LISTA LIGADA SIMPLE

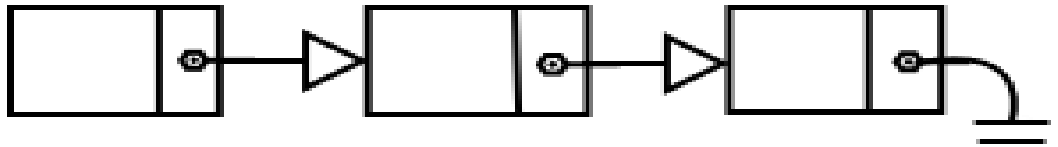
Otoño 2018

Otoño 2018

Lista ligada

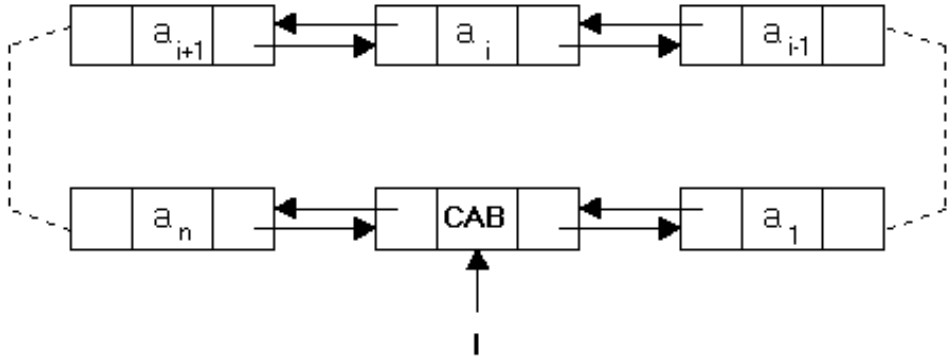
- Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior.

