UAI Universidad Abierta Interamericana

01 – PROGRAMACIÓN I

MG. NICOLÁS BATTAGLIA

Ti UAI TECNOLOGIA INFORMATICA

UAIOnline ultra

UAI Universidad Abierta Interamericana

UNIDAD 2

ESTRUCTURAS DINÁMICAS I - CLASE 4 - LISTAS

Ti UAI TECNOLOGIA INFORMATICA

UAIOnline ultra

ESTRUCTURAS DINÁMICAS I

- LISTAS
  - Enlazadas, Doblemente enlazadas y Circulares
- PILAS
- COLAS
- Operaciones
  - **Recorrido** procesa cada elemento de la estructura
  - **Búsqueda** Recupera la posición de un elemento específico
  - **Inserción** Adiciona un nuevo elemento a la estructura
  - **Borrado** Elimina un elemento
  - **Ordenación** Ordena los elementos de la estructura de acuerdo a sus valores
  - **Mezcla** Combina 2 estructuras

UAIOnline ultra

UAIOnline ultra

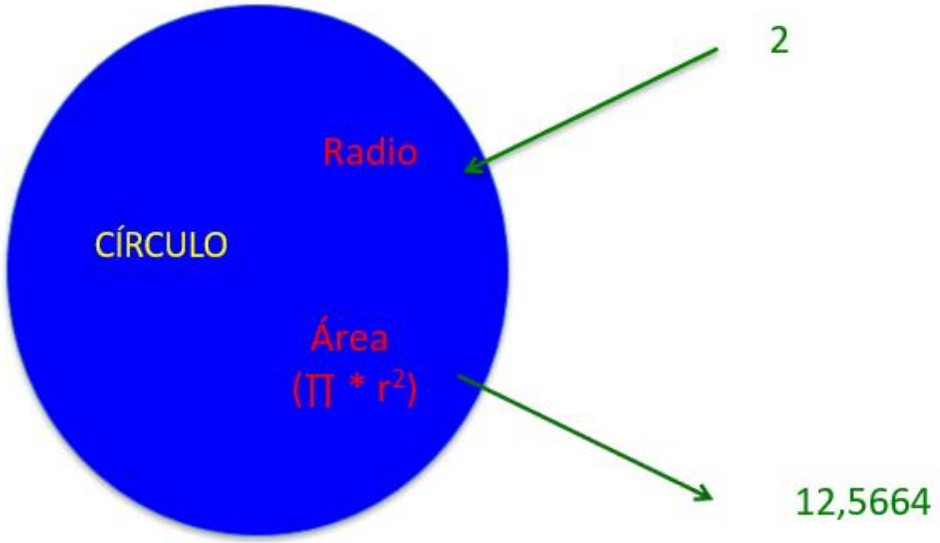
TIPOS DE DATOS ABSTRACTOS

- Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones.
- TAD = valores + operaciones
- TDA y Abstracción de datos son conceptos relacionados.
- Un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

TIPOS DE DATOS ABSTRACTOS

- Una abstracción es la simplificación de un objeto o de un proceso de la realidad en la que sólo se consideran los aspectos más relevantes.
- La abstracción se utiliza por los programadores para dar sencillez de expresión al algoritmo

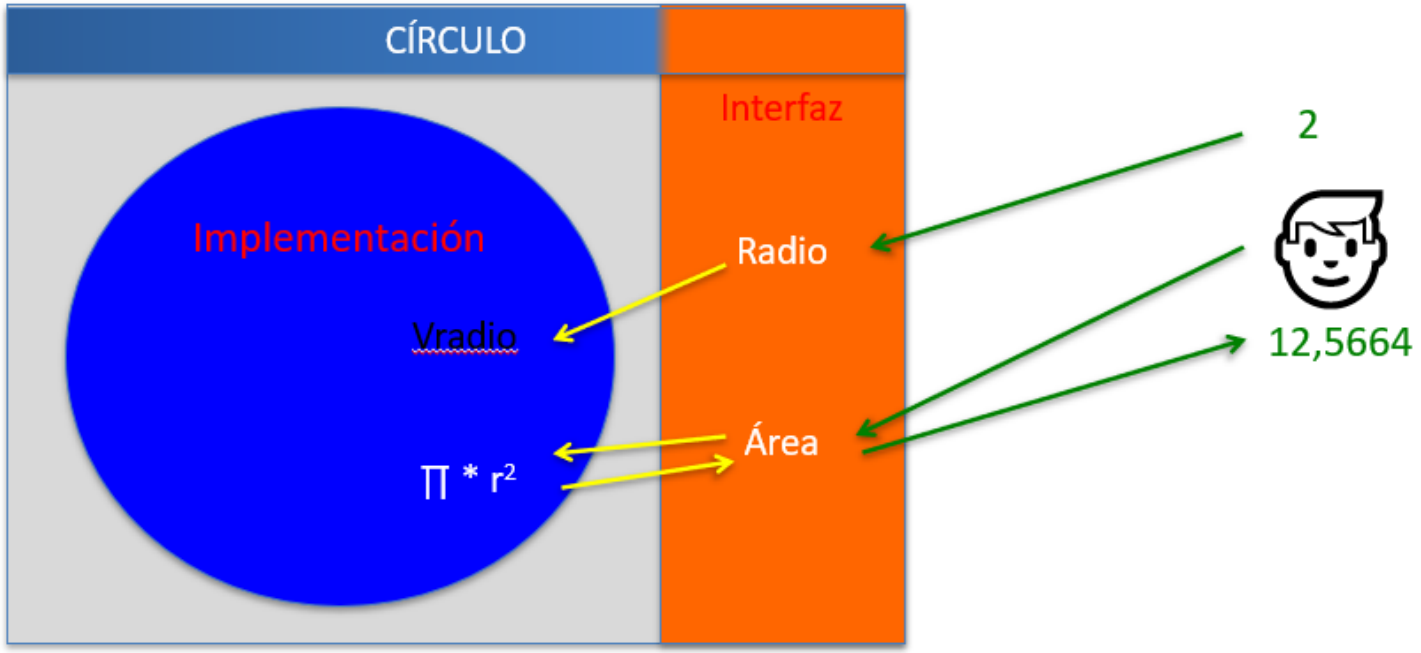
Ejemplo de TDA y Circulo



TIPOS DE DATOS ABSTRACTOS

- Una abstracción es la simplificación de un objeto o de un proceso de la realidad en la que sólo se consideran los aspectos más relevantes.
- La abstracción se utiliza por los programadores para dar sencillez de expresión al algoritmo

