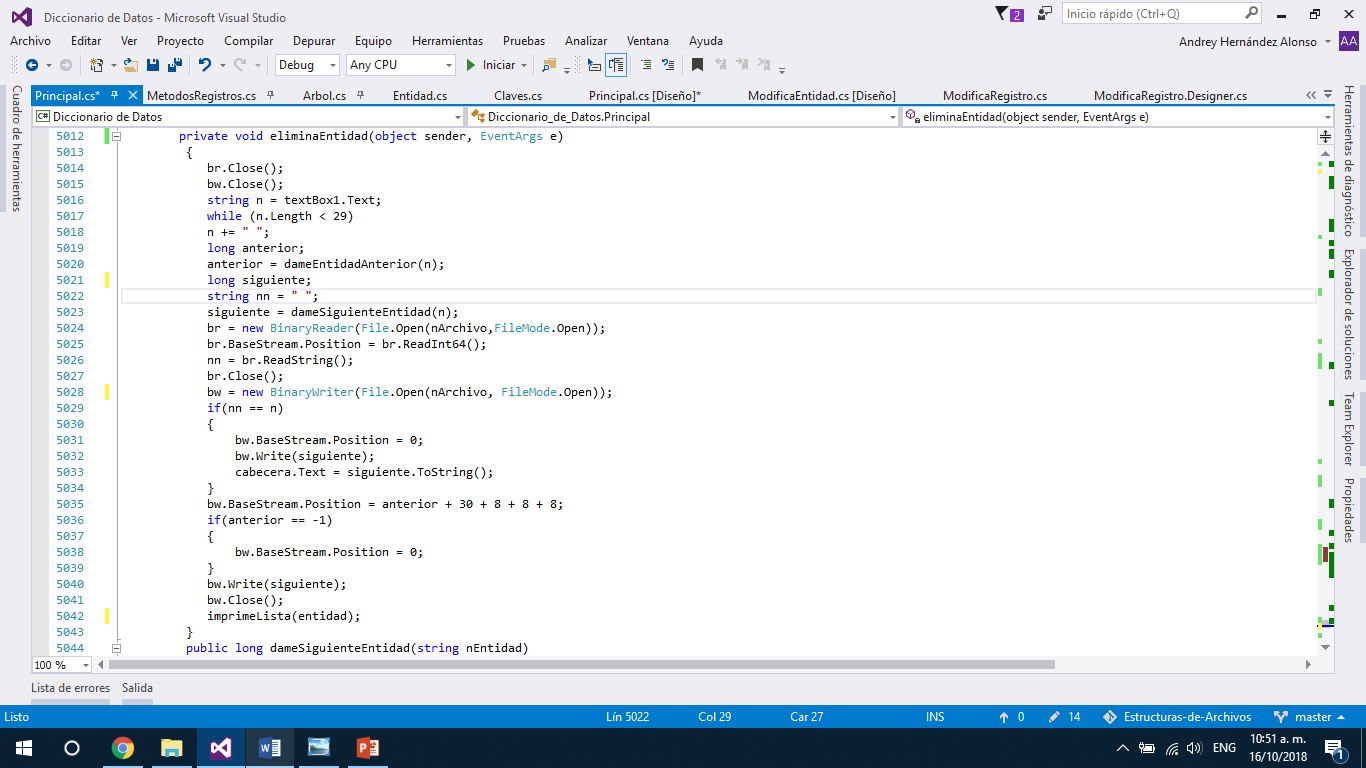
**Algoritmo: Eliminación de Entidad**

Consideraciones al eliminar:

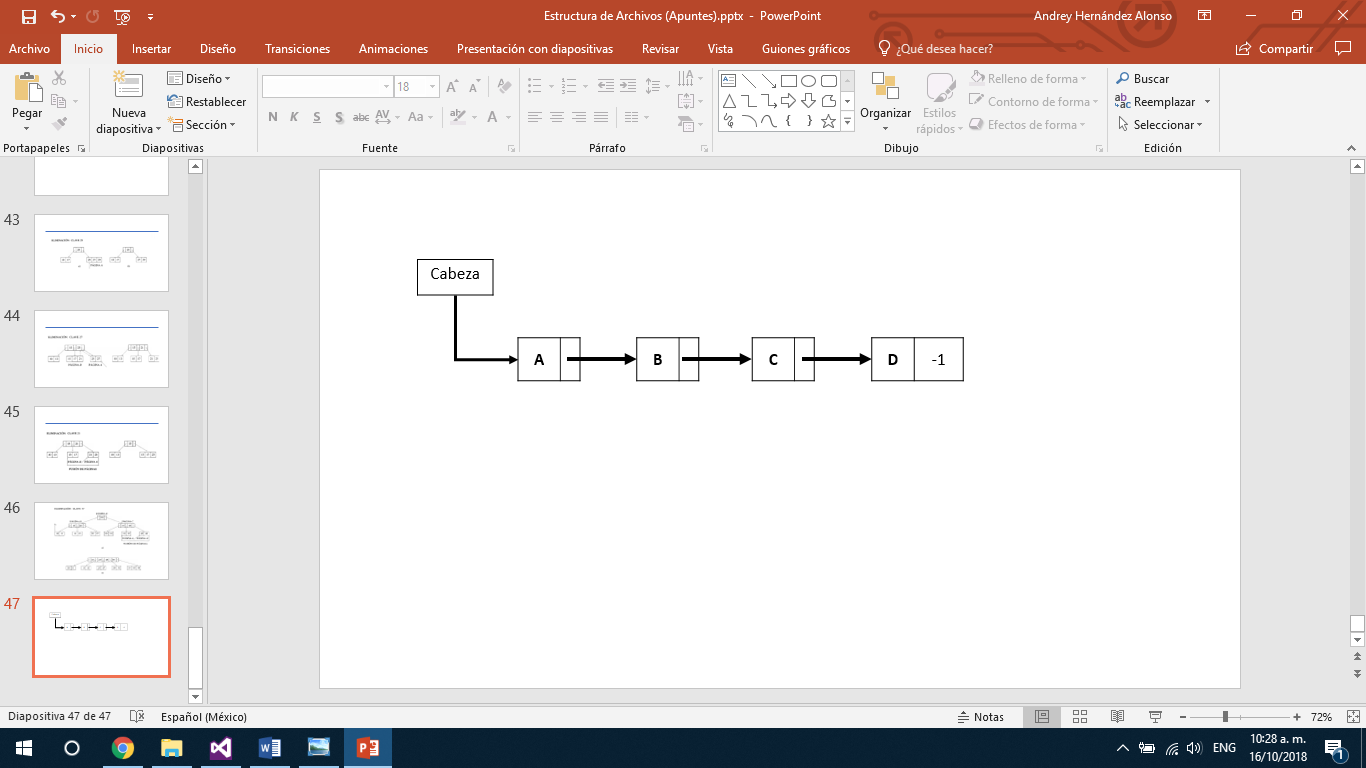
* Las Entidades no se eliminan, solo se deja de apuntar a ellas.
* Se debe guardar en un apuntador auxiliar el apuntador que apunta a la Entidad a Eliminar
* Se debe guardar en un apuntador auxiliar el apuntador siguiente de la Entidad a eliminar
* Si se elimina la primera entidad, actualizar la Cabeza de la lista con su siguiente

Pasos para la Eliminación de Entidad:

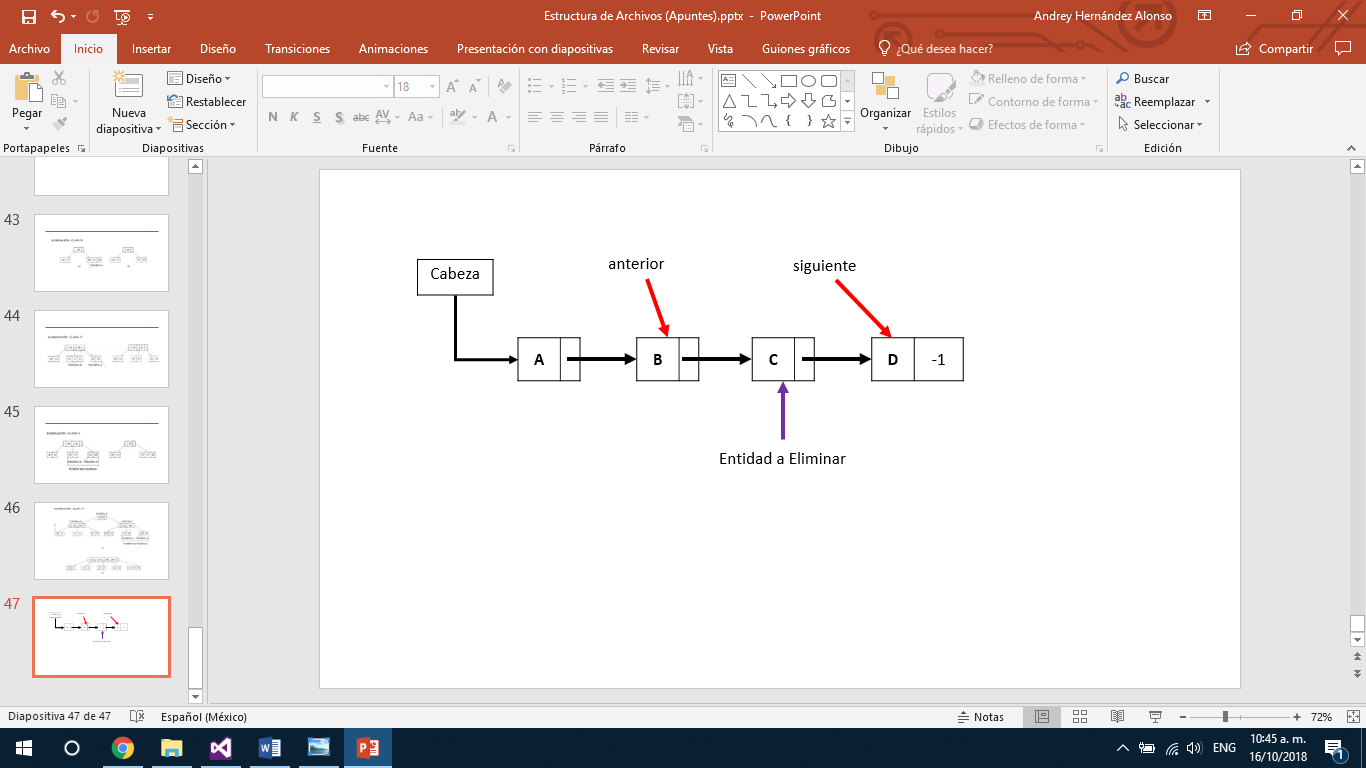
1. Leer la Cabeza del Archivo
2. Buscar en la lista la entidad que se desea eliminar
3. Obtener sus apuntadores anterior y siguiente
4. Enlazar su anterior ahora apunta a su siguiente