Estructura de un nodo

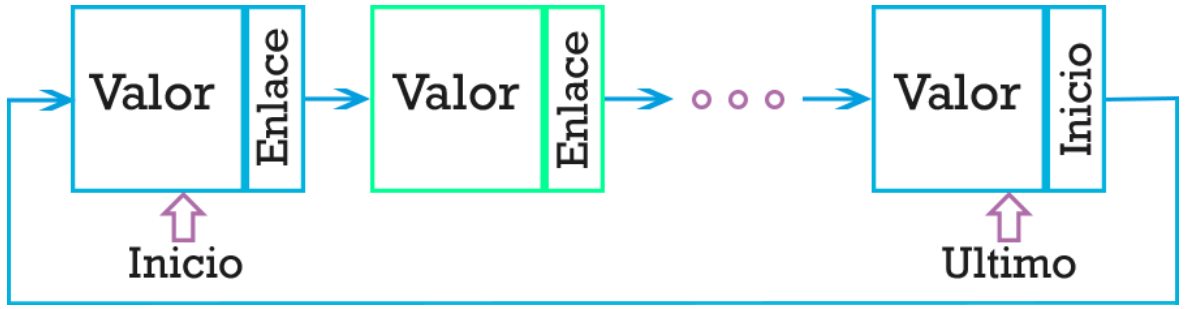


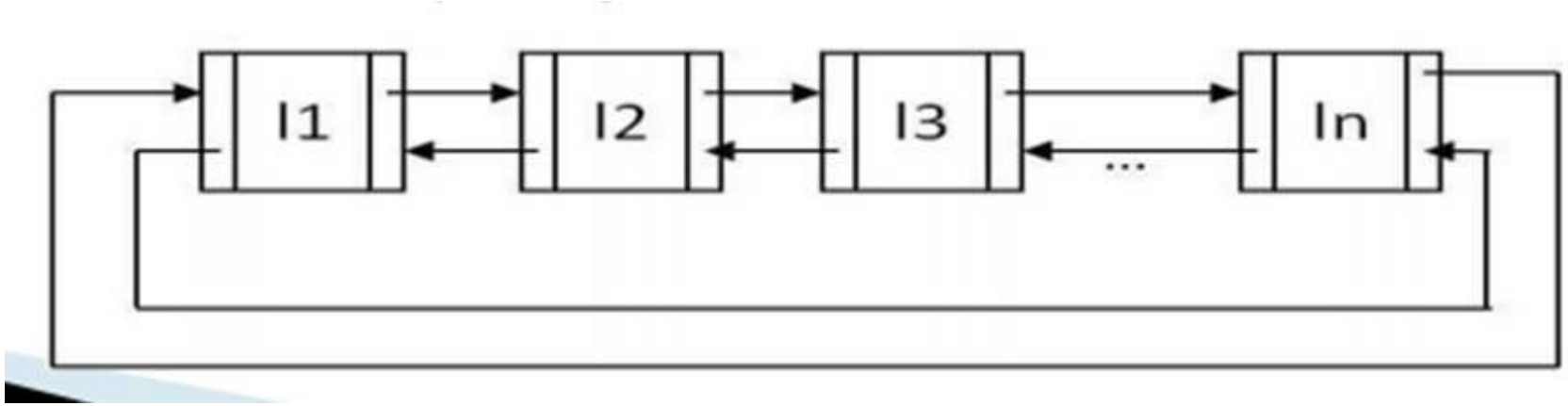
Lista enlazada

Tipo de listas ligadas



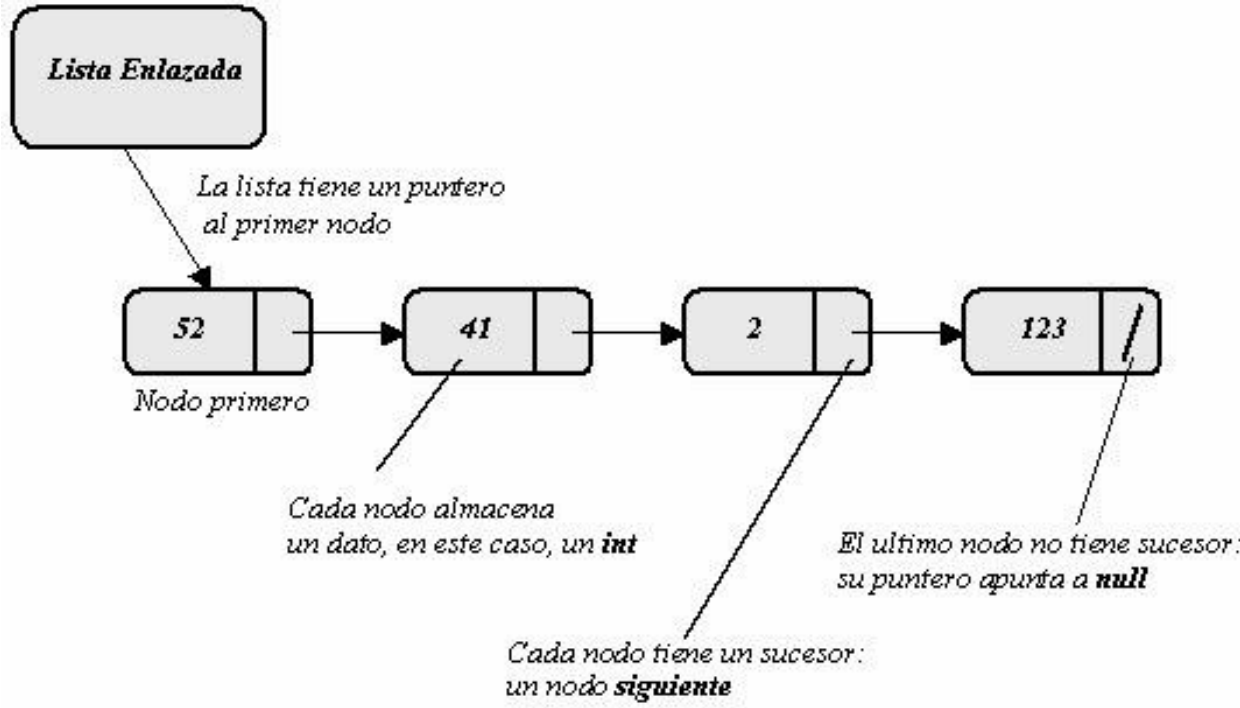
- Listas enlazadas simples
- Listas doblemente enlazadas
- Listas enlazadas circulares
- Listas enlazadas doblemente circulares.





