

MATERIA: ALGORITMOS Y ESTRUCTURAS DE DATOS.

SEMANA: 3

TEMAS SEMANA 3:

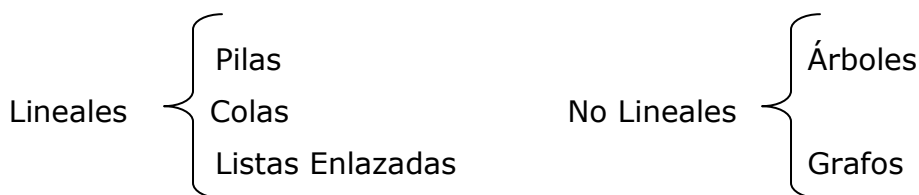
- a) Estructuras Dinámicas de datos**
- b) Diferencia**
- c) Apuntador, nodo**
- d) Listas Enlazadas**

- **¿Cuál es la diferencia entre una estructura dinámica y una estructura estática?**

La principal ventaja de manejar una estructura dinámica es que se pueden adquirir posiciones de memoria a medida que se necesitan, y liberarlas cuando ya no se requieran. Al contrario que una estructura estática como los arreglos, que contienen espacios para almacenar un número fijo de elementos, una estructura dinámica de datos se amplía y contrae durante la ejecución del programa.

Las estructuras dinámicas de datos se utilizan para almacenamiento de datos del mundo real que están cambiando constantemente. Un ejemplo típico de una estructura dinámica de datos es la lista de pasajeros de una línea aérea.

Las estructuras dinámicas de datos se pueden dividir en dos grandes grupos:



- **¿Cómo se declara y crea un puntero?**

Puntero o apuntador. El tipo de variable que almacena una dirección se denomina puntero o apuntador (un puntero es la dirección de una variable).

Una variable puntero es una variable que contiene direcciones de otras variables. Todas las variables contienen valores de datos, por el contrario las variables puntero contienen valores que son direcciones de memoria donde se almacenan datos.

Cuando una variable se declara se asocian tres atributos fundamentales con la misma: su nombre, su tipo y su dirección de memoria.

Ejemplo:

El siguiente ejemplo muestra la sintaxis del lenguaje de programación C# para la declaración y creación de un puntero.

```
static void Main(string[] args)
{
unsafe           // se debe declarar este bloque para poder utilizar punteros
{
int entero = 5;   // declaración de variable numérica entera, solamente
                    almacenará números
int* puntero;    /* declaración de un puntero que almacenará direcciones
                    de memoria de variables numéricas enteras */
    // se inicializa el puntero, que apuntará a un valor entero
puntero = &entero; /* puntero almacenará la dirección donde está
                    alojada la variable denominada entero */
    // asignación de un puntero entero a una variable de tipo entero
int temporal = *puntero; /* la variable temporal almacenará el
                    el valor de la variable a la que apunta
                    la variable llamada puntero */
    // asignación de un valor en un puntero
*puntero = 3;     // ahora, a donde apunta la variable llamada puntero
                    se le asignará el valor de 3
    Console.WriteLine("'puntero' contiene {0:d}", *puntero);
    Console.WriteLine("y la variable 'entero' contiene lo mismo que el
                    'puntero', {0:d}", entero);
} // fuera de este bloque no existen las variables tipo puntero
Console.ReadLine(); }
```

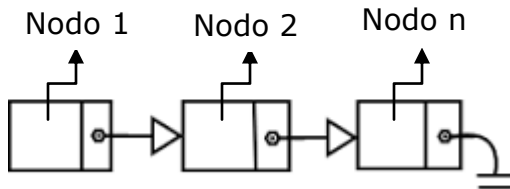
- ¿Para qué sirve un nodo?

Nodo es un registro con dos campos, uno de ellos contiene los componentes y se le denominará datos (o data) y el otro es un valor que señala al siguiente nodo y se le denominará enlace (o next).

