# Hoja de Trabajo 1

## ¿Qué es un TDA?

Es una abstracción de una estructura de dato, la cual especifica lo que se puede almacenar dentro de él y que operaciones se pueden realizar sobre el mismo. Las operaciones más comunes son: búsqueda, inserción, extracción, ordenamiento, entre otros.

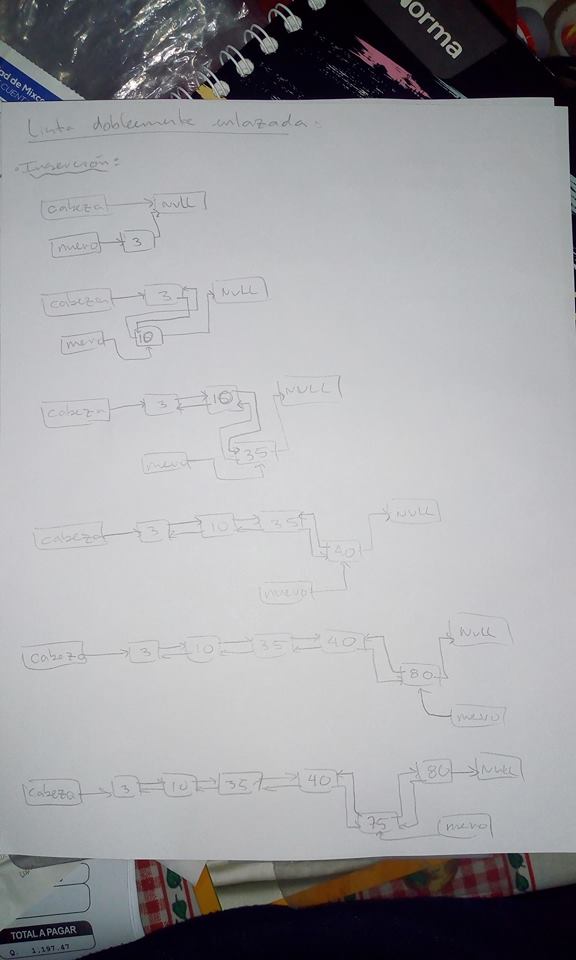
## ¿Para qué se utiliza el nodo pivote?

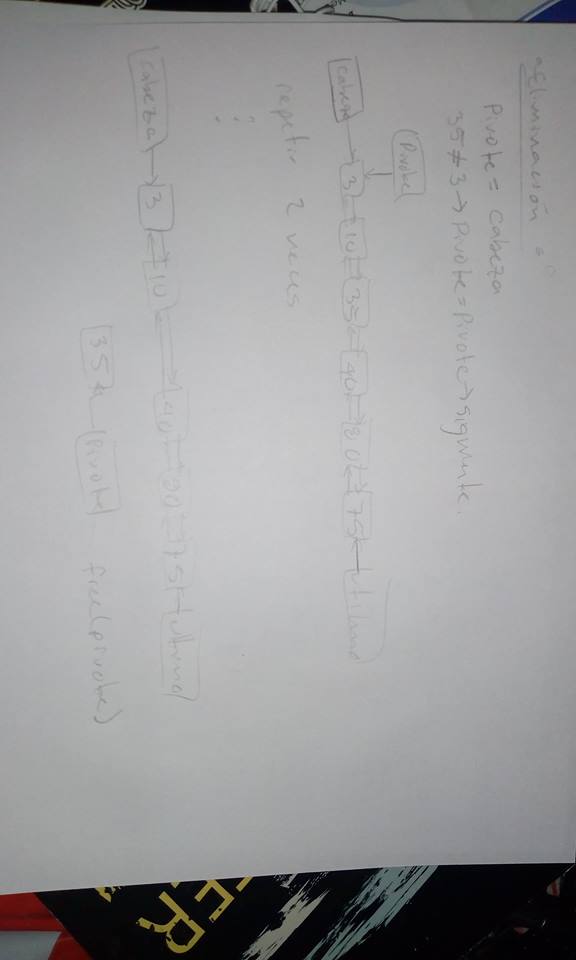
El nodo pivote se utiliza para recorrer una estructura e indicar en qué posición se encuentra.