Ejemplo de TDA y Circulo



UAIOnline ultra

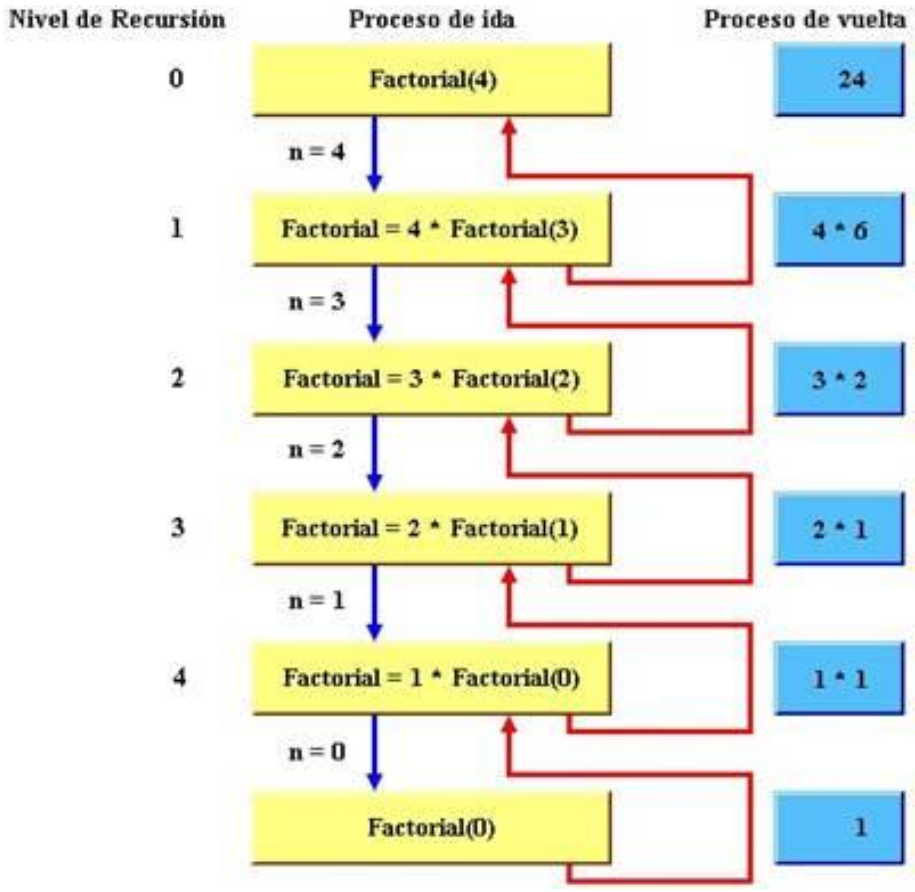
RECURSIVIDAD Y FUNCIONES RECURSIVAS

- Muchos problemas que se resuelven con iteración los podemos fácilmente resolver con recursividad.
- Las funciones recursivas son funciones que se invocan a sí mismas.
- Son equivalentes a estructuras tipo bucle pero permiten especificar muchos problemas de un modo más simple y natural que éstos, de ahí su importancia en programación.
- dentro del cuerpo de la función se incluyen llamadas a la propia función.
- Es una alternativa al uso de bucles.
- Cuando un programa llama a una función que llama a otra, la cual llama a otra y así sucesivamente, las variables y valores de los parámetros de cada llamada a cada función se guardan en la pila o stack, junto con la dirección de la siguiente línea de código a ejecutar

UAIOnline ultra

RECURSIVIDAD Y FUNCIONES RECURSIVAS

- Toda recursividad debe tener una condición de parada. Es la forma en que se detiene su ejecución.
- LA recursividad ocupa memoria a medida que crece. Su mal uso puede generar errores como "Stack Overflow" o "Desbordamiento de pila".



UAIOnline ultra

RECURSIVIDAD Y FUNCIONES RECURSIVAS

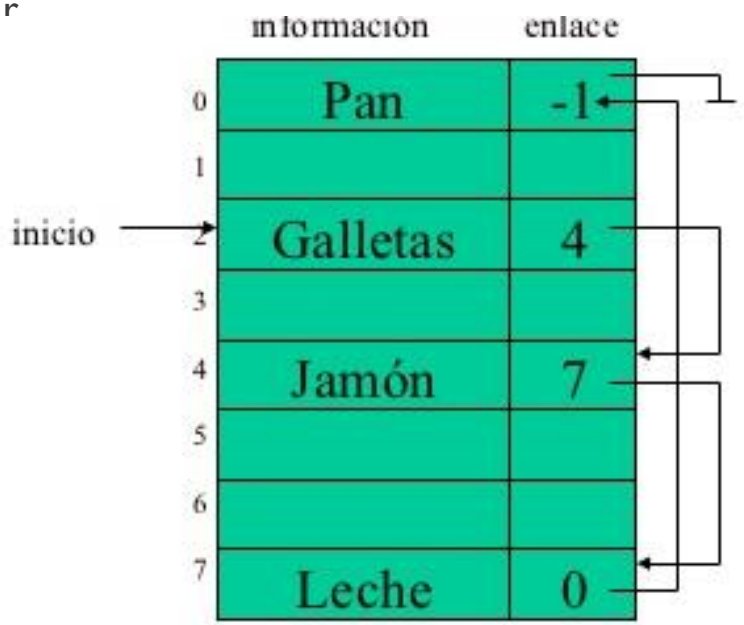
- Recursividad vs Ciclos

```
function factorial(n) {
    if (n<=1) return 1;
    return n* factorial(n-1);
}
```

```
function factorial(n){
  var res = 1;
  for(var i=n; i>=1; i--){
    res = res * i;
  }
  return res;
}
```

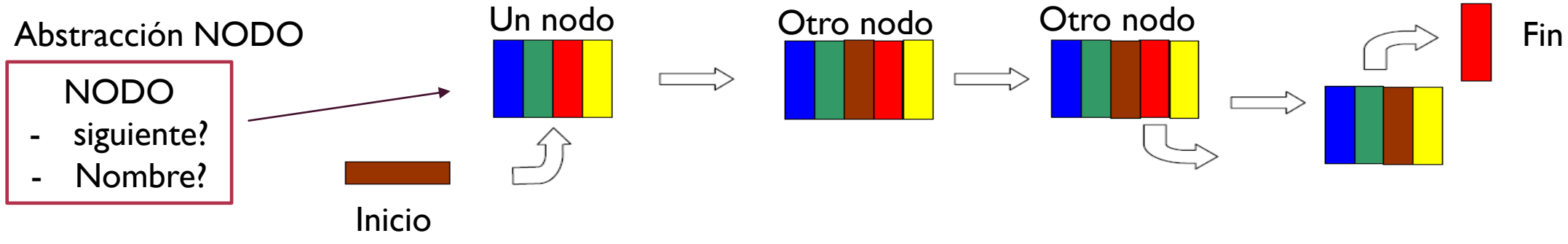
LISTAS ENLAZADAS

- Eliminar un elemento
  - Uso de punteros temporales para almacenar la dirección de los elementos a borrar
  - Eliminar al principio
  - Eliminar en otro lugar
- Recorrer lista
  - Hasta que el puntero a Siguiente sea NULL
  - Hasta que el puntero a Anterior sea NULL
- Agregar un elemento
  - Al principio
  - Al final
  - En una posición determinada