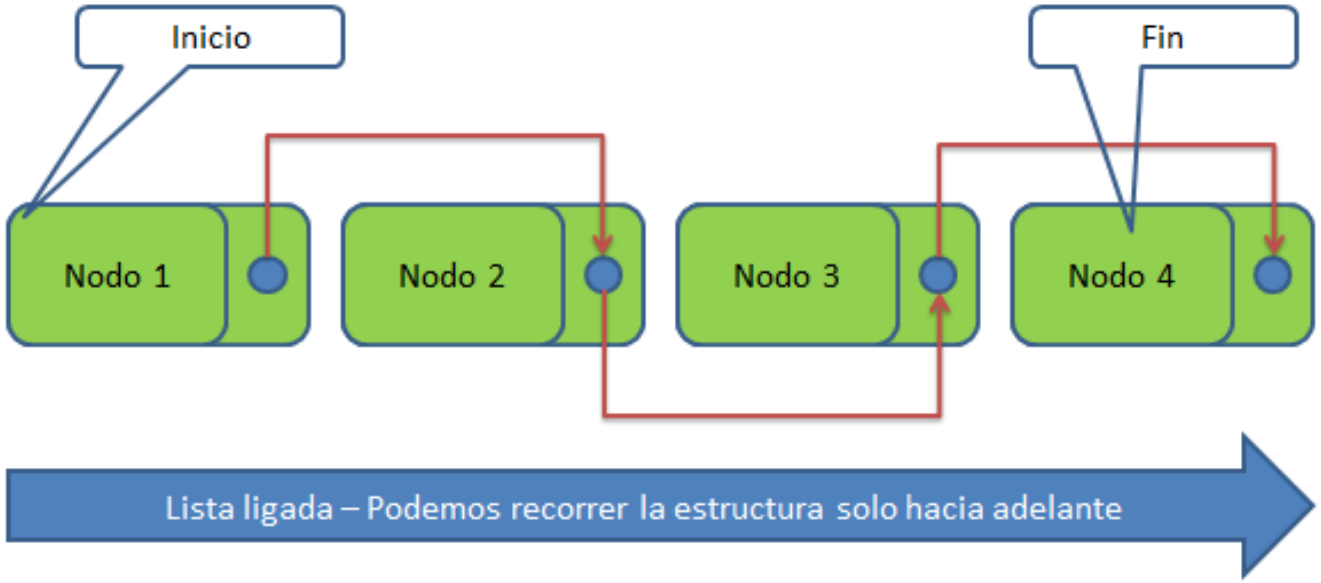
Lista ligada simple

- Es una lista enlazada de nodos
- Cada nodo tiene un único campo de enlace
- Una variable de referencia contiene una referencia al primer nodo
- Cada nodo (excepto el último) enlaza con el nodo siguiente
- El enlace del último nodo contiene NULL (final de la lista)



Características

- No es preciso conocer la cantidad de elementos en tiempo de compilación.
- Es un tipo de dato autorreferenciado porque contienen un puntero o enlace a otro dato del mismo tipo

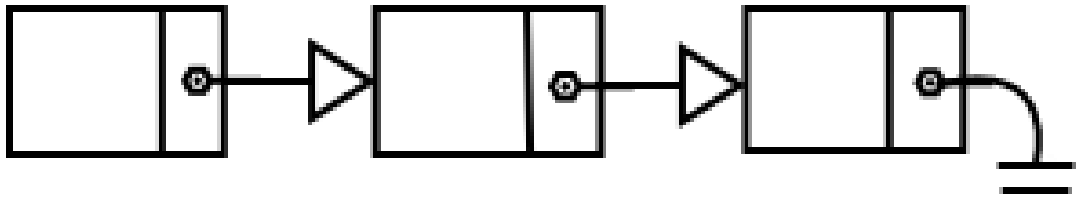


Implementación

```
struct listaelementos {
  int Dato;
  listaelementos *Enlace;
}Elemento;
```