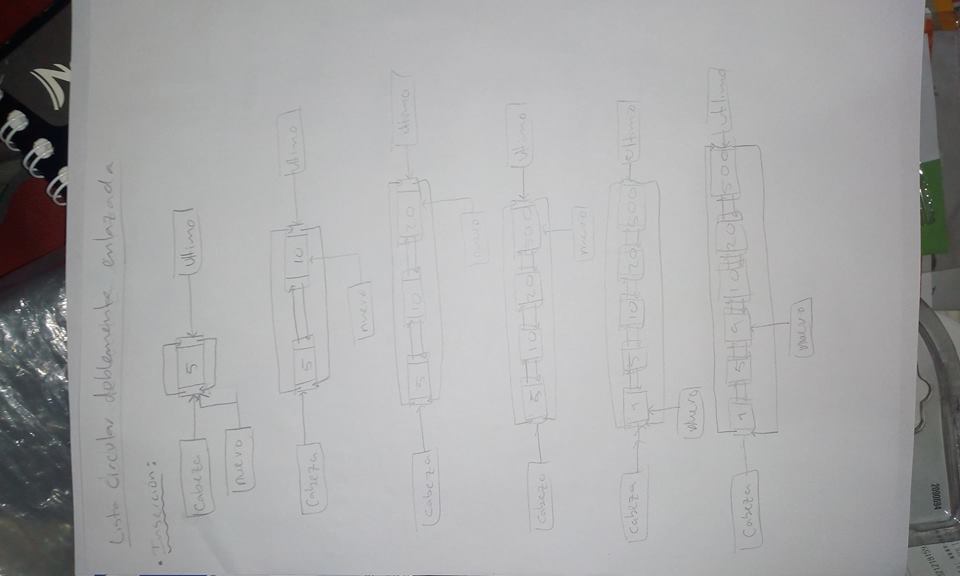
## ¿Para qué se utiliza el nodo cabeza?

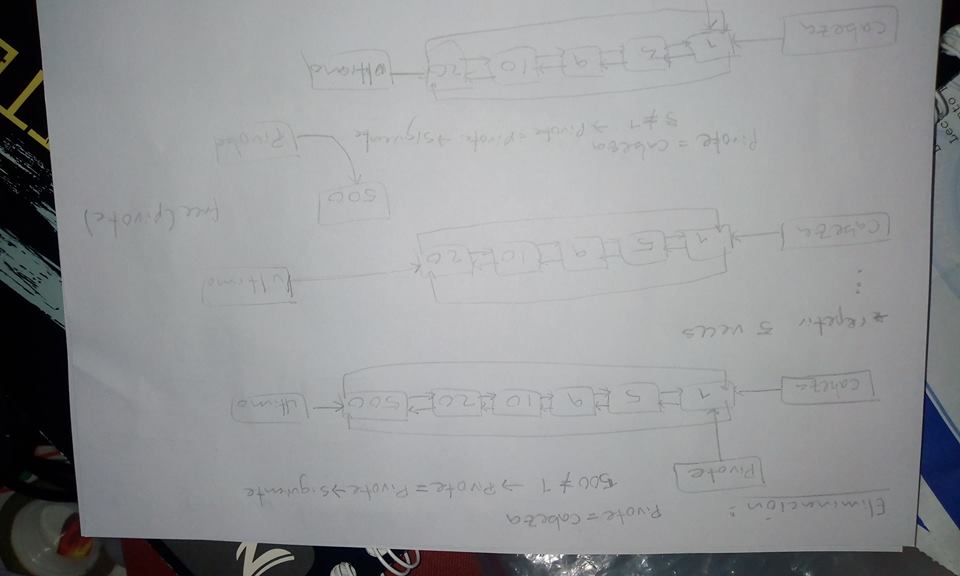
El nodo cabeza se utiliza para indicar el comienzo de una lista. Este no almacena ningún valor, solo apunta al primer elemento.

