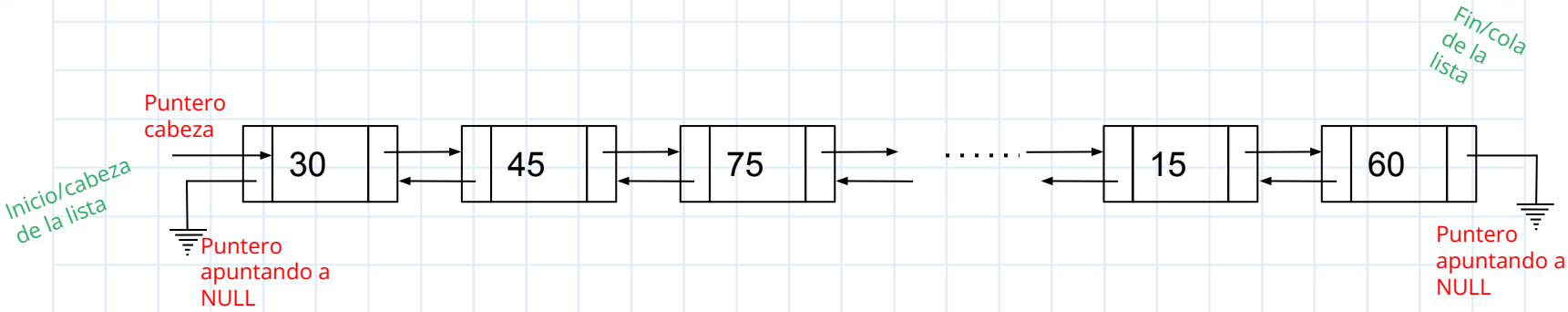
Lista lineal doblemente enlazada

La apariencia de una lista de este tipo es la siguiente:



Donde...

Cada nodo tiene esta estructura

```
struct nodo{

   // Campos...

   // Enlace al siguiente nodo
   // Enlace al nodo anterior

};
```

Los campos pueden ser los que se necesiten. Por ejemplo:

```
int dato;
```

El campo del puntero que apunta al siguiente nodo:

```
struct nodo *sig;
```

El campo del puntero que apunta al nodo anterior:

```
struct nodo *ant;
```

Así, tendremos:

```
struct nodo{

   int dato;

   struct nodo *sig;
   struct nodo *ant;

};
```

El programa principal

En la función main siempre declararemos el objeto lista

```
int main(void)
{
   cout << "MANEJO DE LISTAS DOBLEMENTE ENLAZADAS" << endl << endl;

   ListaDoble objListaDoble;

   // Invocamos las funciones miembro o
   // invocamos un menú.

   cout << endl;
   return 0;
}
```

Las funciones miembro

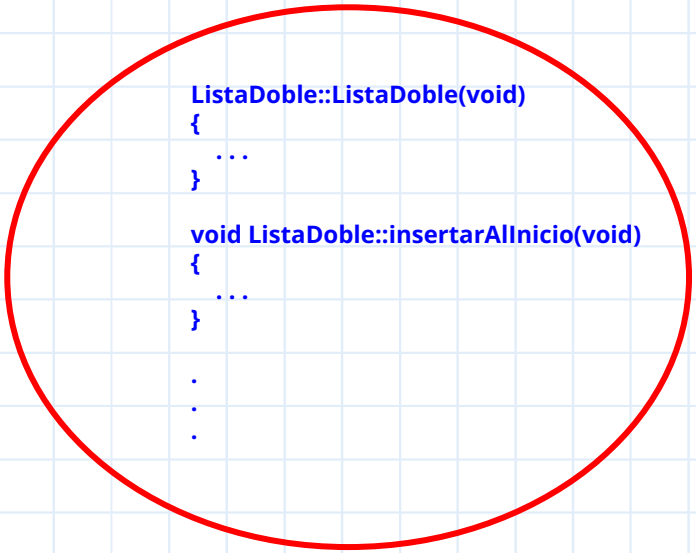
```
class ListaDoble{
   private:
      nodo *pInicio;

   public:
      ListaDoble();
      ~ListaDoble();
      void insInicio(int);
      void mostrarListaDoble(void);
      void mostrarListaRecursiva(void);
      void mostrarListaRecursivaAux(nodo *);
      void mostrarListaInversa(void);
      void mostrarListaInversaAux(nodo *);
      void insertarFinalLista(void);
      void insFinal(int dato);
      void insFinalRec(int, nodo *);
      nodo *irUltimoNodoRec(nodo *);
      void insertarDespuesDeElemento(int, int);
      void insertarAntesDeElemento(int, int);
      bool buscarEnListaDoble(int);
      bool eliminarElemento(int);
};
```

Están declaradas dentro de una clase

Su tarea es:
Administrar la estructura de datos (insertar elementos, buscar elementos, eliminar elementos, ...)