El campo dato debe entenderse conceptualmente, es decir en una implementación concreta pueden ser varios los campos que se consideran como datos.



Estructura de un nodo



Lista enlazada

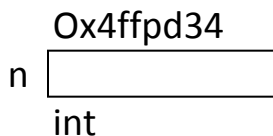
Tal enlace que se usa para acceder a un nodo particular, se conoce como apuntador.

Cuando una variable se declara, se asocian tres atributos fundamentales con la misma: su nombre, su tipo y su dirección

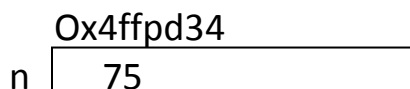
Ejemplo:

```

int n           // asocia el nombre n, el tipo int y la
                  // dirección de alguna posición de memoria
                  // donde se almacena el valor de n
    
```



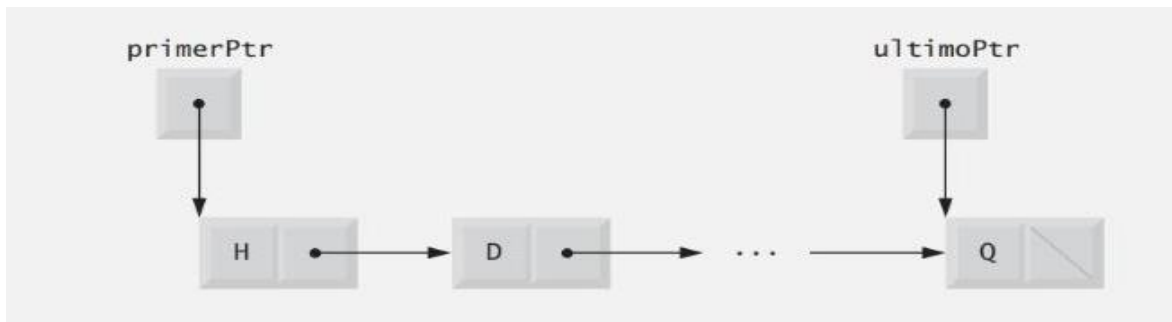
Esta caja representa la posición de almacenamiento en memoria. El nombre de la variable está a la izquierda de la caja, la dirección de la variable esta encima de la caja y el tipo de variables está debajo de la caja. Si el valor de la variable se conoce, se representa en el interior de la caja;



- ¿Cómo se declara y crea una lista enlazada?

Una lista enlazada es una colección lineal (una secuencia) de objetos llamados generalmente **nodos** que están conectados por enlaces de referencia; es por ello que se utiliza el término lista "enlazada". En una lista enlazada cada elemento contiene la posición o dirección del siguiente elemento de la lista.

Cada elemento de la lista enlazada debe tener al menos dos campos; un campo que tiene el valor del elemento y un campo enlace (link) que contiene la posición del siguiente elemento, es decir, la conexión o enlace de los elementos de una lista se hace por medio de los campos enlaces. Por convención, la referencia de enlace en el último nodo de una lista se establece en **null** para indicar el final de la lista.



Representación gráfica de una lista enlazada.

El siguiente ejemplo muestra la sintaxis del lenguaje de programación C# para la declaración y creación de una lista enlazada. Lo que hace es crear un objeto lista que contiene n elementos de tipo entero, y almacena la referencia a la lista en la variable **num**:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace listaenlazada
{
```

```
    class Program
```

```
    {
```

```
        static void Main(string[] args)
```

```
        {
```

```
            ListaEnlazada lista = new ListaEnlazada();
            Console.WriteLine("Ingrese los numeros que desee para la lista Enlazada:
                               (pulse dos veces enter para finalizar)");
```

```
            string num;
```

```
            do
```

```
            {
```

```
                num = Console.ReadLine();
```

```
                if (!String.IsNullOrEmpty(num))
```

```
                {
```

```
                    lista.Insertar(int.Parse(num));
```

```
                }
```

```
            } while (!String.IsNullOrEmpty(num));
```

Bibliotecas del programa

Declaración de una lista enlazada

Csptura de los valores de la lista hasta encontrar el valor null