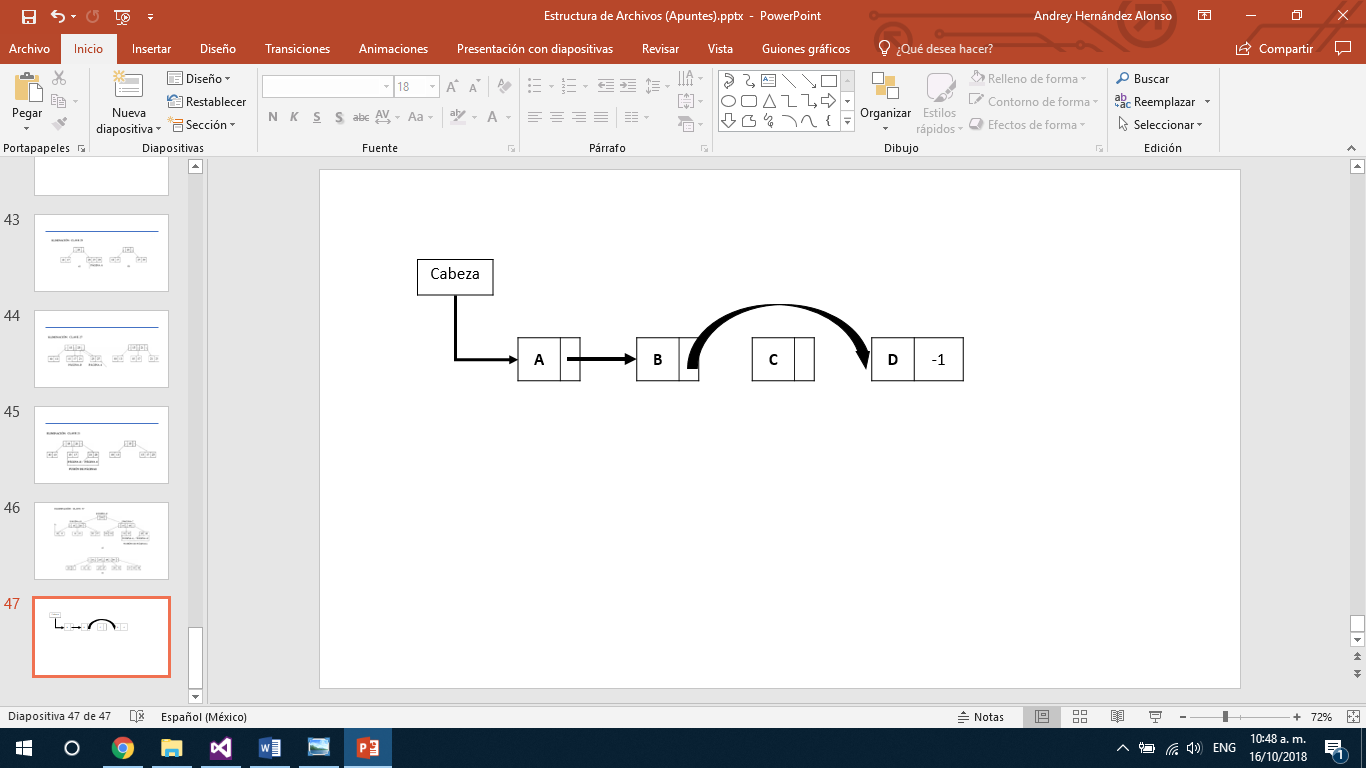


Ejemplo gráfico de Eliminación:



Eliminar ( C )





**Algoritmo: Modificación de Entidad**

Una modificación es un poco más compleja ya que requiere de una eliminación como de una inserción por lo que se necesita guardar en auxiliares las direcciones a modificar.