Los cuerpos de las funciones miembro



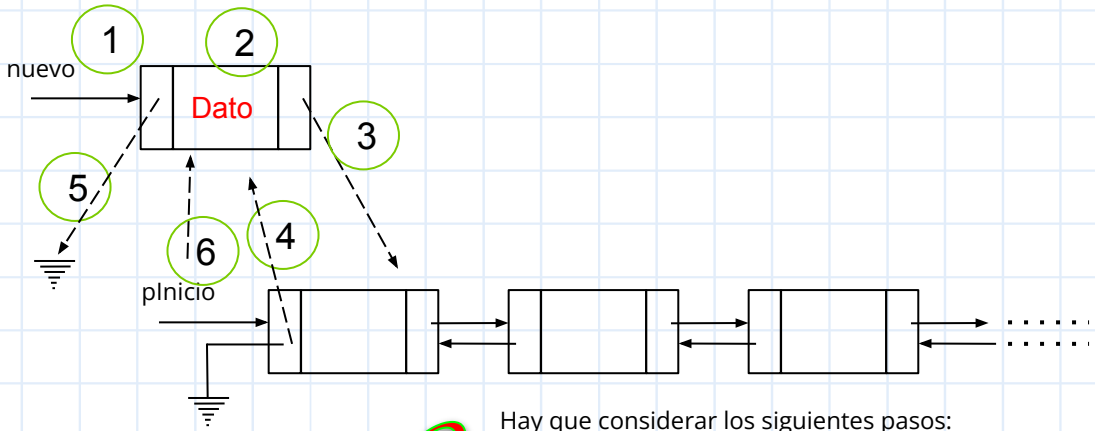
Se ubican entre la definición
de la clase y la función main

Inserción de elementos en una lista lineal doblemente enlazada

La inserción puede realizarse:

- Al inicio de la lista.
- Al final de la lista.
- En orden ascendente.
- En orden descendente.
- Antes de un elemento determinado.
- Después de un elemento determinado.
- Etc.

La inserción al inicio de la lista



Algoritmo

Hay que considerar los siguientes pasos:

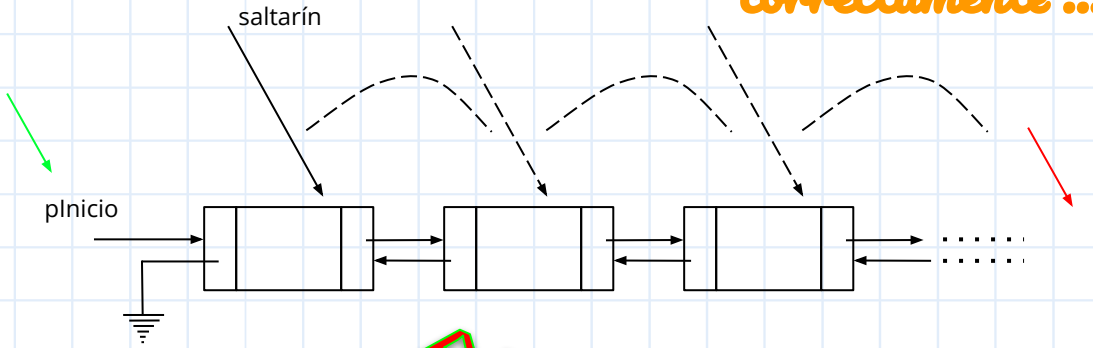
1) Crear el nodo.
2) Introducir el dato en el nodo.
3) Hacer que el puntero a siguiente, del nuevo nodo, apunte al primer elemento de la lista.
4) Hacer que el puntero a anterior, del nodo que está siendo señalado por el puntero cabeza, apunte al nuevo nodo.
5) Hacer que el puntero a anterior, del nuevo nodo, apunte a NULL.
6) Hacer que el puntero cabeza apunte al nuevo nodo creado.

Este mismo esquema funciona si la lista está vacía o hay que hacer otra consideración

Esto es lo nuevo para esta lista

Mostremos los elementos de la lista

... a ver si es cierto que los insertamos correctamente ...



