LISTAS DINÁMICAS

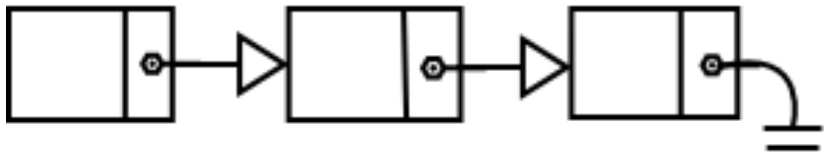
Emanuel García Pérez

March 25, 2014

¿Qué es una lista enlazada?

Una lista enlazada es una colección o secuencia de elementos dispuestos uno detrás de otro, en la que cada elemento se conecta al siguiente elemento por medio un **“enlace”** o **“puntero”**.

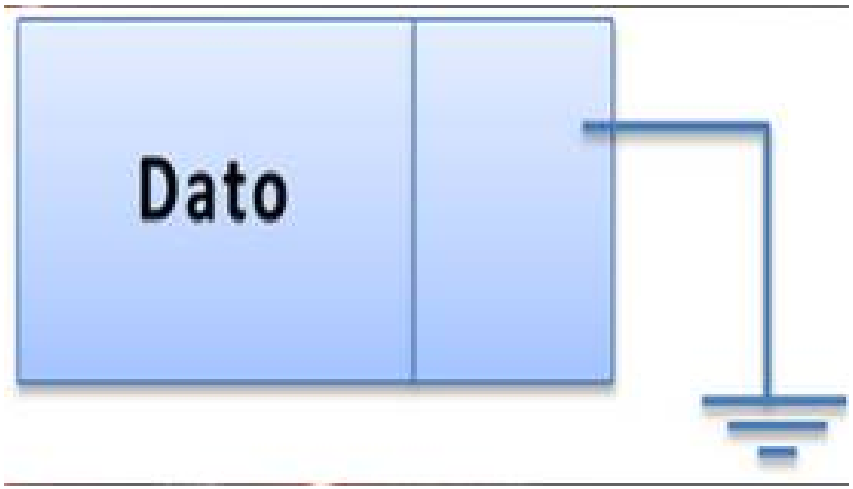
Estructura de un nodo



Lista enlazada

Constitución de una lista enlazada

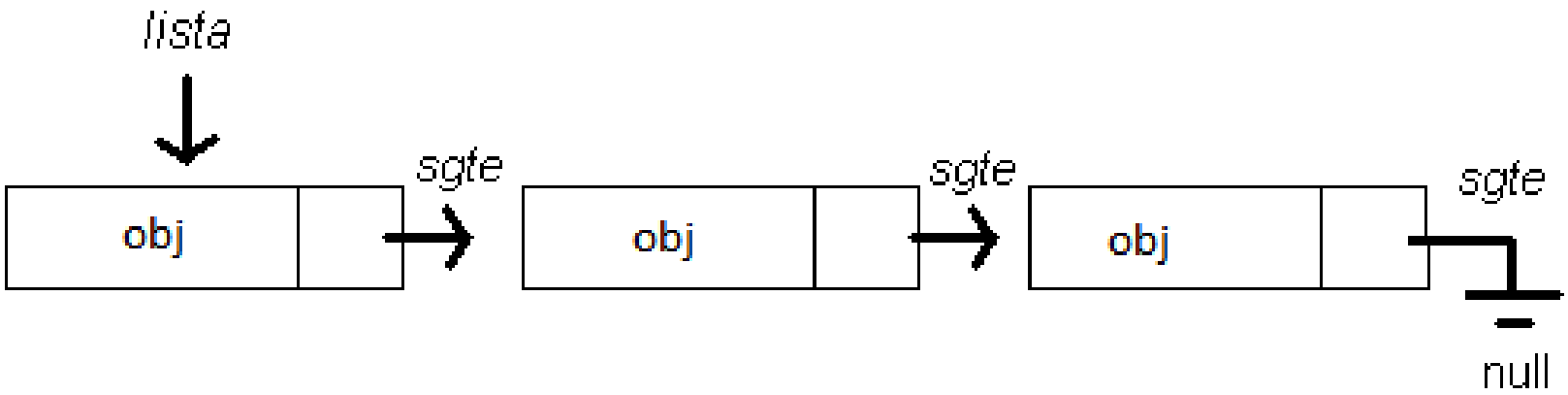
La idea básica consiste en construir una lista cuyos elementos llamados nodos se componen de dos partes o campos: el primer campo contiene la **información** o datos del elemento, y por tanto es de tipo genérico (Dato, TipoElemento, Info), y el segundo campo es un **puntero** (enlace, conector, siguiente) que apunta al siguiente elemento de la lista.