Consideraciones al modificar:

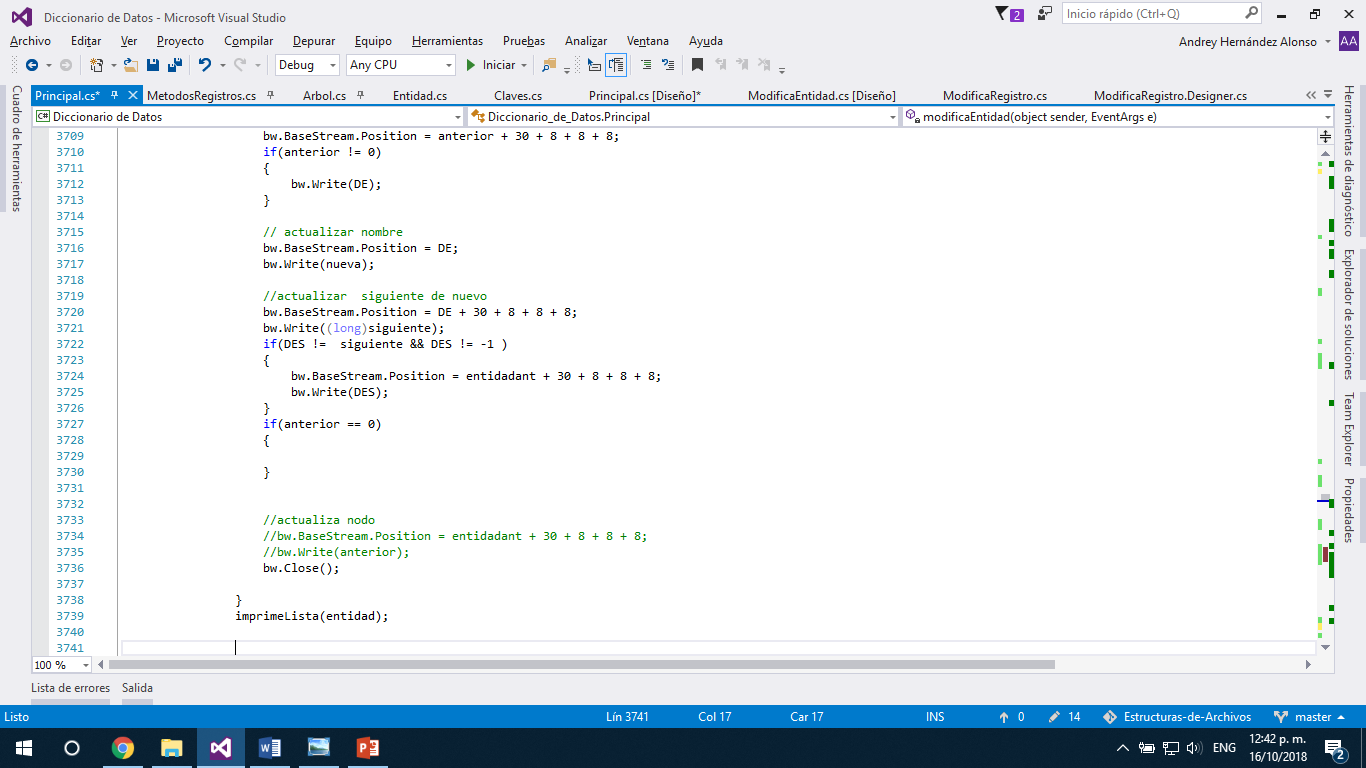
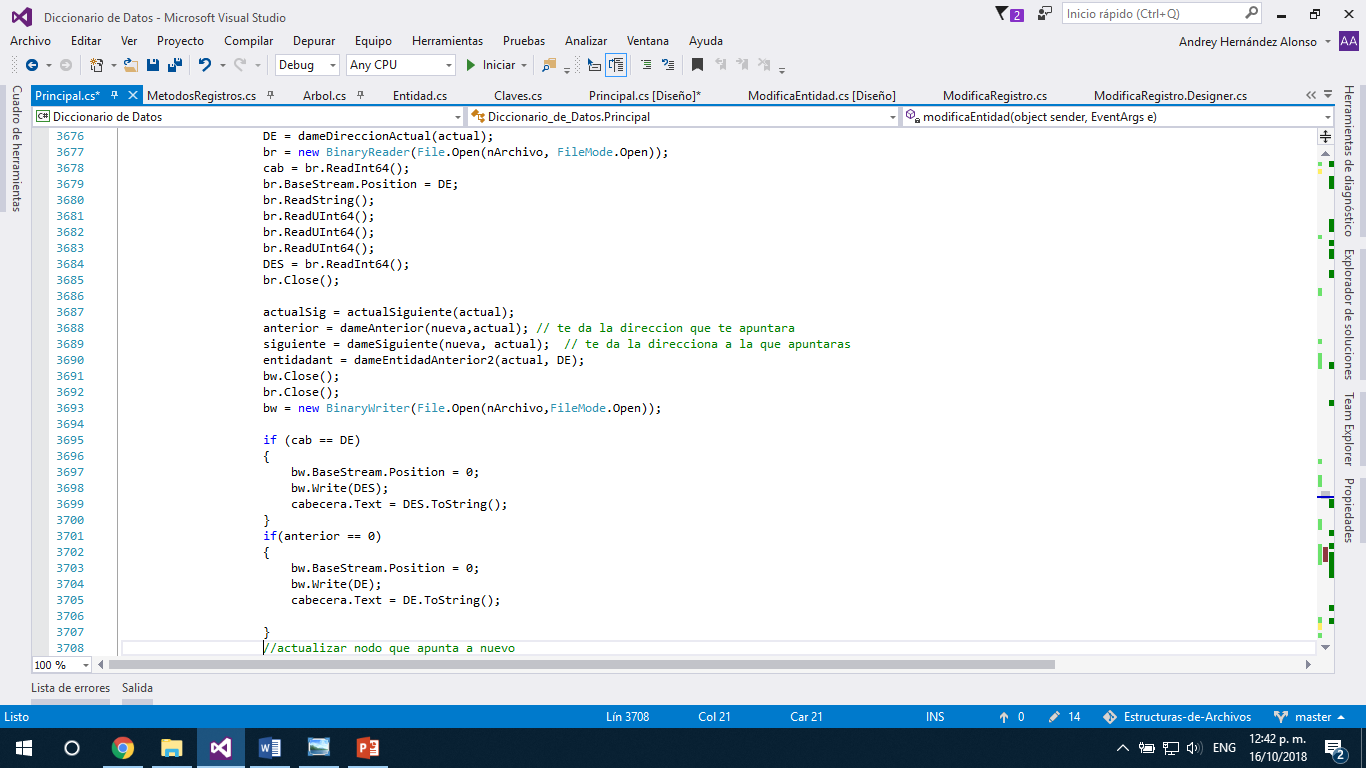
* Apuntador que tenga la dirección anterior de la Entidad a modificar
* Apuntador que tenga la dirección siguiente de la Entidad a modificar
* Apuntador que tenga la nueva dirección anterior
* Apuntador que tenga la nueva dirección siguiente
* Apuntador de la entidad a modificar