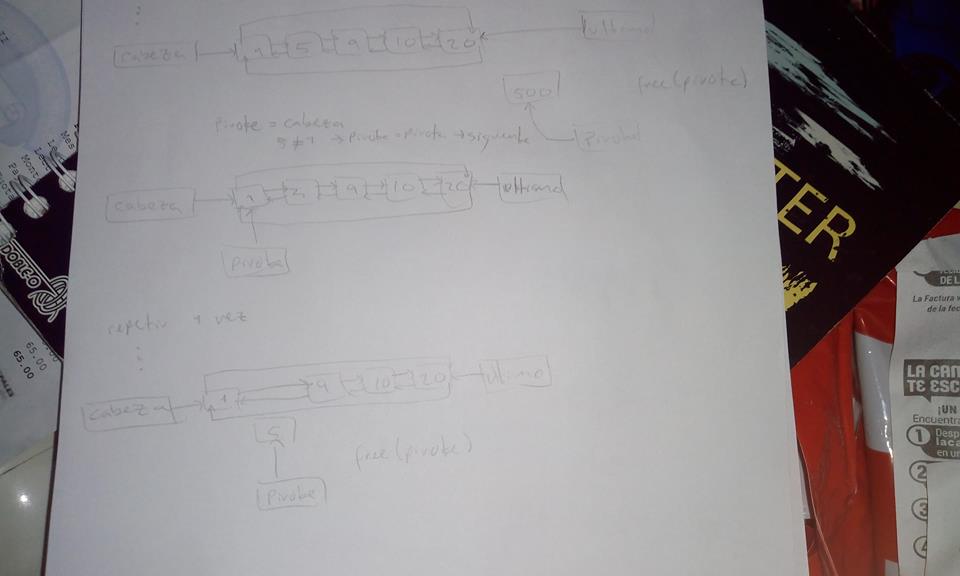
|  |  |  |  |
| --- | --- | --- | --- |
| Estructuras dinámicas | | Estructuras estáticas | |
| Ventajas | **Desventajas** | **Ventajas** | **Desventajas** |
| Se puede manejar más espacio en memoria. | Es más lento acceder al elemento.  No hay acceso directo. | Permiten acceder más rápido al elemento de la lista.  Hay acceso directo. | El espacio en memoria ya está definido y no se puede alterar |











## Código

typedef struct nodo {

int id;

char valor;

struct nodo \*siguiente;

struct nodo \*anterior;

} nodo;

void insertarLSE()

{

nodo \*nuevo\_nodo = (nodo\*)malloc(sizeof(nodo));

cout<<"\n Ingrese un id:"<<endl;

cin>>nuevo\_nodo->id;

cout<<"\n Ingrese un valor:"<<endl;

cin>>nuevo\_nodo->valor;

if(p == NULL){

p = nuevo\_nodo;

p->siguiente = p;

u = nuevo\_nodo;

}

else if(p->siguiente == p){

if(nuevo\_nodo->id < p->id){

nuevo\_nodo->siguiente = u;

p = nuevo\_nodo;

}

else if(nuevo\_nodo->id >= p->id){

p->siguiente = nuevo\_nodo;

nuevo\_nodo->siguiente = p;

u = nuevo\_nodo;

}

}

else{

nodo \*aux = (nodo\*)malloc(sizeof(nodo));

aux = p;

do{

if(nuevo\_nodo->id < p->id){

nuevo\_nodo->siguiente = p;

p = nuevo\_nodo;

u->siguiente = p;

break;

}

else if(nuevo\_nodo->id >= aux->id && aux->siguiente == p){

u->siguiente = nuevo\_nodo;

nuevo\_nodo->siguiente = p;

u = nuevo\_nodo;

break;

}

else if(nuevo\_nodo->id >= aux->id && nuevo\_nodo->id <= aux->siguiente->id){

nuevo\_nodo->siguiente = aux->siguiente;

aux->siguiente = nuevo\_nodo;

break;

}

else{

aux = aux->siguiente;

}

}while(aux != p);

}

cout<<"\n Id: "<<nuevo\_nodo->id<<" Valor: "<<nuevo\_nodo->valor<<endl;

}

Void imprimir(){

nodo \*aux = (nodo\*)malloc(sizeof(nodo));

aux = ultimo;

while(aux != NULL){

cout<<"\n Id: "<<aux->id<<" Valor: "<<aux->valor<<endl;

aux = ultimo->anterior;

}

}