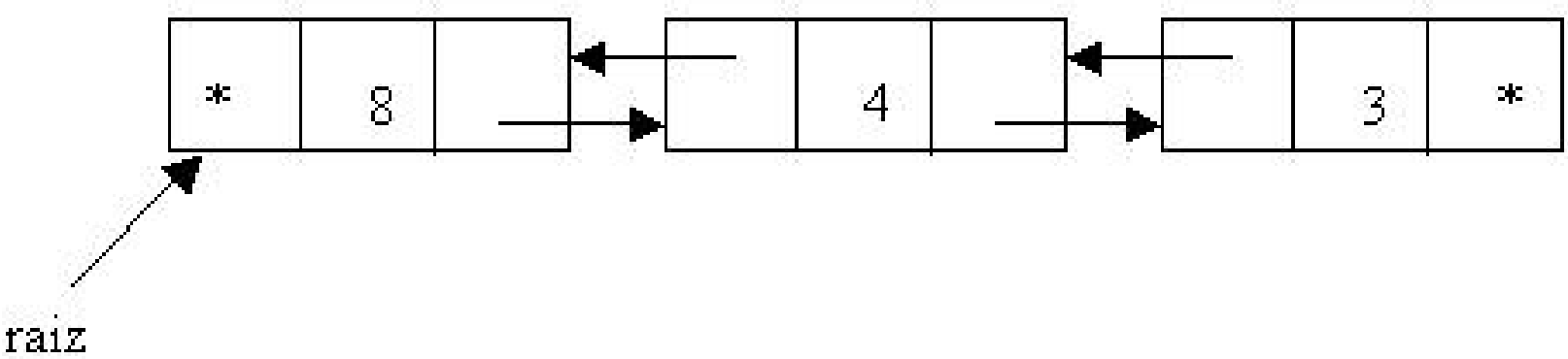
Clasificación de listas enlazadas

1. Lista Enlazada Simple
2. Lista Doblemente Enlazada
3. Listas Circular Simple
4. Lista Circular Doblemente Enlazada

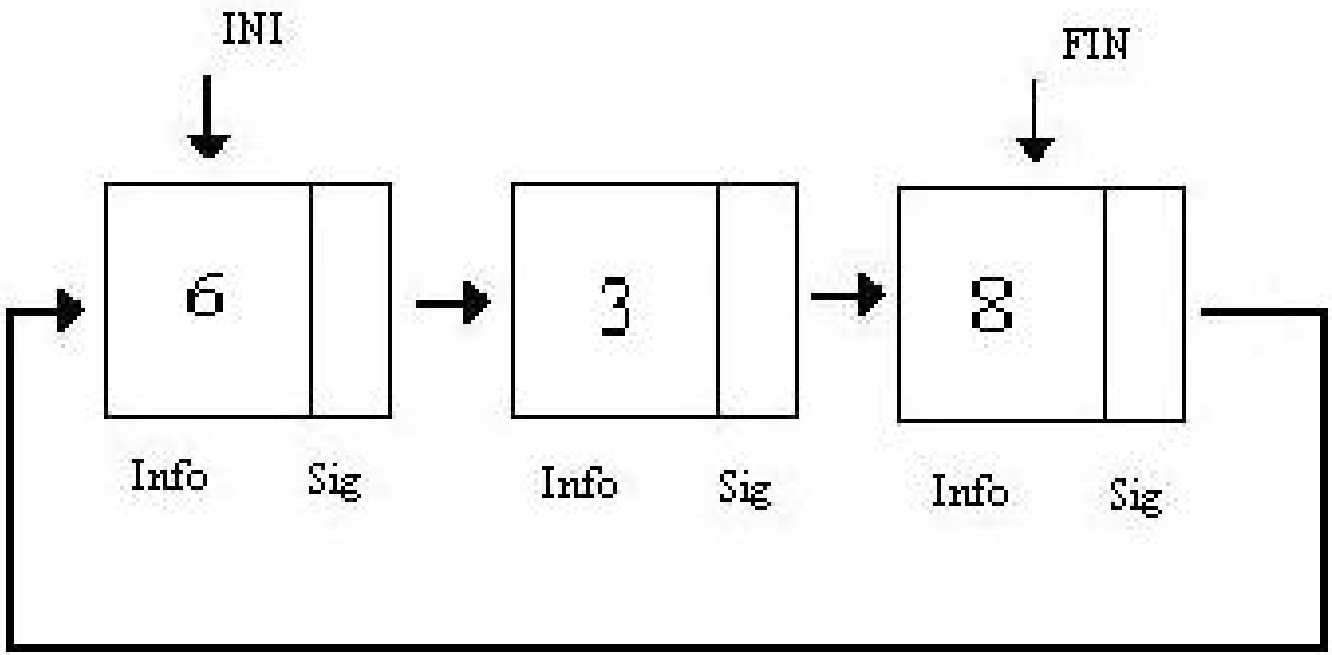
Lista Simple



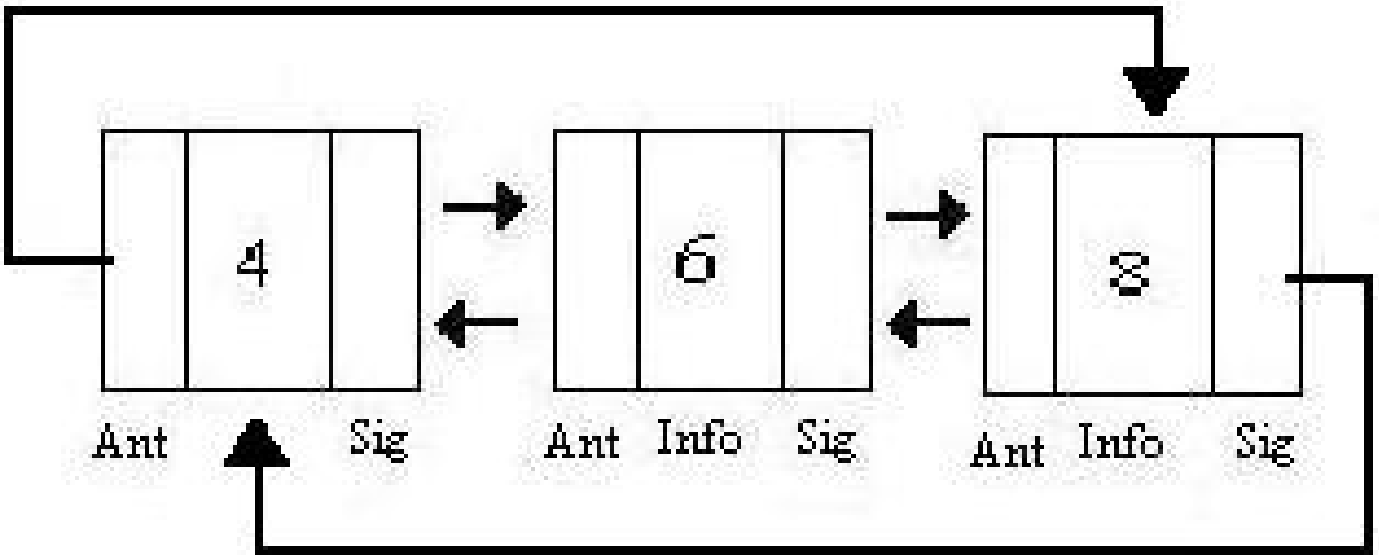
Lista Doble



Lista Circular Simple



Lista Circular Doble



Operaciones sobre listas enlazadas

- Inicialización
- Insertar elementos
  - Al inicio
  - En medio
  - Al final
- Eliminar elementos
  - Al inicio
  - En medio
  - Al final
- Buscar elementos
  - Hacia adelante
  - Hacia atrás
- Recorrer la lista
  - Hacia adelante
  - Hacia atrás

TDA's

- PILAS
- COLAS
- Tablas HASH
- Arboles
- Grafos

Definición del tipo de dato abstracto: Nodo

```
struct Nodo{     // Nodo tipo struct.
        int numero;
        struct Nodo* siguiente;
};

typedef struct Nodo{     // Nodo de tipo tsNODO.
        int numero;
        struct Nodo *siguiente;
}tsNODO;

struct Nodo* ptr_nodo;   // Tipo struct Nodo.
tsNODO *ptrNodo;         // Tipo tsNODO.
```