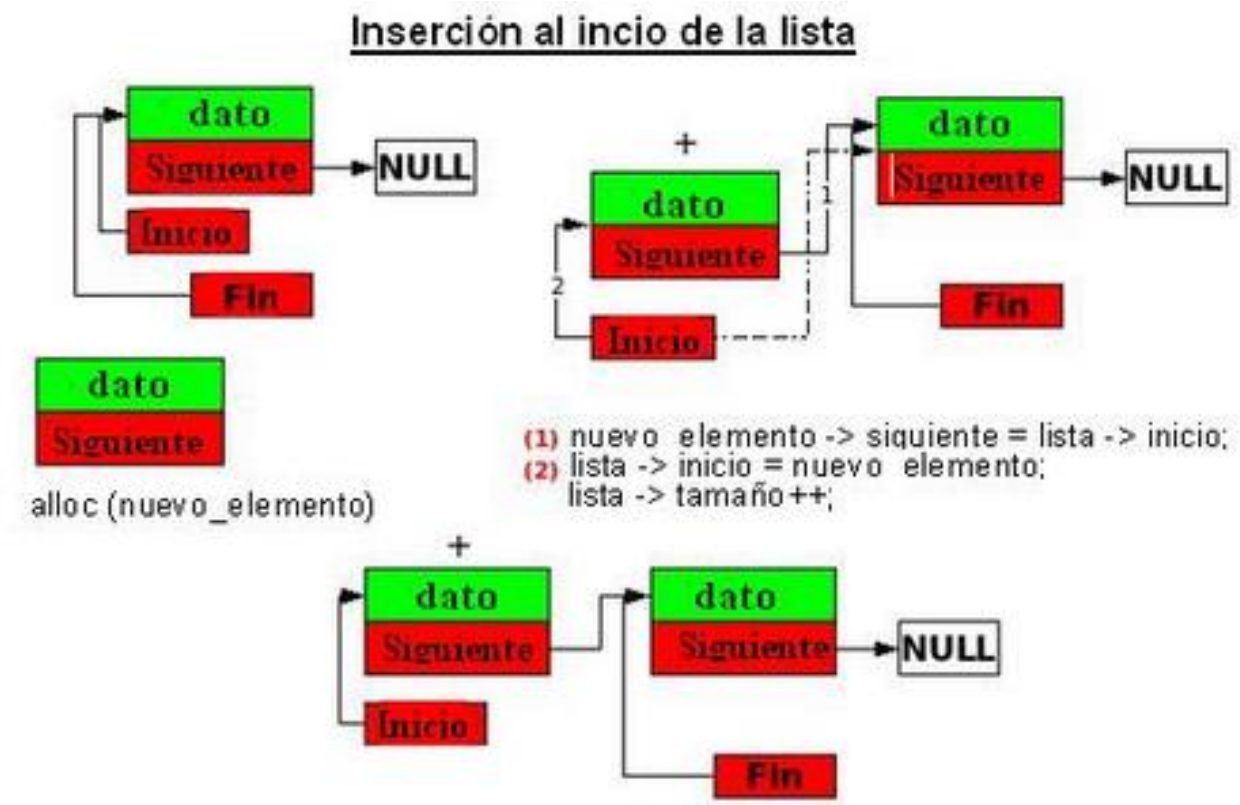
LISTAS ENLAZADAS

- Una lista es una estructura de datos en la cual los elementos almacenados en la misma pueden ser agregados, borrados y accedidos sin restricciones, en cualquier punto de la estructura.
- Una lista se compone de NODOS.
- En las listas se pueden ver todos los elementos de la estructura, permitiendo realizar recorridos y consultas de los datos.
- En la estructura de una lista se distinguen dos elementos:
  - El principio, a partir del cual se inician las búsquedas y recorridos.
  - El corriente, elemento de referencia en la lista, a partir del cual se realizan borrados, inserciones y modificaciones.