Estructura de un nodo

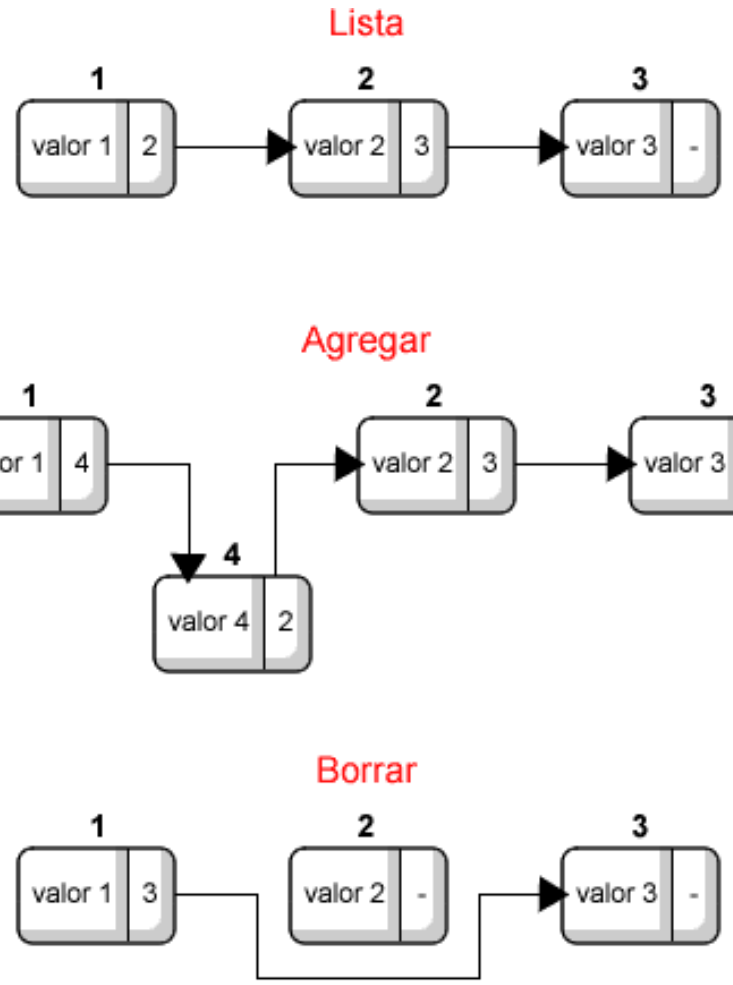
```
typedef struct ListaIdentificar {
   Elemento *inicio;
   Elemento *fin;
   int tam;
 }lista;
```



Lista enlazada

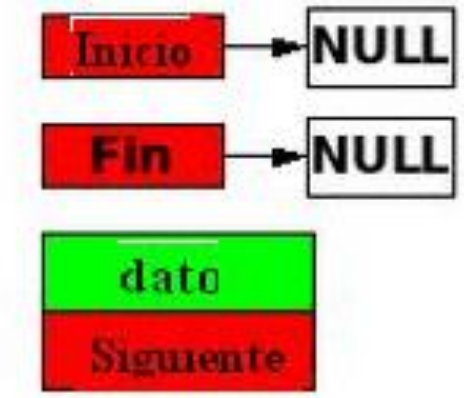
Operaciones

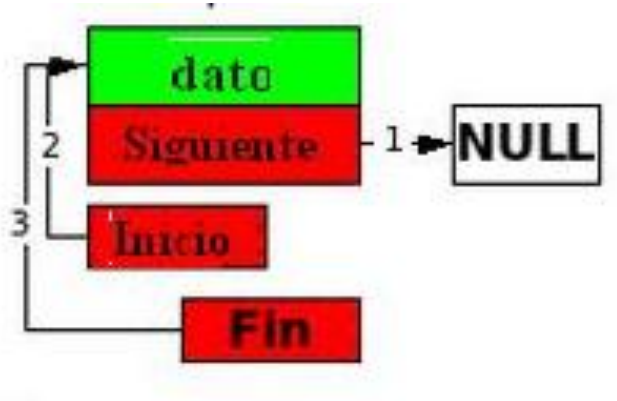
- Insertar: inserta un nodo con dato x en la lista, pudiendo realizarse esta inserción al principio o final de la lista o bien en orden.
- Eliminar: elimina un nodo de la lista, puede ser según la posición o por el dato.
- Buscar: busca un elemento en la lista.
- Vaciar: borra todos los elementos de la lista



Inserta en lista vacía

```
Elemento *nuevo_elemento;
if ((nuevo_elemento = (Elemento *) malloc (sizeof (Elemento))) == NULL) exit 1;
nuevo_elemento->dato =10;
nuevo_elemento->siguiente = NULL;
lista->inicio = nuevo_elemento;
lista->fin = nuevo_elemento;
lista->tam++;
```





Insertar al inicio

Elemento *nuevo_elemento;