Algoritmo

Hay que considerar los siguientes pasos:

1) Hacer que el puntero auxiliar apunte al primer nodo de la lista.
2) Desplegar el dato del nodo.
3) Iterativamente saltar a los consecutivos nodos realizando despliegue de su dato.
4) Hasta encontrar el final de la lista.

¿En qué momento podemos decir que saltarín ha terminado de recorrer toda la lista?

Mostremos los elementos de la lista

... a ver si es cierto que los insertamos correctamente ...

Podemos probar también si los punteros que señalan en sentido inverso están enlazando correctamente la estructura.

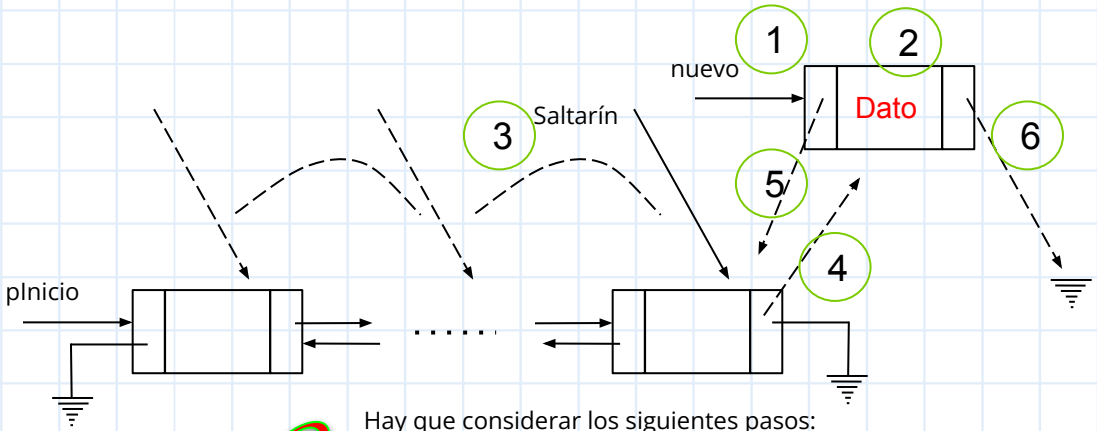
Así que probemos imprimir iterativamente en sentido directo y en sentido inverso.

Realicemos nuestro ejercicio con dos punteros auxiliares.

Uno para recorrerla e imprimir en sentido directo

Y otro para recorrerla e imprimir en sentido inverso.

La inserción al final de la lista



Algoritmo

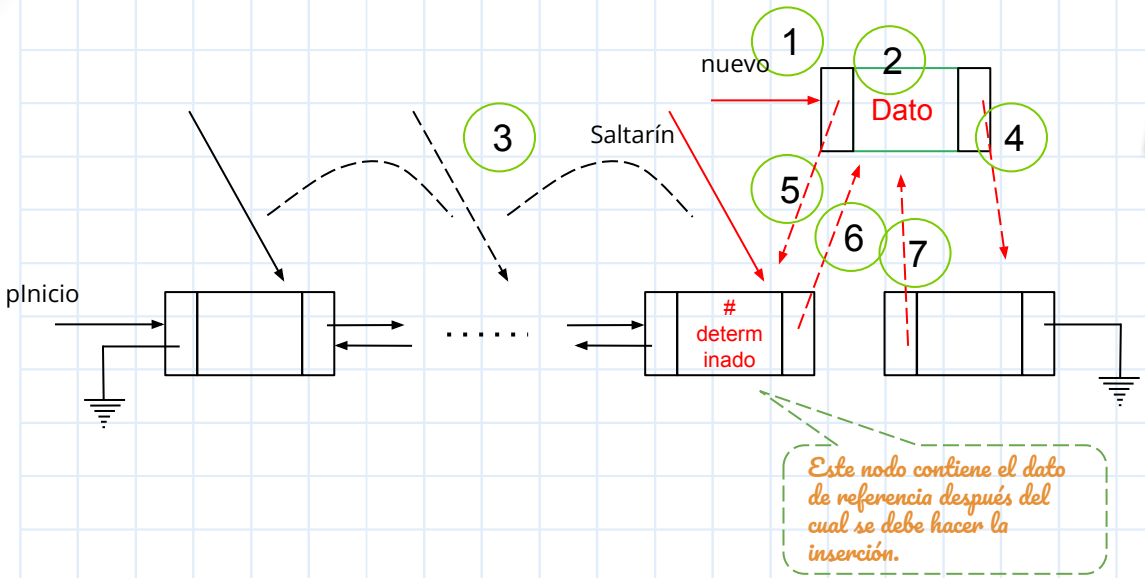
Hay que considerar los siguientes pasos:

1) Crear el nodo.
2) Introducir el dato en el nodo.
3) Llevar un puntero auxiliar hasta el final de la lista, sin salirse de la lista.
4) Hacer que el puntero a siguiente, del último nodo de la lista, apunte al nuevo nodo.
5) Hacer que el puntero a anterior, del nuevo nodo, apunte donde señala el puntero auxiliar.
6) Hacer que el puntero a siguiente, del nuevo nodo, apunte a NULL.