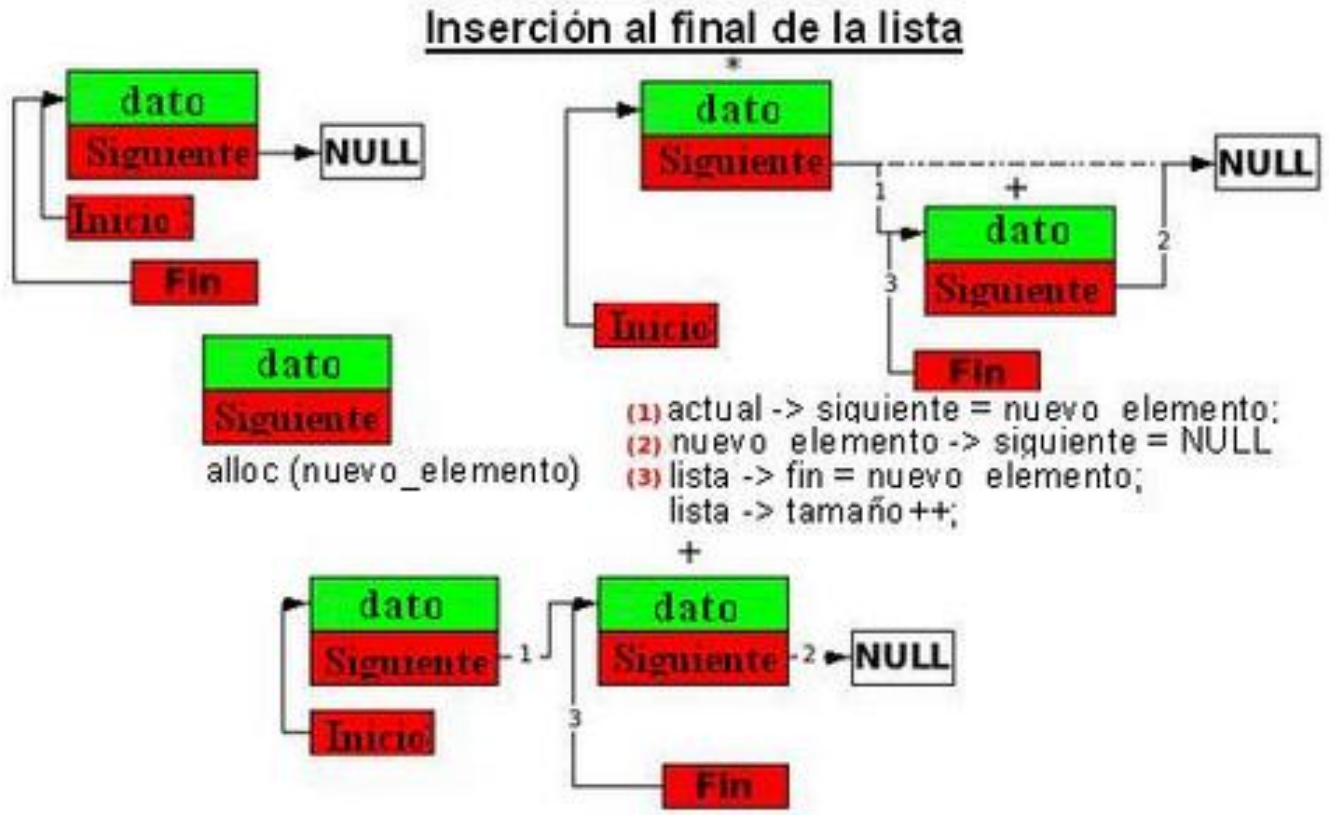


LISTAS ENLAZADAS SIMPLES

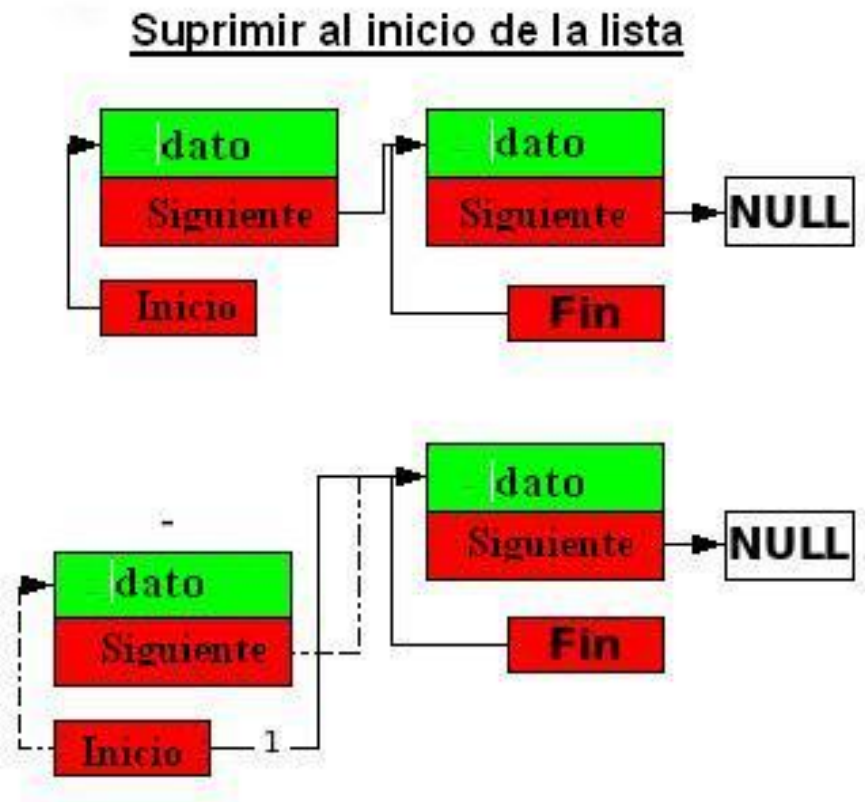
LISTAS ENLAZADAS SIMPLES



LISTAS ENLAZADAS SIMPLES

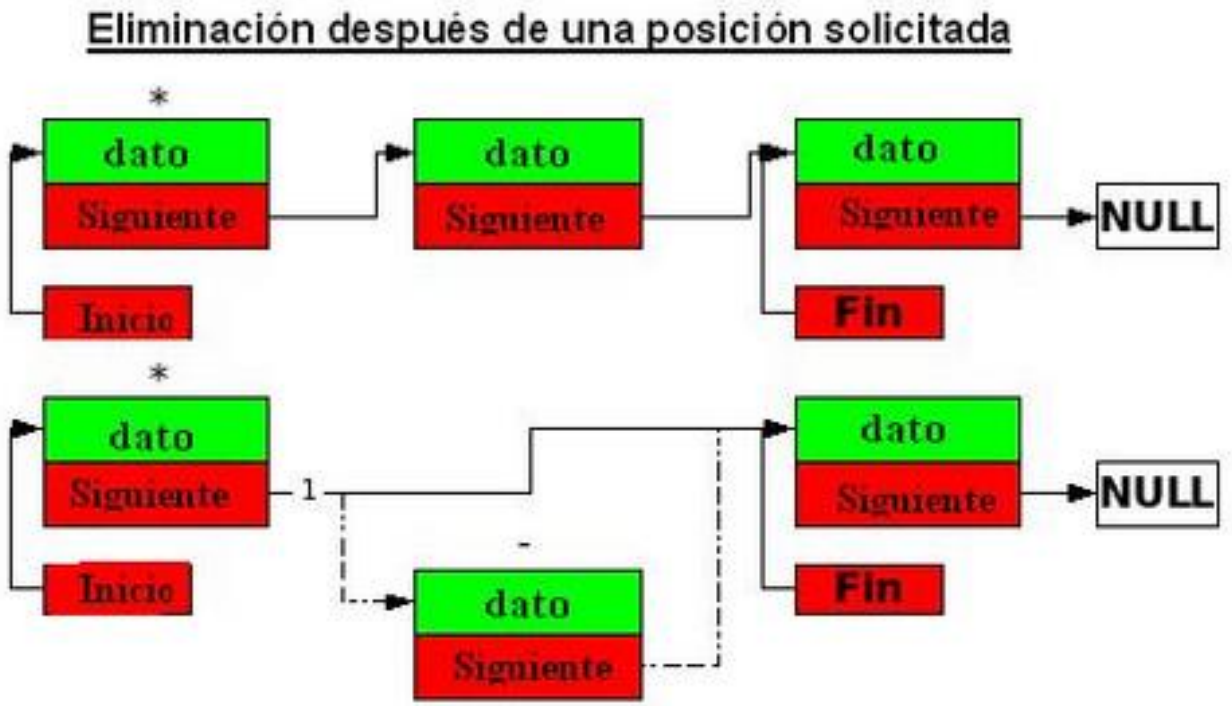


LISTAS ENLAZADAS SIMPLES




```
    sup_elemento = lista -> inicio;
(1) lista -> inicio = lista -> inicio -> siguiente;
    lista -> tamaño++;
```

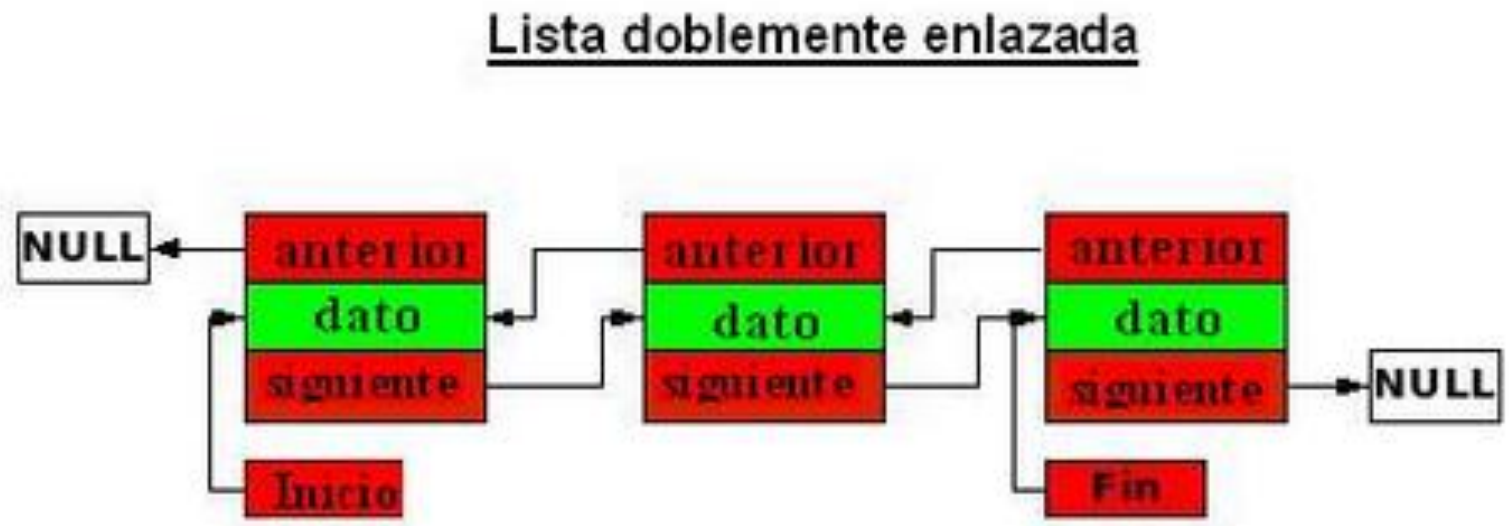
LISTAS ENLAZADAS SIMPLES



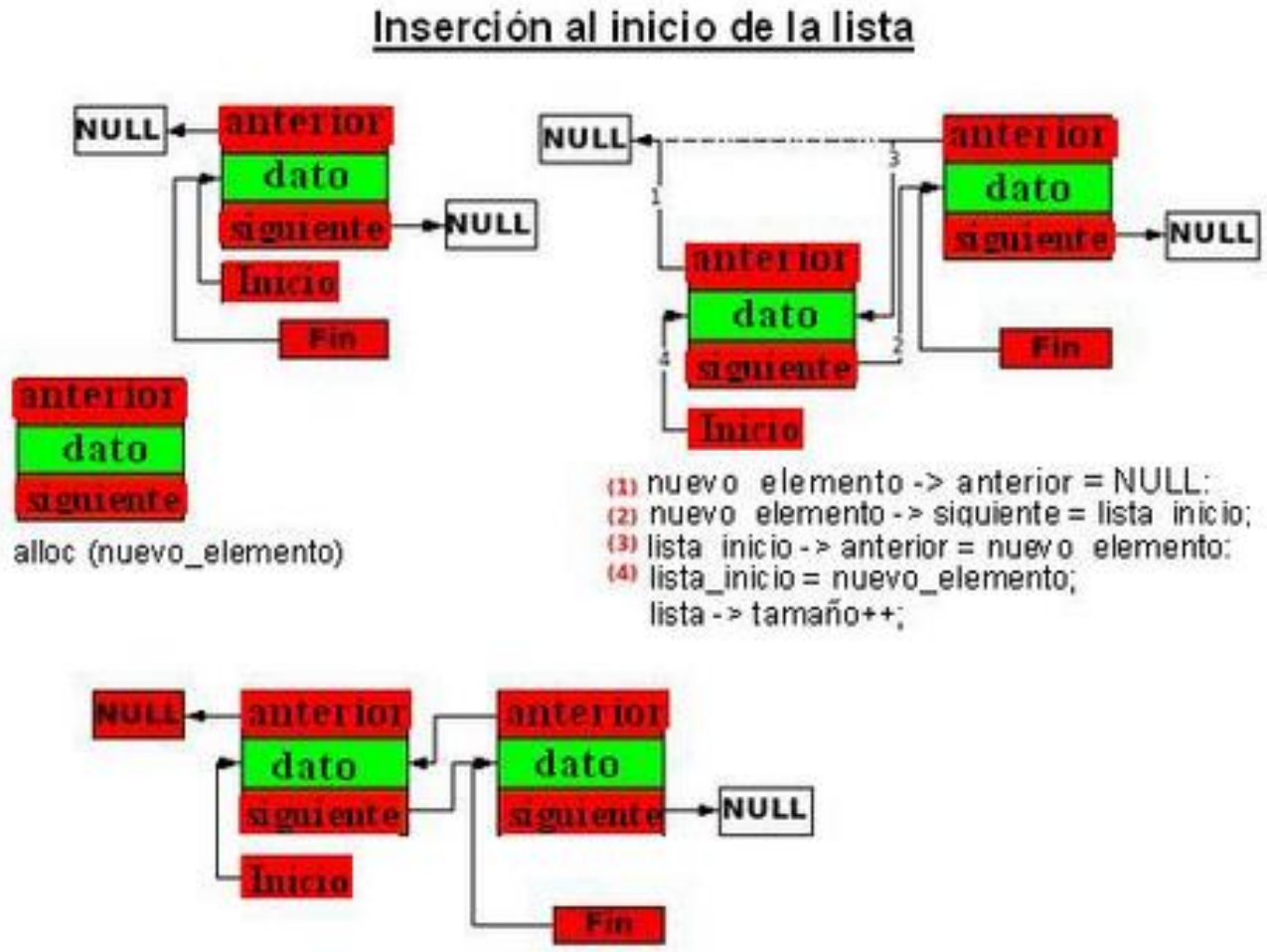

```
    sup_elemento = actual -> siguiente;
(1) actual -> siguiente = actual -> siguiente -> siguiente;
    lista -> tamaño++;
```

