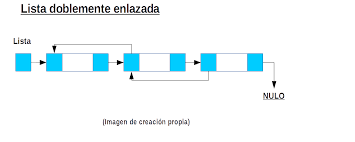
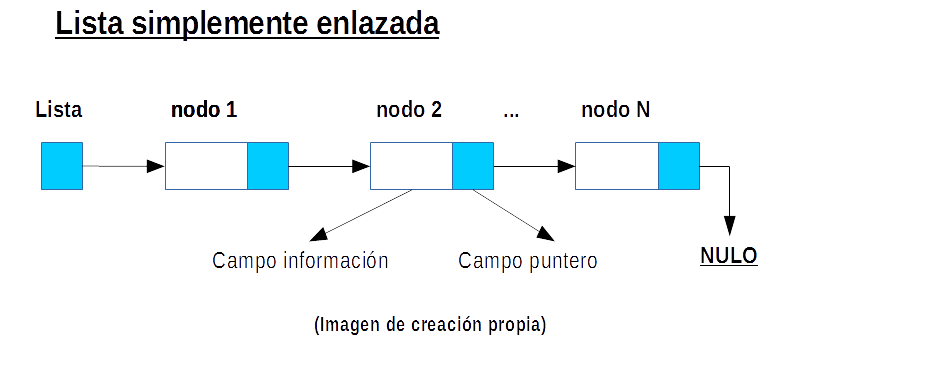
**¿Qué es una lista enlazada?**   
Una lista enlazada es un número de nodos, los cuales poseen 2 campos, un valor, por ejemplo INT y un apuntador al siguiente nodo.

**¿Qué es un nodo?**

Un nodo es una combinación de dos partes:

un tipo de dato (entero, real, doble, carácter o tipo predefinido) y

un enlace (apuntador/puntero) al siguiente nodo.



**Operaciones en Listas Enlazadas:**

1. [Insertar elementos](https://www.youtube.com/watch?v=cPcDuIUqEO4&ab_channel=Programaci%C3%B3nATS)

struct Nodo{

int dato;

Nodo \*siguiente;

}

int main(){

Nodo \*lista = NULL;

return 0;

}

void insertarLista(Nodo \*&lista, int n){

Nodo \*nuevo\_nodo = new Nodo(); //creamos nuevo nodo

nuevo\_nodo->dato = n; //asignamos un dato al nuevo nodo

Nodo \*aux1 = lista;  
 Nodo \*aux2;

//bucle while para siempre tener ordenada la lista

WHILE((aux1 != NULL) && (aux1->dato < n){

aux2 = aux1;

aux1 = nuevo\_nodo;

}

// IF se cumple si el elemento va al principio de la lista

IF(lista == aux1){

lista = nuevo\_nodo; //nuevo\_nodo pasa a ser el 1°elemento de la list

}ELSE{

//este else se cumple si se ha entrado al while

aux2->siguiente = nuevo\_nodo;

}

nuevo\_nodo->siguiente = aux1; //marcamos la ultima posicion como NULL

}

1. [Mostrar elementos](https://www.youtube.com/watch?v=lbp-sAvM9D8&ab_channel=Programaci%C3%B3nATS) – 3 pasos
   1. Crear un nuevo nodo
   2. igualar ese nuevo\_nodo a lista
   3. recorrer lista

void mostrarLista(Nodo \*lista){

//paso 1

Nodo \*actual = new Nodo();

// paso 2

actual = lista;

//paso 3

while(actual != NULL){

std::cout<<actual->dato<<" ->";

actual = actual->siguiente;

}

}

1. [Buscar un elemento](https://www.youtube.com/watch?v=W86jnUUY2zA&ab_channel=Programaci%C3%B3nATS) // 4 pasos
   1. Crear un nuevo nodo (actual)
   2. igualar ese nuevo nodo a la lista
   3. recorrer la lista
   4. determinar si ese elemento existe en la lista

void buscarLista(Nodo \*lista, int n){

bool bandera = false;

Nodo \*actual = new Nodo();

actual = lista;

while((actual != NULL) && (actual->dato <= n)){

if(actual->dato == n){

bandera=true;

}

actual = actual->siguiente;

}

if(bandera == true){

std::cout<<"Elemento "<<n<<" Si a sido encontrado en la lista\n";

}

else{

std::cout<<"Elemento "<<n<<" No a sido encontrado en la lista";

}

}

1. [Eliminar un elemento](https://www.youtube.com/watch?v=ljYbVM6j11s&ab_channel=Programaci%C3%B3nATS) // 5 pasos
   1. preguntar si la lista está vacía
   2. crear un \*aux\_borrar y \*anterior = NULL
   3. Igualar \*aux\_borrar al inicio de la lista
   4. Recorrer la lista
   5. Eliminar el elemento

void eliminarNodo(Nodo \*&lista, int n){

//preguntamos si la lista esta vacia

if(lista != NULL){

Nodo \*aux\_borrar;

Nodo \*anterior = NULL;

aux\_borrar = lista;

//recorrer la lista

while((aux\_borrar != NULL) && (aux\_borrar->dato != n)){

anterior = aux\_borrar;

aux\_borrar = aux\_borrar->siguiente;

}

//el elemnto no ha sido encontrado

if(aux\_borrar == NULL){

std::cout<<"\nEl elemento " <<n<<" no existe\n";

}

//El primer elemento es el q vamos a borrar

else if(anterior == NULL){

lista = lista->siguiente;

delete aux\_borrar;//se elimina el primero

}

//El nodo esta en la lista pero no es el primero

else{

anterior->siguiente = aux\_borrar->siguiente;

delete aux\_borrar;

}

}

}