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.

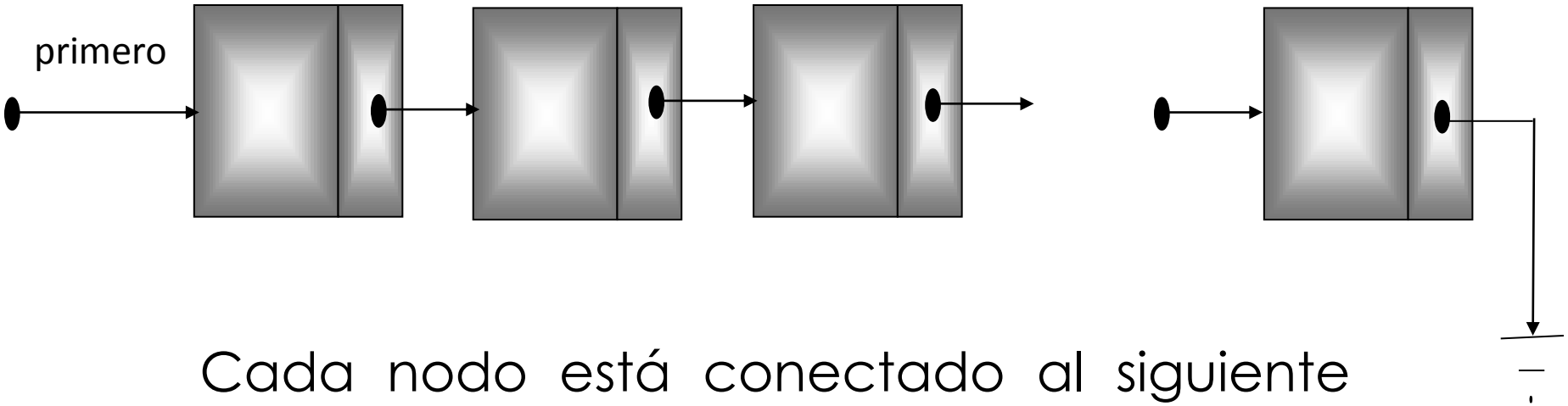
Para inserciones y eliminaciones frecuentes hay que hacer corrimientos costosos.

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.

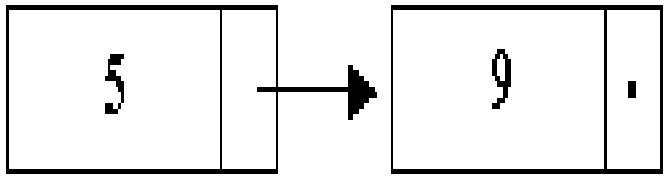


Cada nodo está conectado al siguiente por un solo enlace, a esta estructura de datos se llama *lista simplemente enlazada*.

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

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

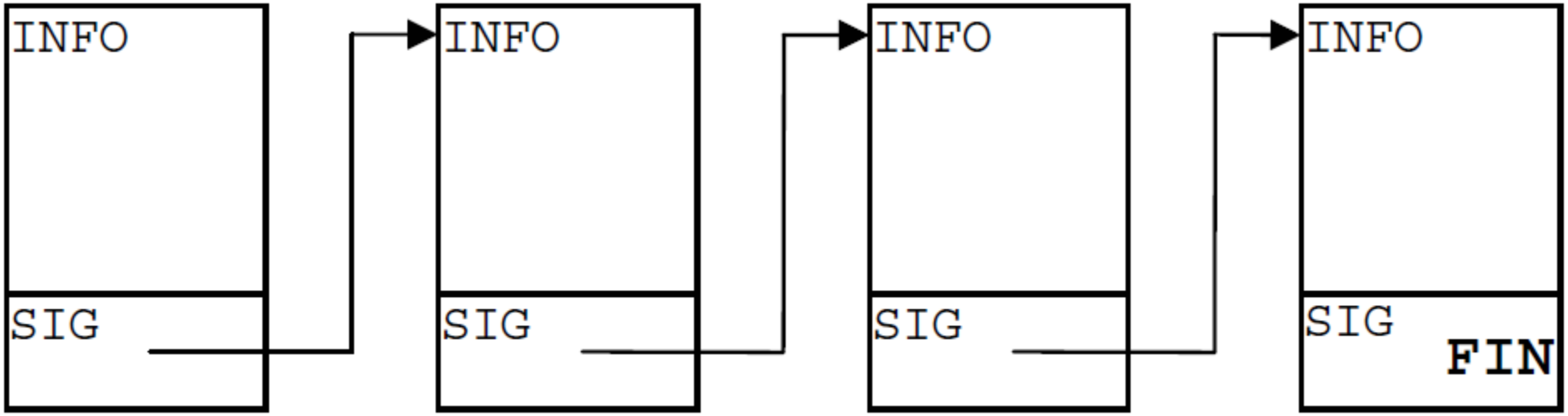
No permite el acceso directo a un elemento arbitrario de la lista. Para acceder al *i-ésimo* elemento, debemos recorrer la lista, comenzando por el primer nodo, hasta llegar al elemento deseado.

CREAINICIO(P)

{Este algoritmo crea una lista, agregando cada nuevo nodo al inicio de la misma}

{ P y Q son variables de tipo puntero. P apuntará al inicio de la lista}

1. CREA (P) {Crea el primer nodo de la lista}
2. Leer P->INFORMACIÓN
3. Hacer P->LIGA=NIL
4. Escribir "¿ Deseas ingresar más datos a la liga? 1: SI 2: NO"
5. Leer RES
6. Mientras (RES=1)Repetir
    CREA (Q)
    Leer Q->INFORMACIÓN
    Hacer Q->LIGA= P y P = Q
    Escribir "¿ Deseas ingresar más datos a la liga? 1: SI 2: NO"
    Leer RES
7-. (Fin del ciclo del paso 6)


INFO
SIG
INFO
SIG
INFO
SIG
INFO
SIG
FIN

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;
};
```