LISTAS DOBLEMENTE ENLAZADAS

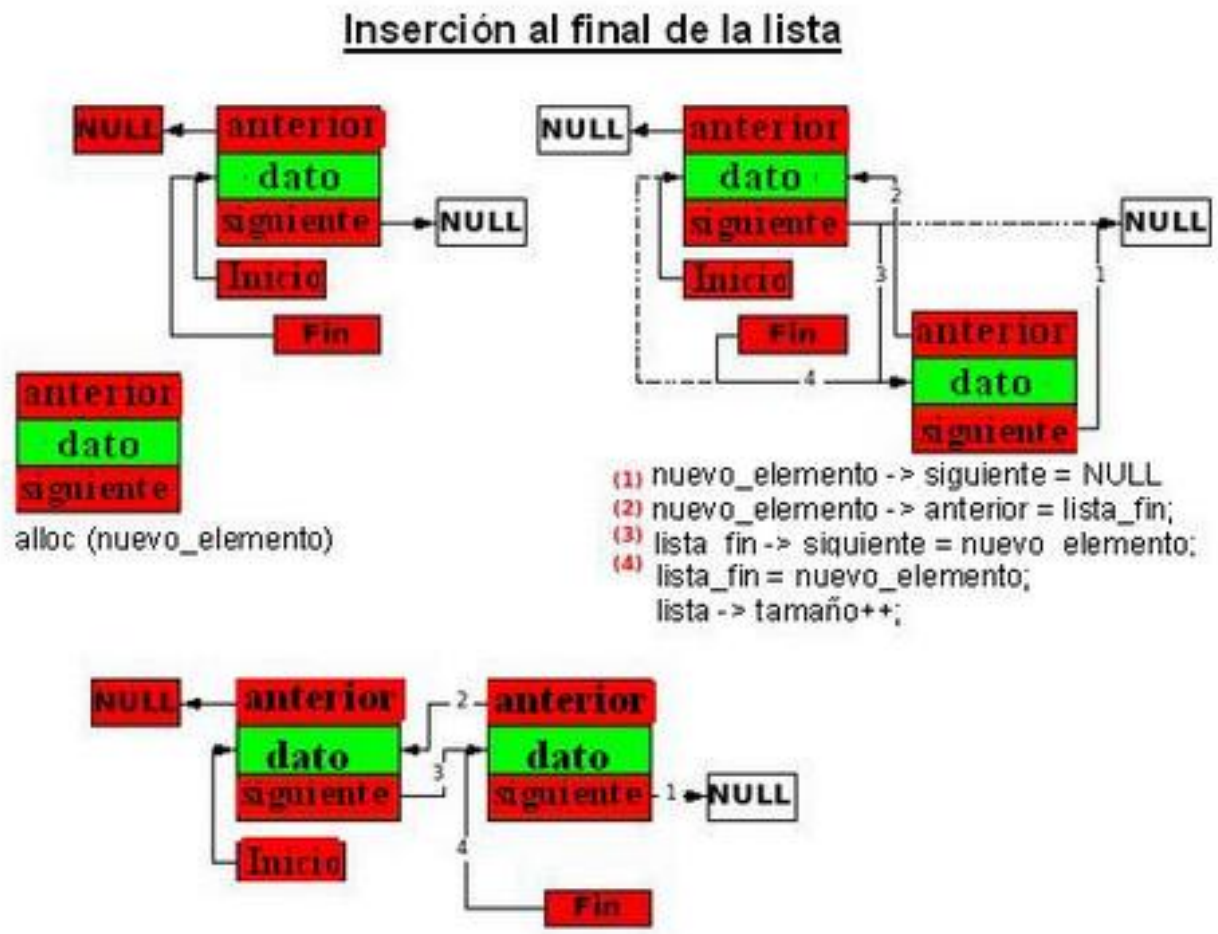
- Similar a las enlazadas simples pero con puntero al elemento anterior.
- Se pueden usar cuando se necesitan varias operaciones de inserción y eliminación.
- El puntero anterior del primer elemento debe apuntar hacia NULL (el inicio de la lista).
- El puntero siguiente del último elemento debe apuntar hacia NULL (el fin de la lista).



