



# ESTRUCTURA DE DATOS

## Listas Ligadas

# LISTAS

Por su forma de almacenamiento, la lista lineal se puede implementar en una de las siguientes disposiciones:

- Secuencial
- Enlazada



# LISTA SECUENCIAL

Una de las formas más simples de implementación de este TDA es usando un arreglo unidimensional.

Todos los elementos de la lista se almacenan en posiciones de memoria consecutivas. Se habla de *disposición secuencial* en la memoria de la computadora.

A la lista se le conoce como *lista secuencial*.



# VENTAJAS Y DESVENTAJAS

## Ventajas

Con esta disposición se accede a cualquier elemento de la estructura de datos en tiempo constante.

## Desventajas

Al asignar el arreglo en tiempo de compilación debe establecerse un límite *a priori* sobre el número de elementos que pueden ser almacenados en las listas.

Para inserciones y eliminaciones frecuentes hay que hacer corrimientos costosos.



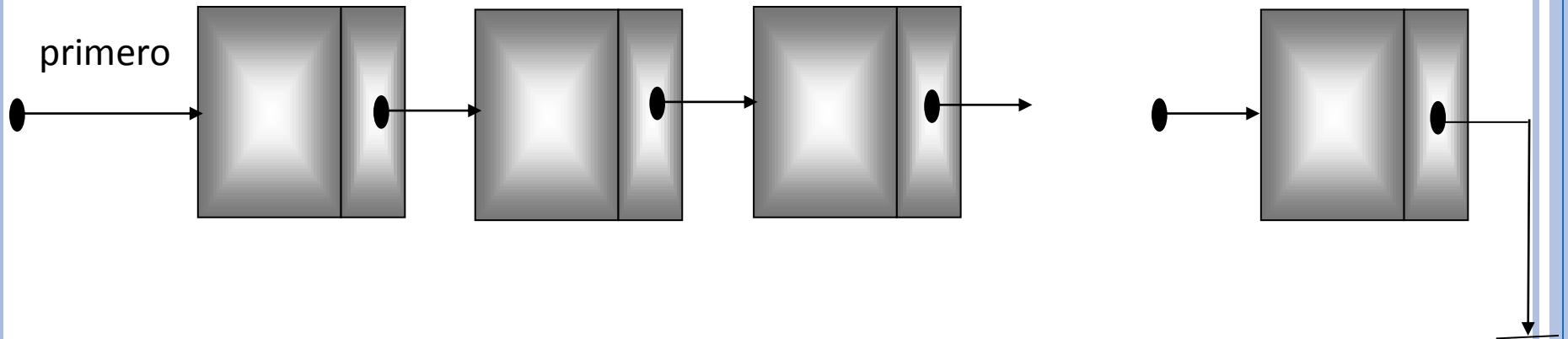
## LISTA ENLAZADA

En una lista enlazada se asigna memoria para el almacenar los elementos de la lista conforme se va necesitando, es decir a medida que se añaden o insertan los elementos, y se conectan los elementos de la lista con punteros.

La memoria es liberada cuando ya no se necesita más un elemento en la lista.

Esquemáticamente una lista enlazada se representa por una secuencia de nodos conectados por enlaces.





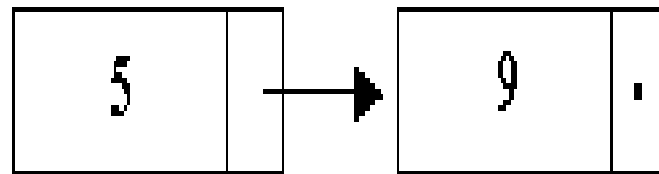
Cada nodo está conectado al siguiente por un solo enlace, a esta estructura de datos se llama *lista simplemente enlazada*.



Un nodo de una lista simplemente enlazada contiene dos campos: *datos* (contiene un elemento de la lista) y *siguiente* (almacena un enlace al siguiente nodo de la lista).

El campo *siguiente* del último nodo contiene un símbolo especial que indica el final de la lista.

Se accede a la lista por medio de un apuntador al primer elemento y solo se puede recorrer la lista en un sentido, del primer nodo al último nodo.



# VENTAJAS Y DESVENTAJAS

## Ventajas

No es preciso conocer la cantidad de elementos en tiempo de compilación.

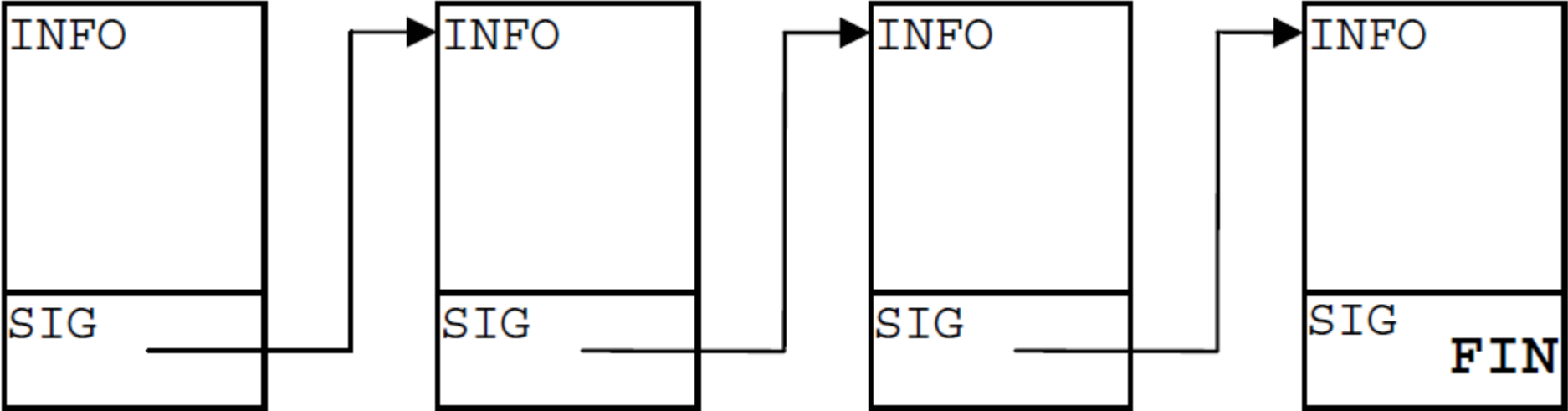
Ni las inserciones ni las eliminaciones implican realizar corrimientos de los elementos de la lista.

## Desventajas

No permite el acceso directo a un elemento arbitrario de la lista. Para acceder al *i-ésimo* elemento, debemos recorrer la lista, comenzando por el primer nodo, hasta llegar al elemento deseado.







## FUNCIONES PARA MANEJAR UNA LISTA

Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.

Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.

Buscar: busca un elemento en la lista.

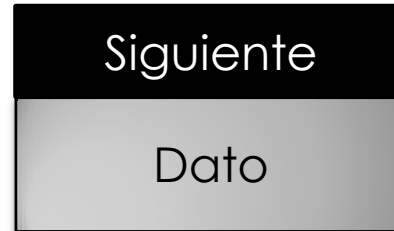
Localizar: obtiene la posición del nodo en la lista.

Vaciar: borra todos los elementos de la lista



## SINTAXIS

```
struct lista {  
    int dato;  
    lista *siguiente;  
};
```



```
struct _agenda {  
    char nombre[20];  
    char telefono[12];  
    struct _agenda *siguiente;  
};
```