Para la modificación de la entidad:

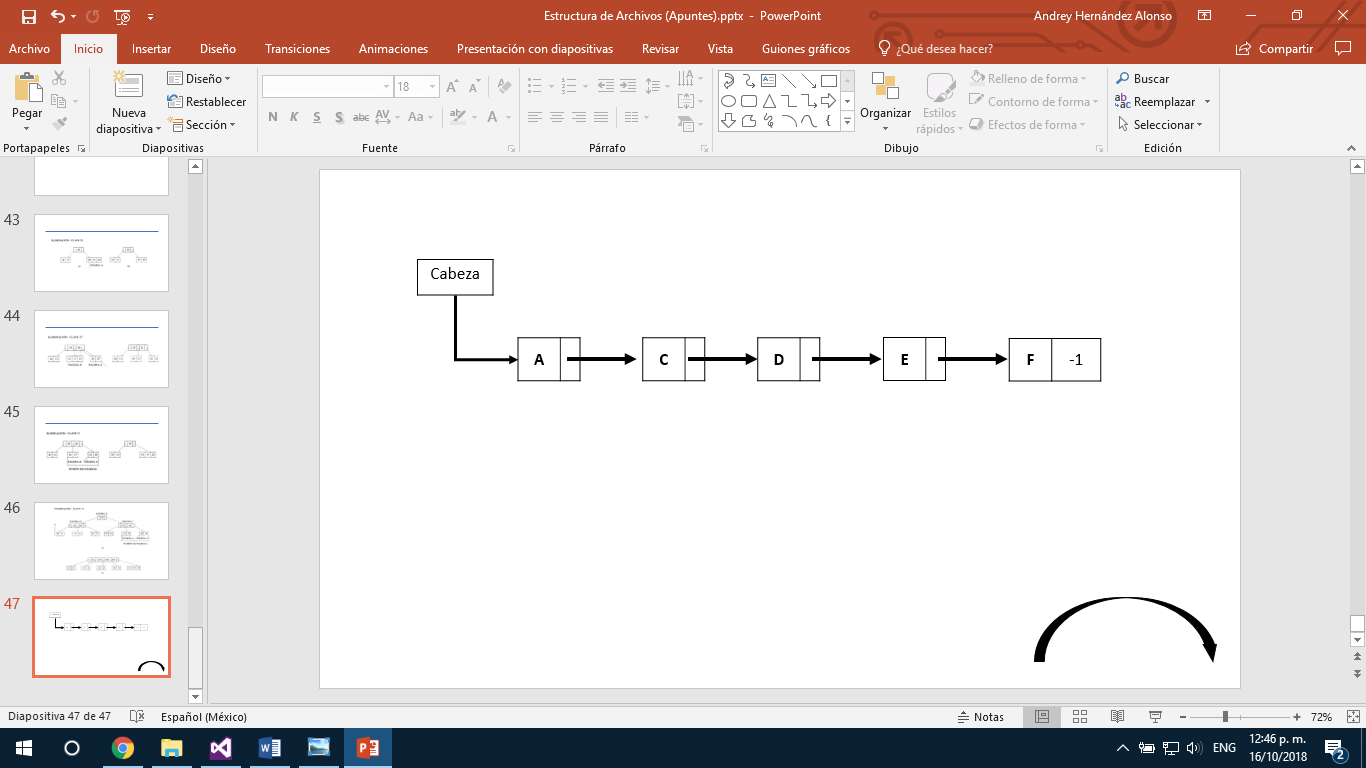
1. Obtener la dirección de la entidad
2. Asignar nuevo nombre
3. Obtener su apuntador anterior
4. Obtener su apuntador siguiente
5. Buscar en su nueva posición (ordenado)
6. Asignar a la nueva dirección la dirección de la entidad
7. La dirección anterior y la anterior siguiente se enlazan