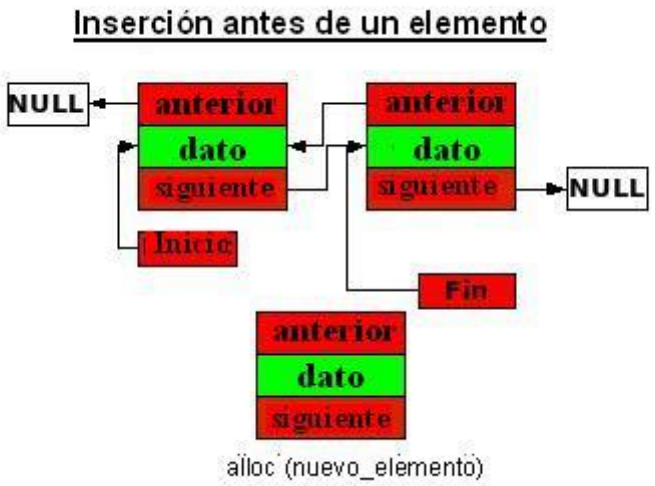
LISTAS DOBLEMENTE ENLAZADAS

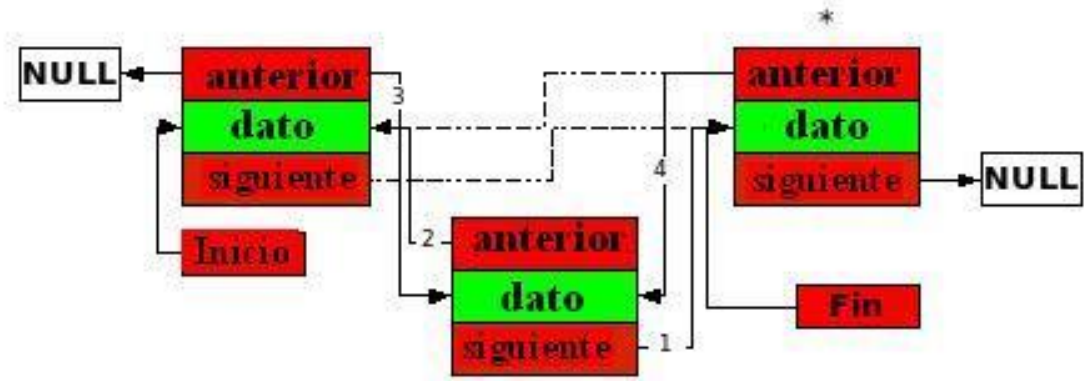


LISTAS DOBLEMENTE ENLAZADAS

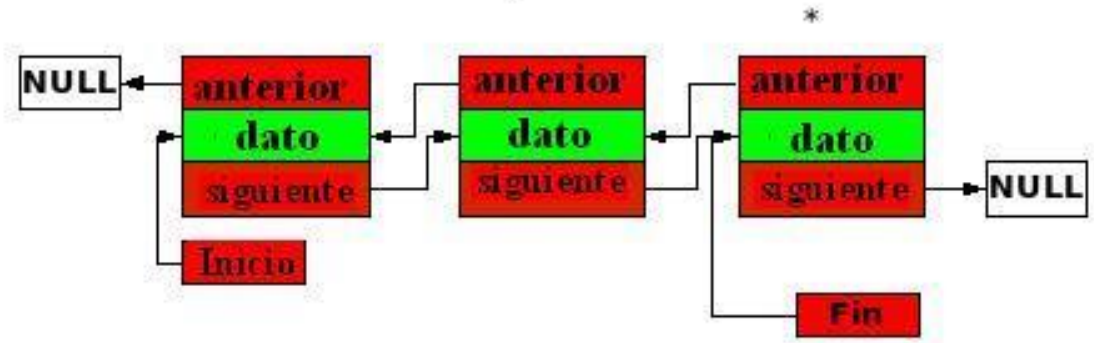


LISTAS DOBLEMENTE ENLAZADAS



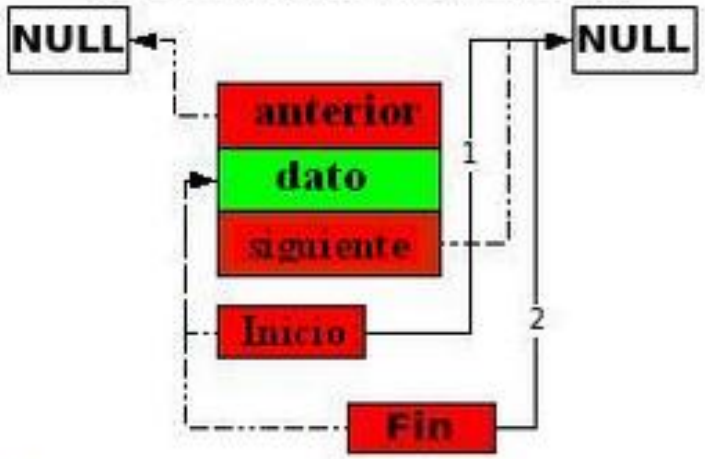


(1) nuevo_elemento -> siguiente = actual -> siguiente;
(2) nuevo_elemento -> anterior = actual;
(3) actual -> anterior -> siguiente = nuevo_elemento;
(4) actual -> anterior = nuevo_elemento;
lista -> tamaño++;



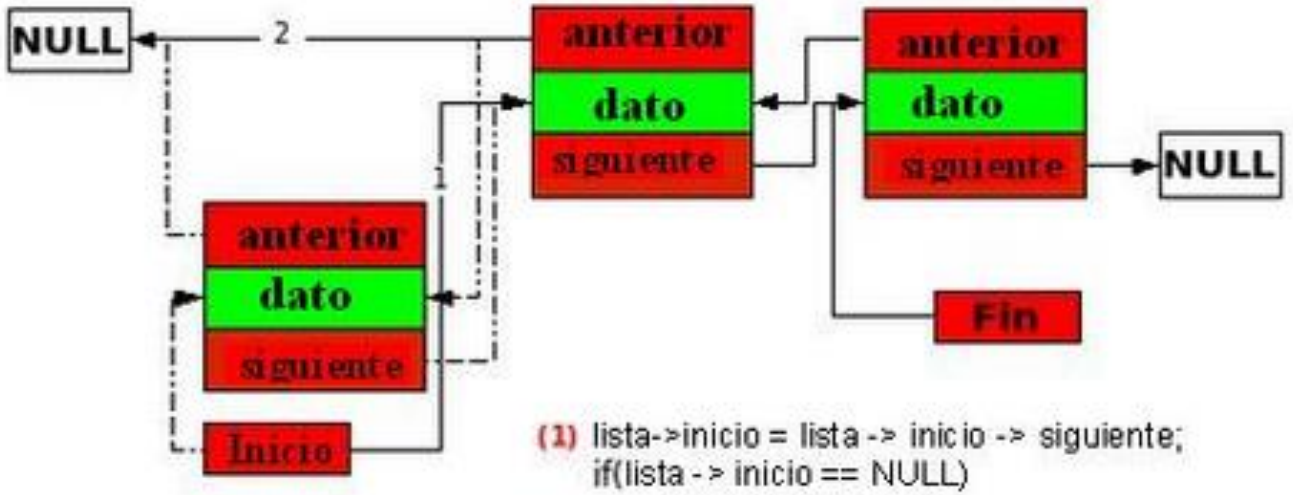
LISTAS DOBLEMENTE ENLAZADAS

Eliminación en la posición 1
lista con un solo elemento



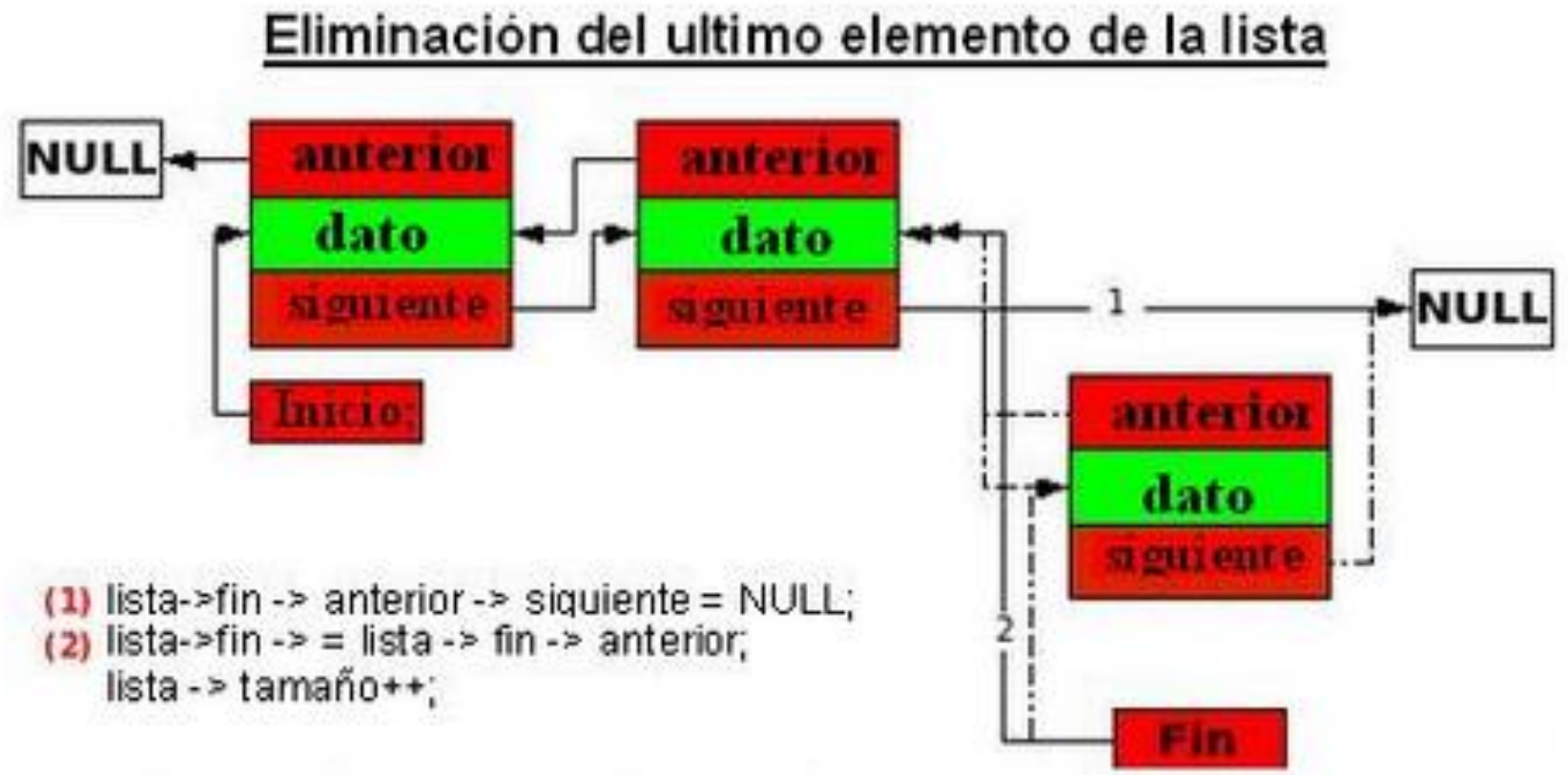

```
(1) lista->inicio = lista -> inicio -> siguiente;
    if(lista -> inicio == NULL)
        lista-> fin = NULL;
    else
(2)     lista->inicio -> anterior == NULL;
    lista -> tamaño++;
```