¿Este mismo esquema funciona si la lista está vacía o hay que hacer otra consideración?

Esto es lo nuevo para esta lista

Inserción después de un elemento determinado

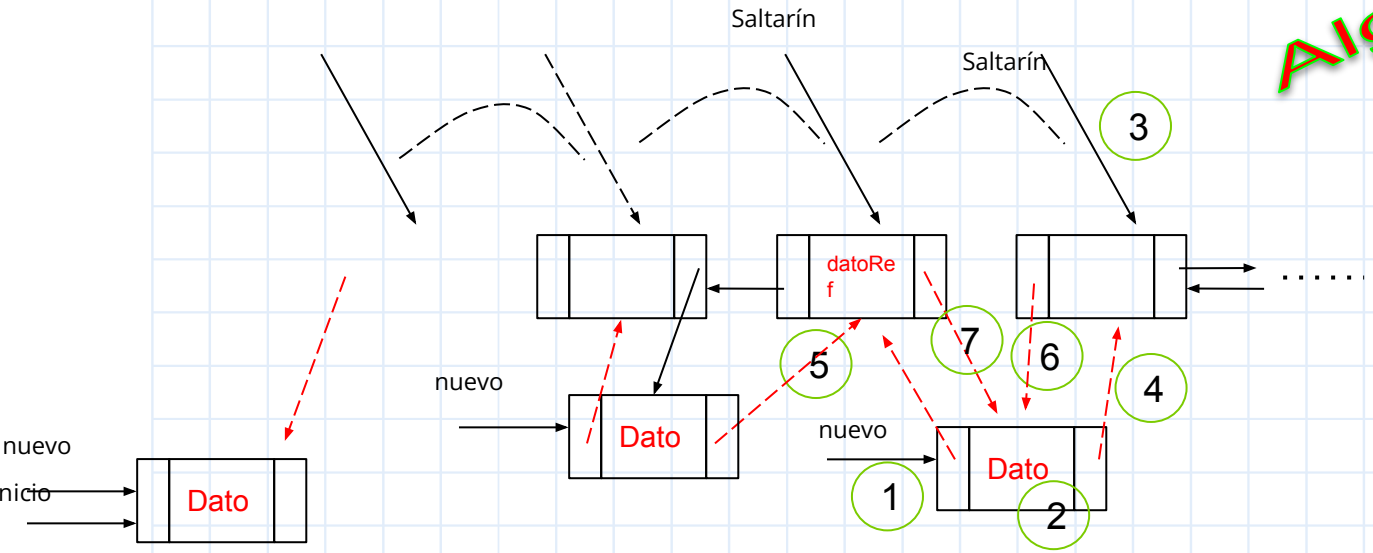


Algoritmo

Hay que considerar los siguientes pasos:

1) Crear el nodo.
2) Introducir el dato en el nodo.
3) Saltar hasta ubicarse sobre el nodo que contiene al dato de referencia.
4) Hacer que el puntero a siguiente, del nuevo nodo, apunte donde apunta el nodo con el dato de referencia.
5) Hacer que el puntero a anterior, del nuevo nodo, apunte donde señala el puntero auxiliar.
6) Si hay un nodo después del que se está insertando, su puntero a anterior debe apuntar al nuevo nodo.
7) Hacer que el puntero a siguiente, del nodo del dato de referencia apunte al nuevo nodo.