8. Enlazar la nueva entidad con la siguiente dirección de la búsqueda

Casos Importantes al modificar:

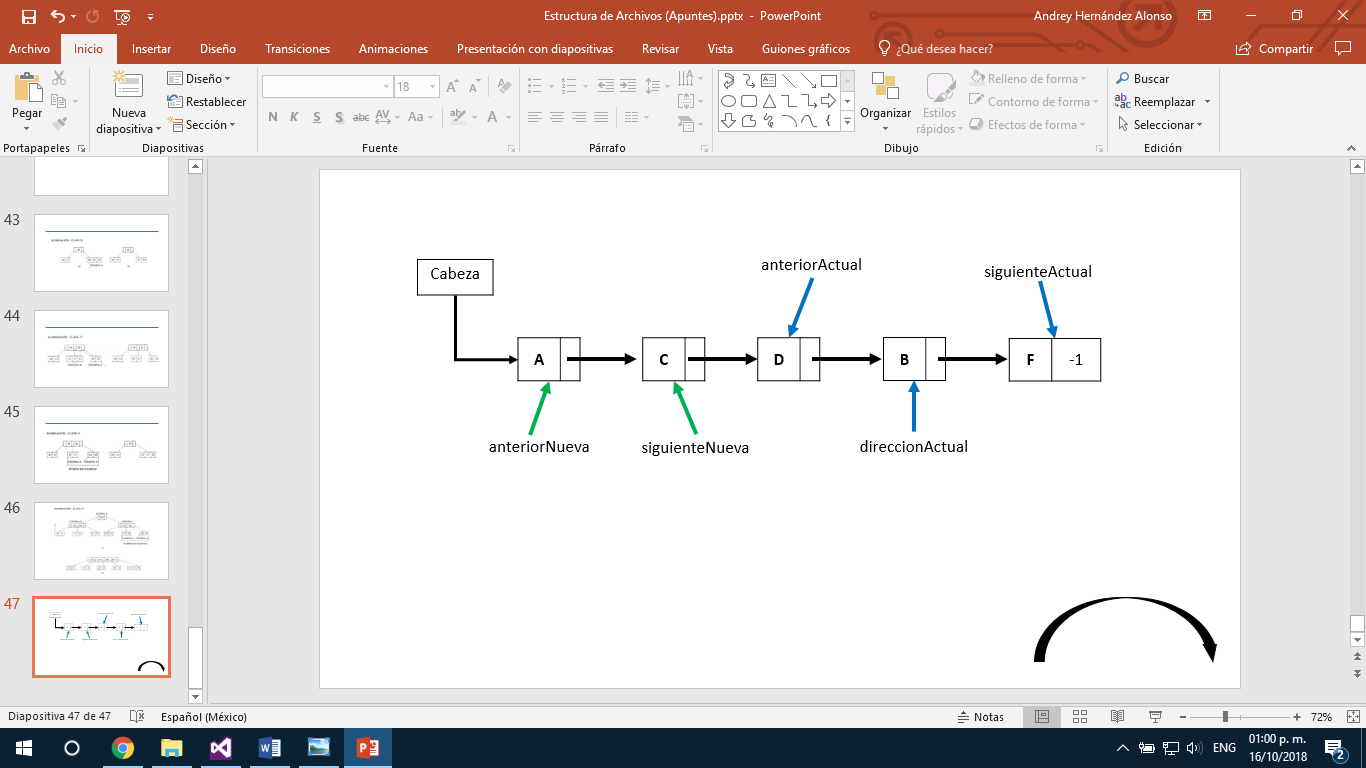
* Si al modificar es la primera entidad
  + Si se modifica a la última entidad
  + Si se modifica a cualquier posición
* Si al modificar es la última entidad
  + Si se modifica a la primera entidad
  + Si se modifica a cualquier posición
* Si al modificar queda en la misma posición