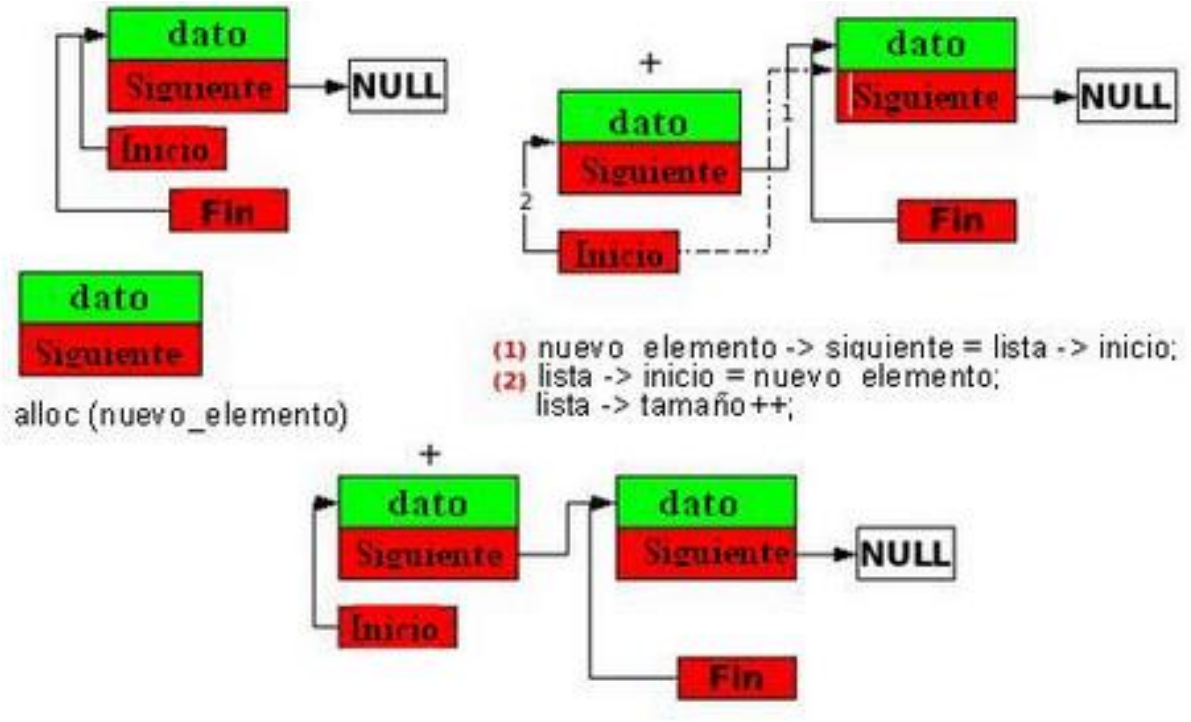
if ((nuevo_elemento = (Elemento *) malloc (sizeof (Elemento))) == NULL) exit 1;

nuevo_elemento->dato=‘a’;

nuevo_elemento->siguiente = lista->inicio

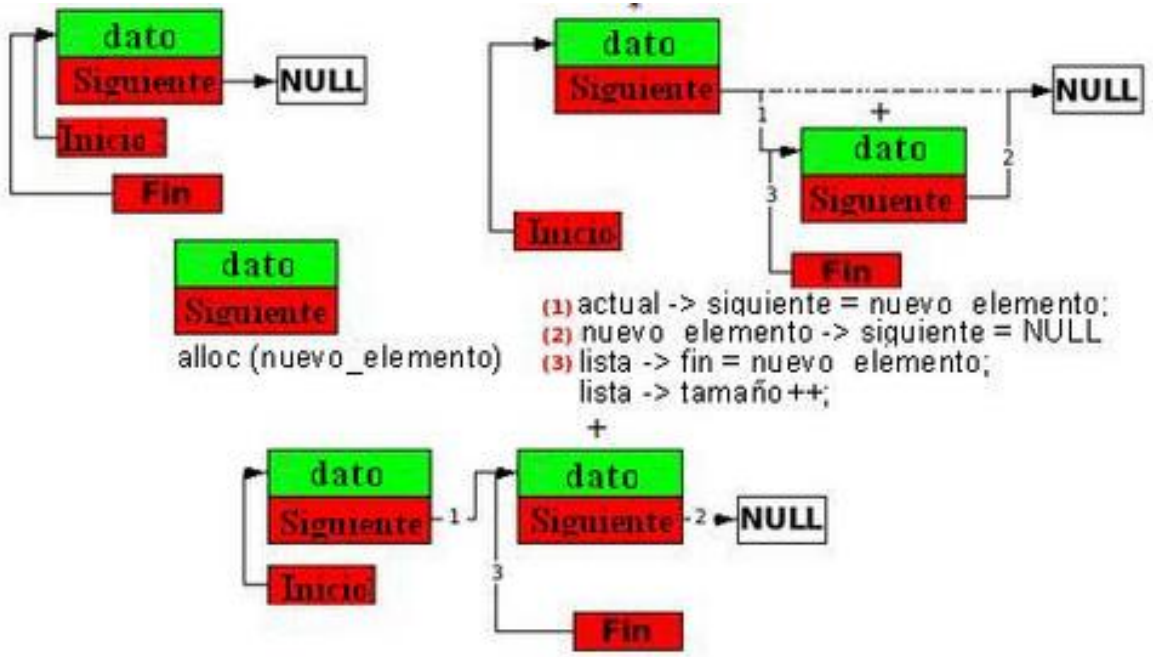
lista->inicio = nuevo_elemento;