Inserción antes de un elemento determinado

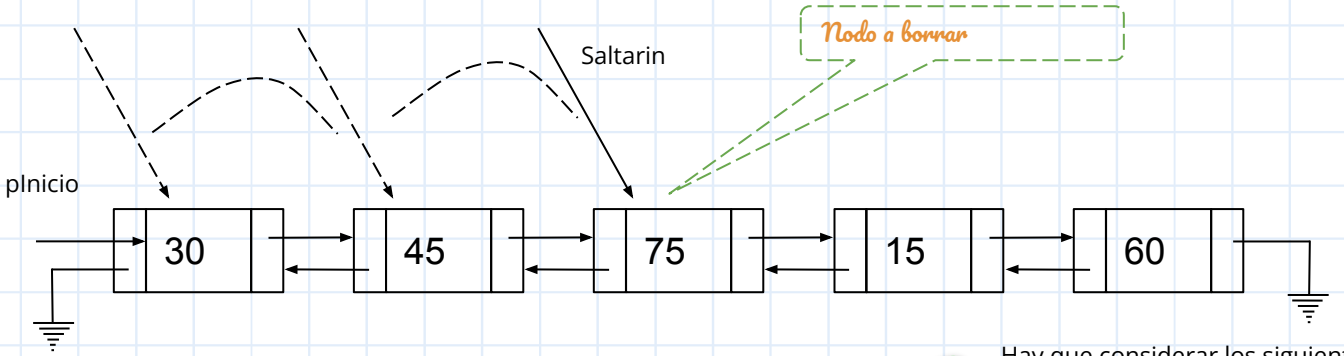


Algoritmo

Hay que considerar los siguientes pasos:

1) Crear el nodo.
2) Introducir el dato en el nodo.
3) Saltar hasta ubicarse sobre el nodo que contiene al dato de referencia.
4) Hacer que el puntero a siguiente, del nuevo nodo, apunte donde apunta el nodo con el dato de referencia.
5) Hacer que el puntero a anterior del nuevo nodo, apunte donde señala en puntero anterior del nodo de referencia.
6) Hacer que el puntero a anterior, del nodo de referencia, apunte al nuevo nodo.
7) Si hay un nodo antes del que se está insertando, su puntero a siguiente debe apuntar al nuevo nodo.

Eliminación un elemento



Ojo 1:
Este es el caso general, cuando se elimina un nodo intermedio. Pero hay casos particulares: los extremos y lista con un solo nodo.

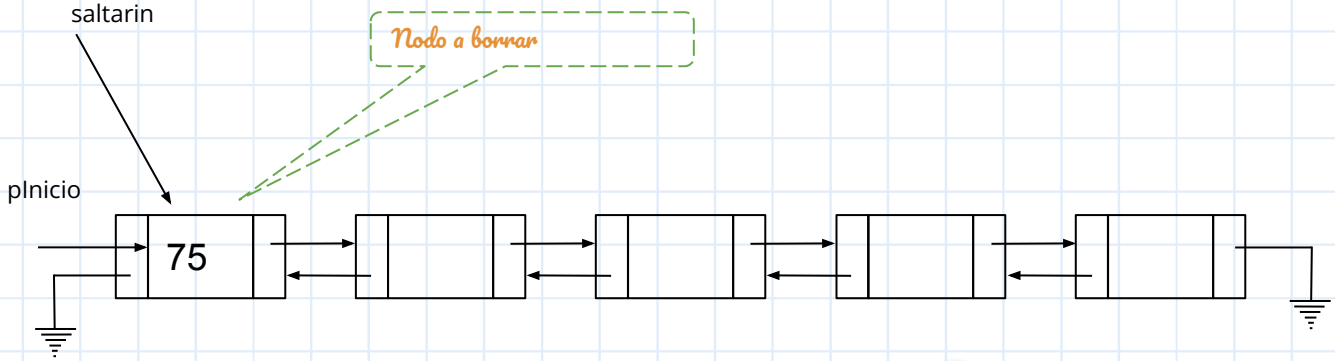
Algoritmo

Hay que considerar los siguientes pasos:

Una vez la función recibe el dato a borrar:

1) Verificar si la lista está vacía: Nada que hacer.
2) Si la lista no está vacía, ubicar un puntero auxiliar al inicio (saltarin) y saltar hasta encontrar el nodo o salirse de la lista.
3) Si saltarín se sale de la lista, el dato no está en la misma.
4) Si saltarin queda encima de algún nodo, entonces allí está el dato a borrar, pero hay varias posibilidades...