malloc()

```
void insercion(tsNODO **lista, int num, int num2){
 tsNODO *nodo1, *nodo2;
 nodo1 = (tsNODO*)malloc(sizeof(tsNODO));
 nodo2 = (struct Nodo*)malloc(sizeof(struct Nodo));
 nodo1->numero = num1; nodo2->numero = num2;
 nodo1->siguiente = NULL; nodo2->siguiente = NULL;
 if(*lista == NULL){
  *lista = nodo1; (*lista)->siguiente = nodo2;
 }else{
  (*lista)->siguiente = nodo1;
  nodo1->siguiente = nodo2;
  }
}
insercion(&Lista, 3, 7);
```

free()

```
int extraccion(tsNODO **lista){
 int num;
 tsNODO *nodo;
 if(*lista){
  nodo = *lista;
  *lista = (*lista)->siguiente;
  num = nodo->numero;
  free(nodo);
 }
 return num;
}

entero = extraccion(&Lista);
```

Dirección, Indirección y acceso a Miembro.

- &: Operador de dirección, da acceso a la dirección de la variable que estamos referenciando.
- *: Operador de indirección, da acceso al contenido de la variable que estamos referenciando.
- ->: Operador de acceso, nos permite acceder a un miembro determinado de una estructura que esta siendo referenciada a través de punteros.