Eliminación en la posición 1
lista con varios elementos

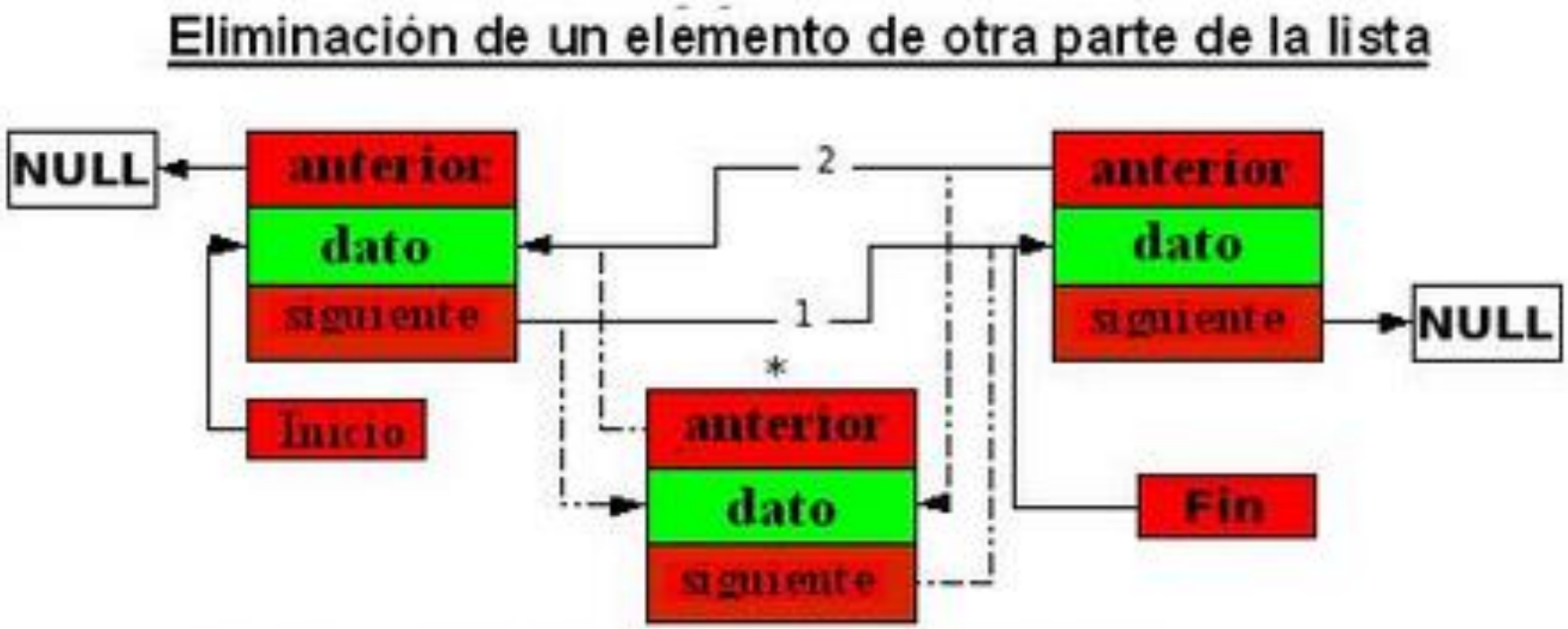



```
(1) lista->inicio = lista -> inicio -> siguiente;
    if(lista -> inicio == NULL)
        lista-> fin = NULL;
    else
(2)     lista->inicio -> anterior == NULL;
    lista -> tamaño++;
```

