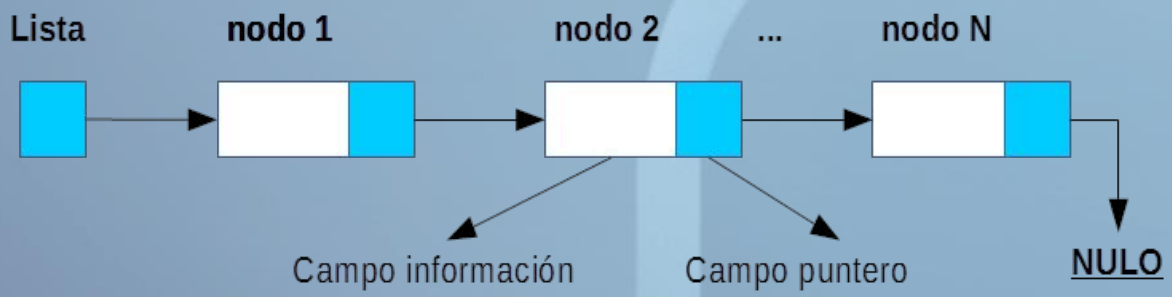
Definición

- Lista lineal de elementos (nodos), se autoreferencian mediante punteros o sus direcciones.

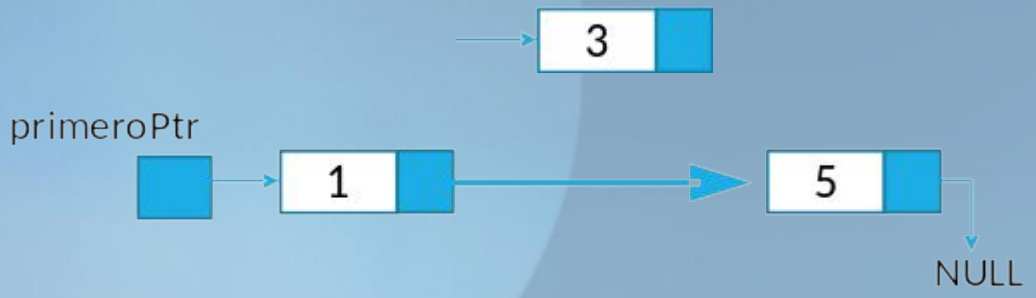
- Accederemos a la lista enlazada mediante el puntero al primer elemento.
- El puntero al siguiente elemento, del último elemento en la lista, tendrá valor nulo

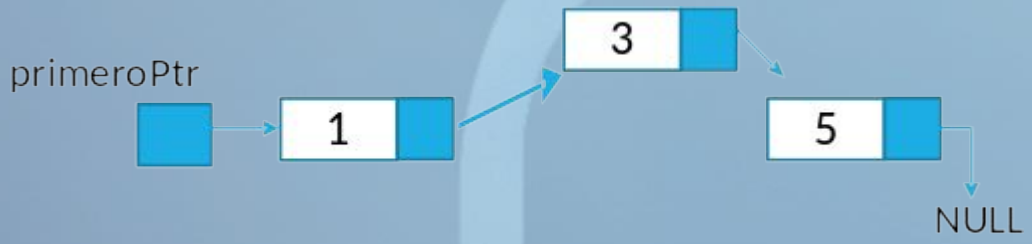


Características

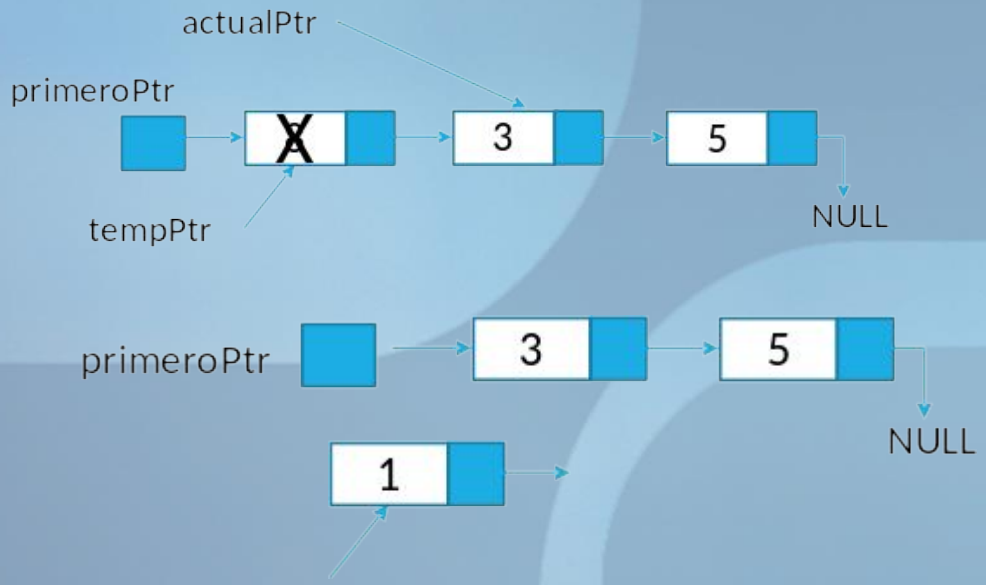
- La lista siempre estará ordenada, sea ascendente o descendentemente.
- Puedo extraer un elemento de la lista sin importar la posición del mismo en ella.