Eliminación un elemento

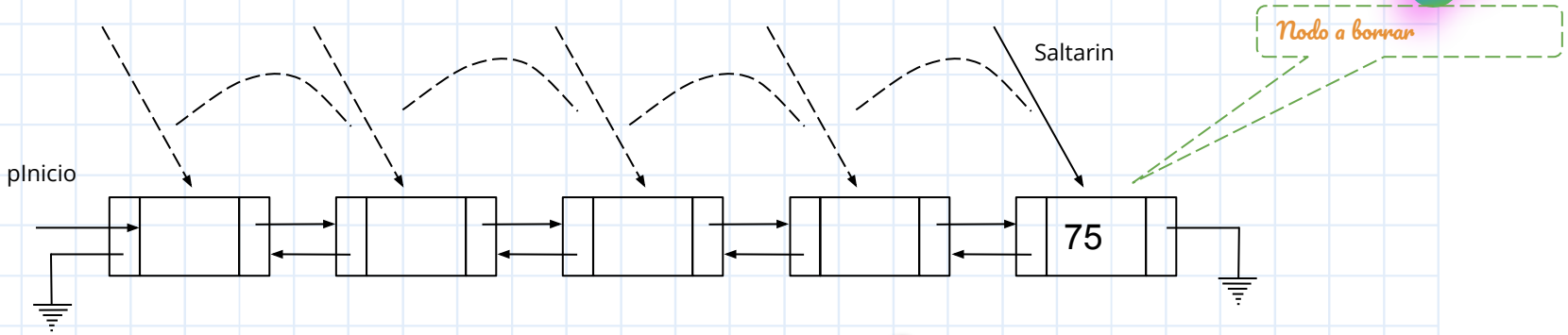


Si el nodo a borrar es el primero de la lista

Algoritmo

1) pInicio avanza al siguiente nodo.
2) pInicio->ant se pone a NULL.
3) Se le da delete a saltarin.

Eliminación un elemento

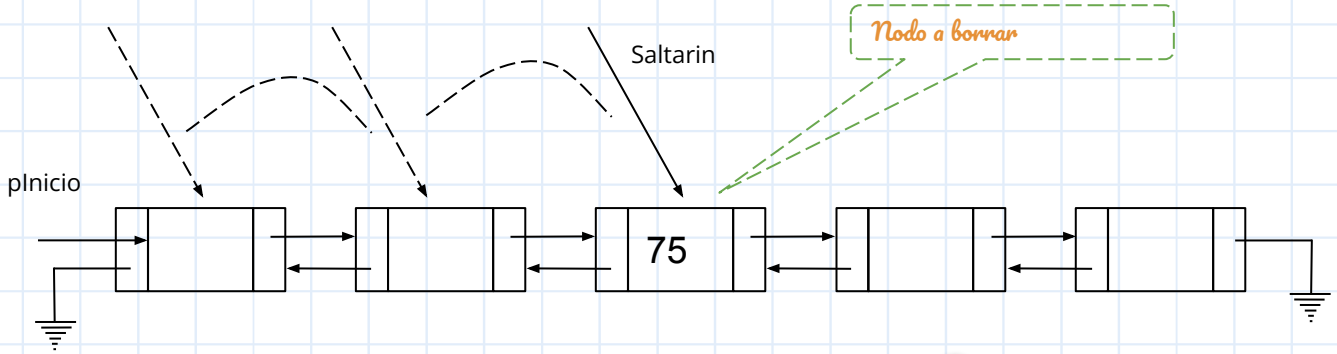


Si el nodo a borrar es el último de la lista

Algoritmo

1) El puntero a siguiente del penúltimo nodo se pone a NULL.
2) Se le da delete a saltarin.

Eliminación un elemento

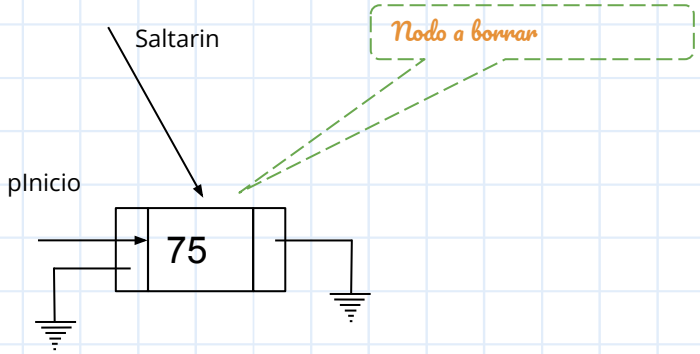


Nodo en el medio (caso más común - caso genérico)

Algoritmo

1) El puntero a siguiente, del nodo anterior, se hace apuntar al siguiente nodo.
2) El puntero a anterior, del nodo siguiente, se hace apuntar al nodo anterior.
3) Se le da delete a saltarin.

Eliminación un elemento



En el caso de que el nodo a borrar sea el único de la lista

Algoritmo

1) pInicio se pone a NULL.
2) Se le da delete a saltarin.