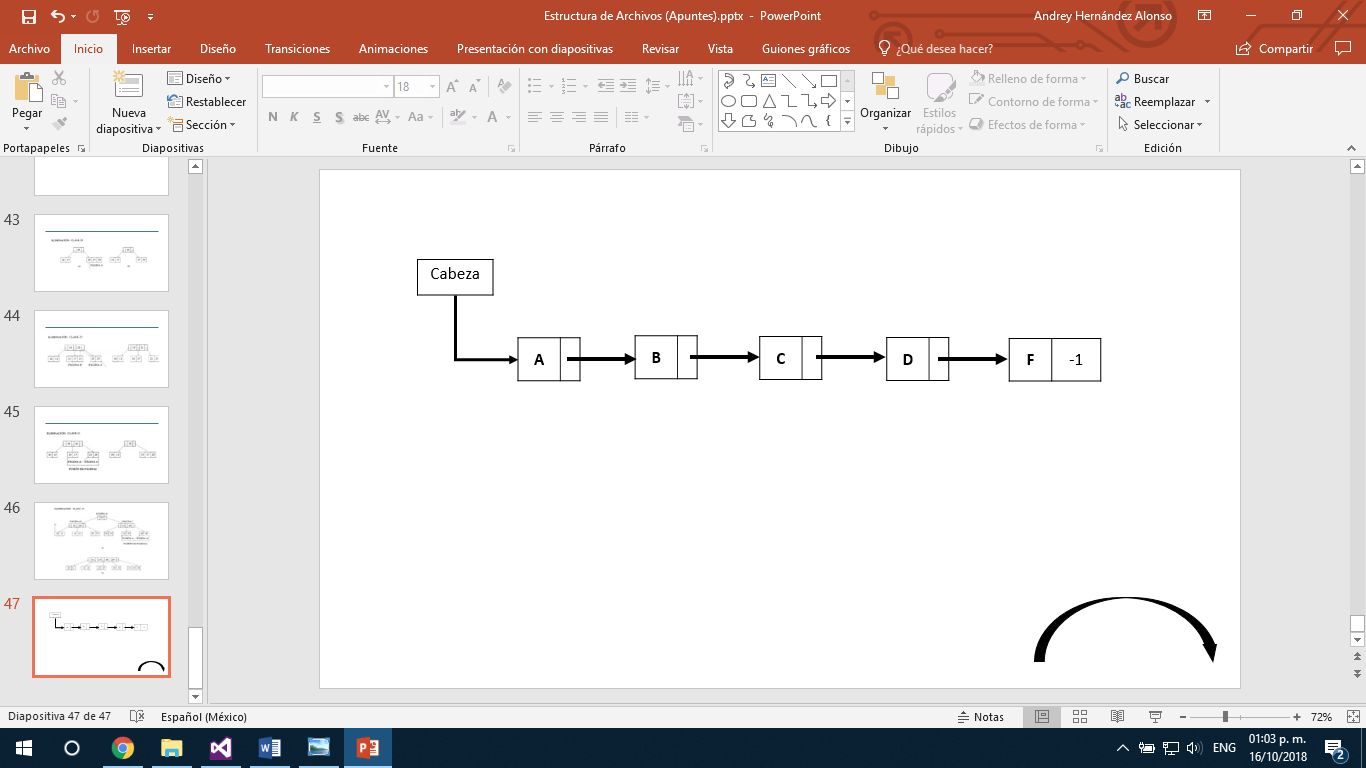


Ejemplo gráfico de Modificación:

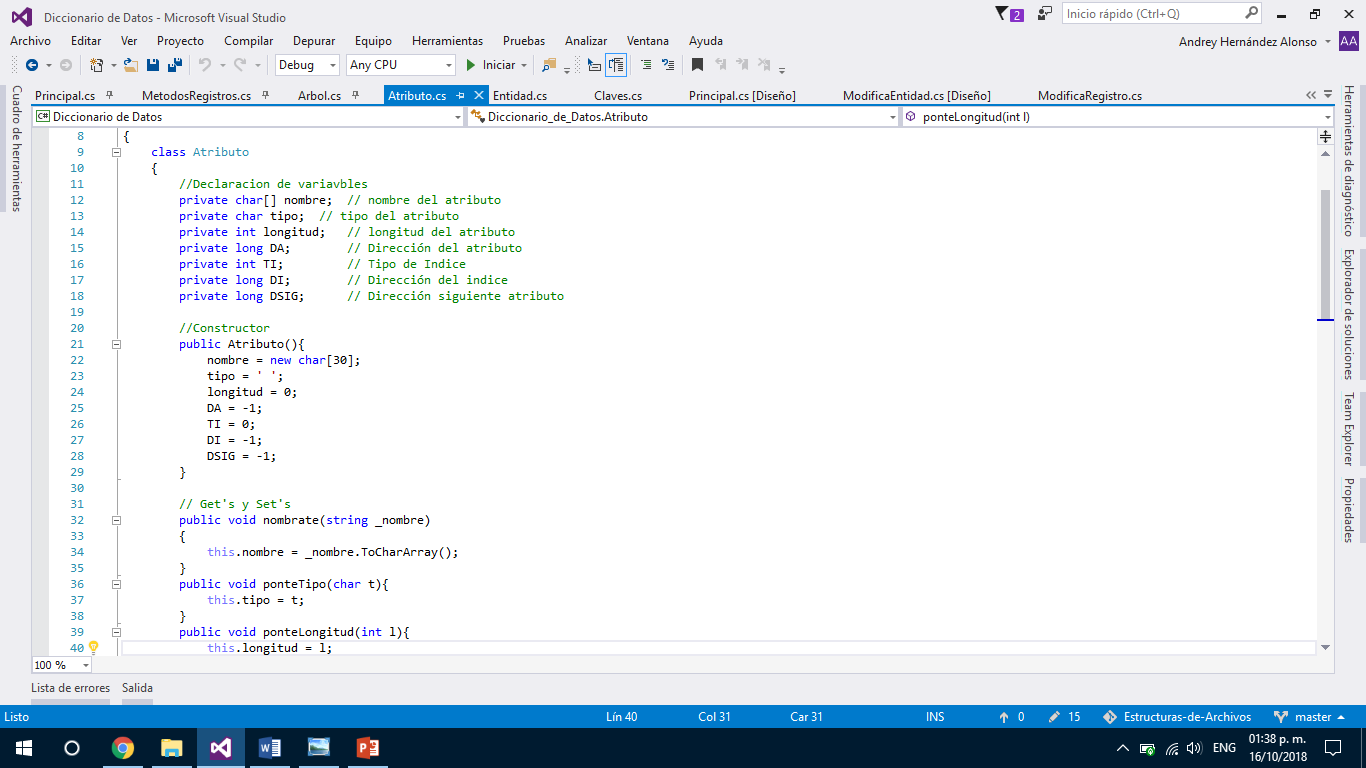


Modificar ( E 🡪 B)





Para poder controlar los Atributos , se utiliza una **Clase Atributo**



**METADATOS DEL ATRIBUTO**

**Nombre**

Identificador propio del atributo

**Tipo de dato**

Se utilizará Entero (E) y Carácter (C)

**Longitud de tipo de dato**

Representa la longitud en bytes del tipo de dato

**Dirección del atributo**

Representa la dirección de archivo en donde se almacena el atributo

**Tipo de Índice**

Identifica el tipo de estructura de datos que se utiliza para representar el índice

* 0 🡪 sin tipo de índice
* 1 🡪 clave de búsqueda
* 2 🡪 índice primario
* 3 🡪 índice secundario
* 4 🡪 Árbol B+

**Dirección de índice**

Representa la dirección de archivo en donde se comienza a almacenar la estructura de datos utilizada para representar el índice

**Dirección del siguiente atributo**

Representa la dirección del siguiente atributo que define a la misma entidad.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre | Tipo | Longitud | DA | TI | DI | DSIG |

|  |  |  |
| --- | --- | --- |
| NOMBRE | SIMBOLOGÍA | METODO |
| Nombre | Nombre | br.ReadString() |
| Tipo | Tipo | br.ReadChar() |
| Longitud | Longitud | br.ReadInt32() |
| Dirección Atributo | DA | br.ReadInt64() |
| Tipo de Índice | TI | br.ReadInt32() |
| Dirección Índice | DI | br.ReadInt64() |
| Dirección Siguiente | DSIG | br.ReadInt64() |

**Algoritmo: Inserción Atributo**