lista->tam++;



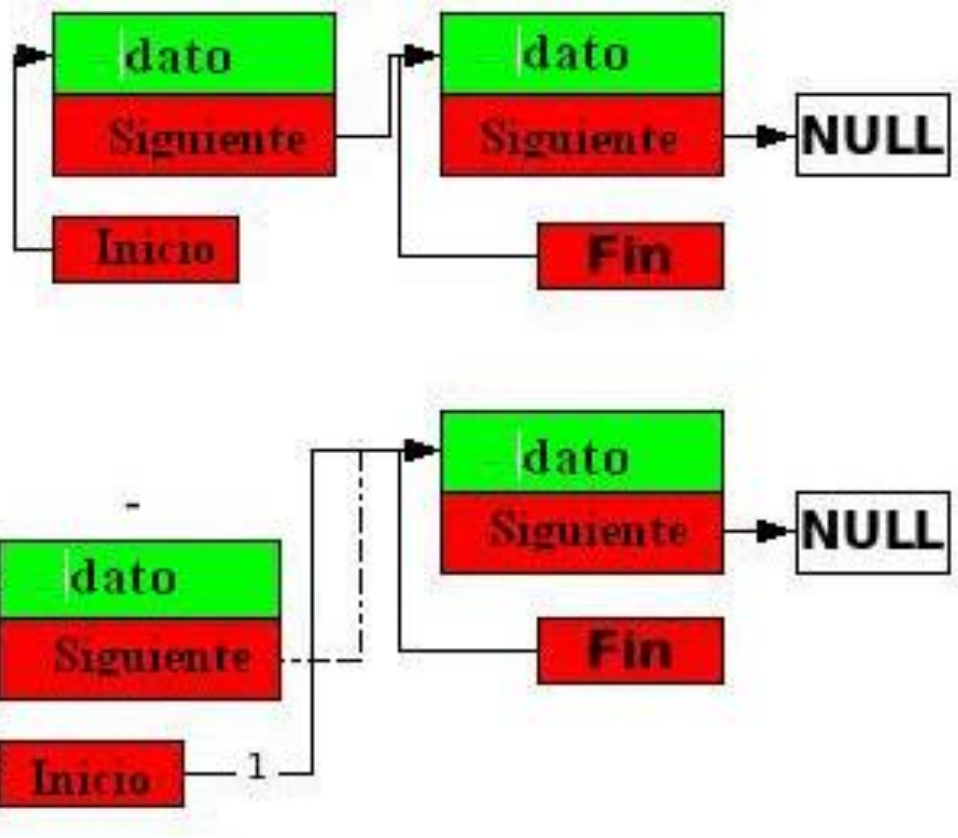
Insertar al final de la lista

```
Elemento *nuevo_elemento;
if ((nuevo_elemento = (Elemento *) malloc (sizeof (Elemento))) == NULL) exit 1;
strcpy (nuevo_elemento->dato, “una parte”);
actual->siguiente = nuevo_elemento;
nuevo_elemento->siguiente = NULL;
lista->fin = nuevo_elemento;
lista->tam++;
```



Eliminar al inicio

```
if (lista->tamaño == 0) exit 1;
Elemento *sup_elemento;
sup_element = lista->inicio;
lista->inicio = lista->inicio->siguiente;
if (lista->tam == 1)
lista->fin = NULL;
free (sup_elemento->dato);
free (sup_elemento);
lista->tam--;
```



Mostrar la lista

```
Elemento *actual;
actual = lista->inicio;
while (actual != NULL)
 {
   printf ("%p - %s\n", actual, actual->dato);
   actual = actual->siguiente;
}
```