LISTAS DOBLEMENTE ENLAZADAS

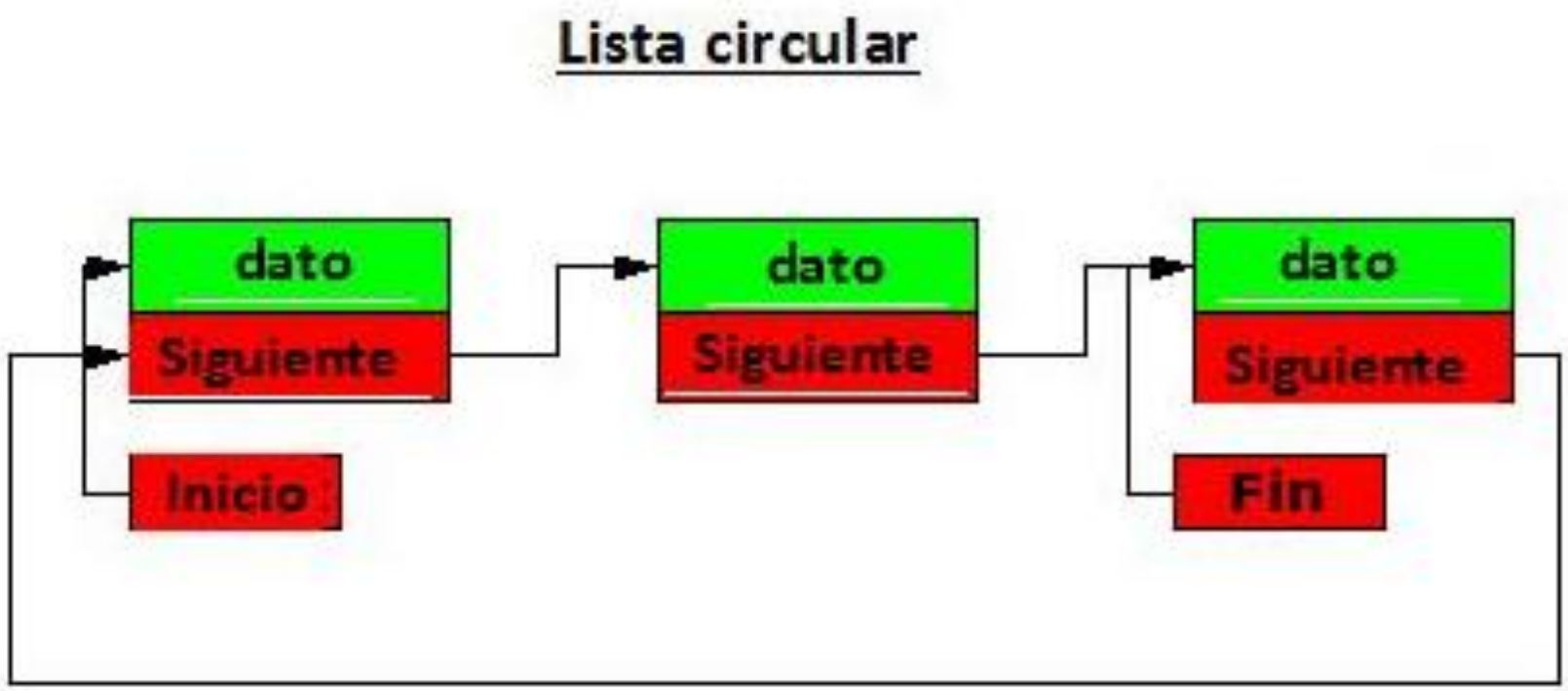


LISTAS DOBLEMENTE ENLAZADAS



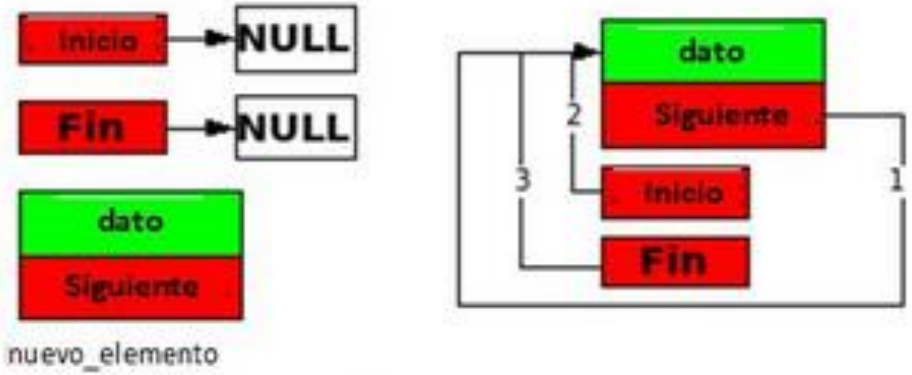
LISTAS CIRCULARES

- Una lista simple o doblemente enlazada, pero el ultimo elemento es un puntero al primero
- Los punteros Inicio y Fin apuntan al mismo elemento



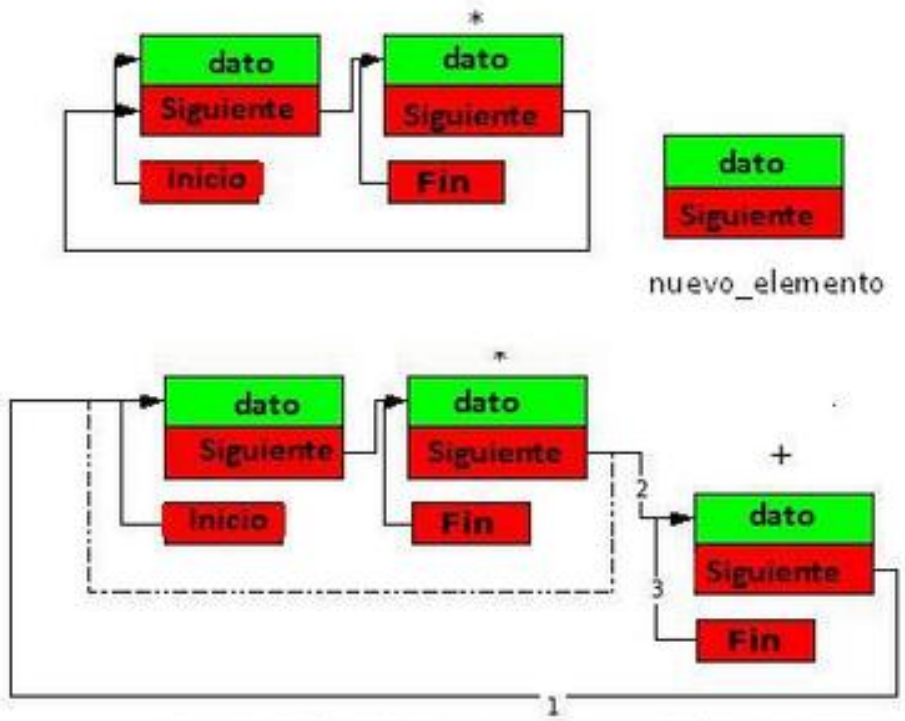
LISTAS CIRCULARES

Inserción en la lista circular vacía




```
(1) nuevo_elemento->siguiente = nuevo_eleme
(2) lista->inicio = nuevo  elemento;
(3) lista->fin = nuevo_elemento;
    lista->tamaño++;
```

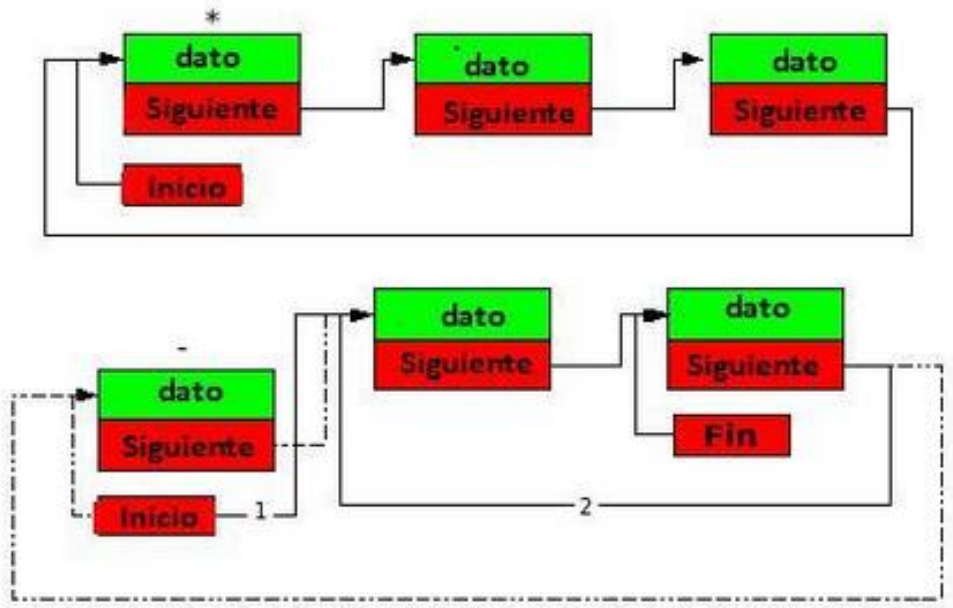
Inserción en una lista no vacía




```
(1) nuevo_elemento->siguiente = actual->siguiente;
(2) actual->siguiente = nuevo_ elemento;
(3) lista->fin = nuevo  elemento;
    lista->tamaño++;
```

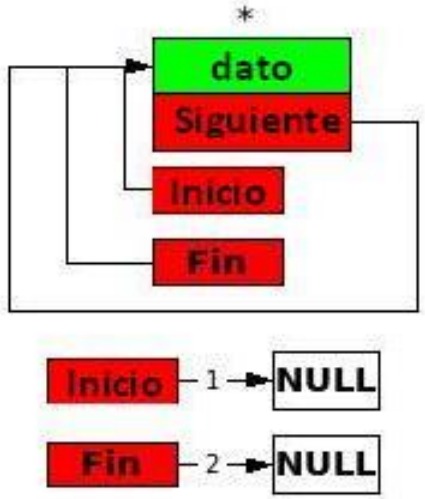
LISTAS CIRCULARES

Eliminación en la lista circular




```
sup_elemento = lista->inicio;
(1) lista->inicio = lista->inicio->siguiente;
(2) lista->fin->Siguiente = lista->inicio;
    lista->tamaño--;
```

Eliminación del último elemento de la lista circular




```
sup_elemento = lista->inicio;
(1) lista->inicio = NULL;
(2) lista->fin = NULL:
    lista->tamaño--;
```