```

        lista.Mostrar();
        Console.ReadKey();
    }
}
class Nodo
{
    public int valor;
    public Nodo sig;
    public Nodo(int v, Nodo s)
    {
        valor = v;
        sig = s;
    }
}
class ListaEnlazada
{
    Nodo primero;
    Nodo actual;
    public ListaEnlazada()
    {
    }
    public void Insertar(int v)
    {
        Nodo anterior;
        if (ListaVacía() || primero.valor > v)
        {
            primero = new Nodo(v, primero);
        }
        else
        {
            anterior = primero;
            while (anterior.sig != null && anterior.sig.valor <= v)
                anterior = anterior.sig;
            anterior.sig = new Nodo(v, anterior.sig);
        }
    }
    public void Mostrar()
    {
        Nodo aux;
        aux = primero;
        while (aux != null)
        {
            Console.Write(aux.valor + "->");
            aux = aux.sig;
        }
        Console.Write(" null");
        Console.WriteLine();
    }
}

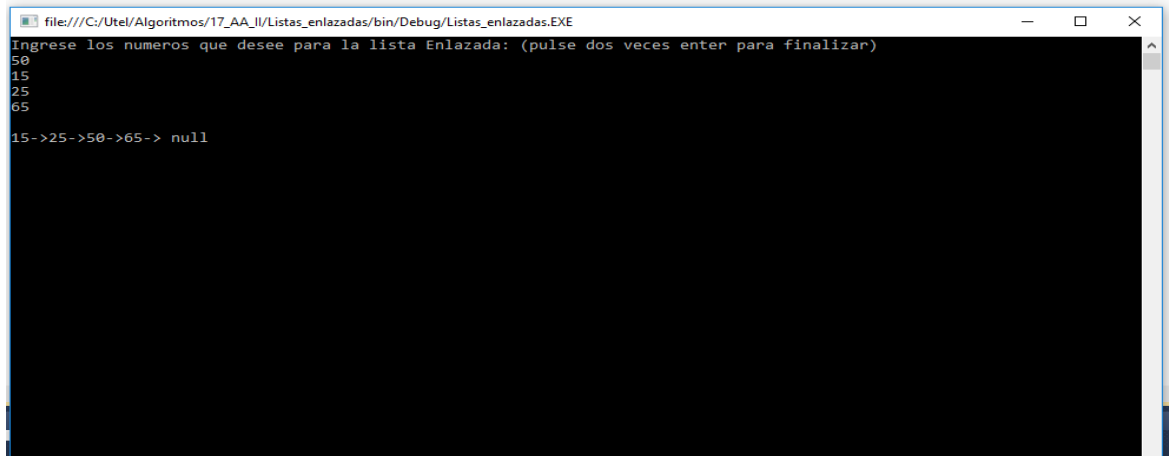
```

Declaración de los valores
de cada nodo de la lista

Asignación de valores
para los nodos de la lista

Impresión en pantalla
de los valores de la lista

Salida del programa:



The screenshot shows a Windows command prompt window titled "file:///C:/Utel/Algoritmos/17_AA_II/Listas_enlazadas/bin/Debug/Listas_enlazadas.EXE". The prompt displays the instruction "Ingrese los numeros que desee para la lista Enlazada: (pulse dos veces enter para finalizar)". Below this, the numbers 50, 15, 25, and 65 are listed vertically. The user input "15->25->50->65-> null" is shown at the bottom of the window.

```
file:///C:/Utel/Algoritmos/17_AA_II/Listas_enlazadas/bin/Debug/Listas_enlazadas.EXE
Ingrese los numeros que desee para la lista Enlazada: (pulse dos veces enter para finalizar)
50
15
25
65
15->25->50->65-> null
```