Inserción en la lista





Eliminar un elemento



```
#include <stdio.h>
#include <stdlib.h>

typedef struct nodo {
    int dato ;
    struct nodo * sgte ;
} lista ;
```

```
void listar ( lista * i ) {

    if ( i != NULL ) {
        printf ( " elemento := %d \n " , i->dato ) ;
        listar (i->sgte ) ;
    }

}
```

```
void insertar ( int d , lista ** i ) {

   if (* i == NULL ) {
      printf ( " Insertando %d en la lista \n " , d ) ;
      * i = ( lista *) malloc ( sizeof ( lista ) ) ;
      (* i ) -> dato = d ;
   } else {
      if ((* i ) -> dato > d ) { // cuando va al inicio
      printf ( " Insertando %d en la lista \n " , d ) ;
      lista * nuevo = ( lista *) malloc ( sizeof ( lista ) ) ;
      nuevo-> dato = d ;
      nuevo->sgte = * i ;
      *i = nuevo ;
    } else if ((* i )->dato < d && (* i )->sgte != NULL &&
                  (* i)->sgte-> dato > d ) { // cuando esta entre dos
          printf ( " Insertando %d en la lista \n " , d ) ;
          lista * nuevo = ( lista *) malloc ( sizeof ( lista ) ) ;
          nuevo->dato = d ;
          nuevo->sgte = (* i ) -> sgte ;
          (* i )->sgte = nuevo ;
    } else if ((* i ) -> dato == d )
          printf ( " Ya existe el elemento %d en la lista \n " , d ) ;
    else
          insertar (d , &(* i ) -> sgte ) ;
    }

}
```

```
void eliminar ( int d , lista ** i ) {

    if (* i == NULL )
        printf ( " No existe el elemento a eliminar \n " ) ;
    else {
        if ((* i )-> dato == d ) { // elimino al primero
        printf ( " Elimino a %d \n " , d ) ;
        lista * aux = (* i ) ;
        (* i ) = (* i ) -> sgte ;
        free ( aux ) ;
    } else if ((* i )-> dato < d && (* i ) -> sgte != NULL &&
                    (* i )-> sgte-> dato == d ) { // elimino al siguiente

            printf ( " Elimino a %d \n " , d ) ;
            lista * aux = (* i )->sgte ;
            (* i )->sgte = (* i )->sgte->sgte ;
            free( aux ) ;
    }
    else
            eliminar (d , &(* i ) -> sgte ) ;
    }
}
```

```
int main () {
    lista * inicio ;
    inicio = NULL ;

    insertar (5 , & inicio ) ;
    insertar (3 , & inicio ) ;
    insertar (4 , & inicio ) ;
    insertar (10 , & inicio ) ;
    insertar (6 , & inicio ) ;
    insertar (8 , & inicio ) ;
    insertar (1 , & inicio ) ;
    insertar (8 , & inicio ) ;
    insertar (1 , & inicio ) ;

    printf ( " Listo elementos \n " ) ;
    listar ( inicio ) ;
    printf ( " Fin del listado \n " ) ;

    eliminar (4 , & inicio ) ;
    eliminar (1 , & inicio ) ;
    eliminar (10 , & inicio ) ;
    eliminar (14 , & inicio ) ;

    printf ( " Listo elementos \n " ) ;
    listar ( inicio ) ;
     printf ( " Fin del listado \n " ) ;

    return 0;
}
```