Estructura de Datos, Unahur

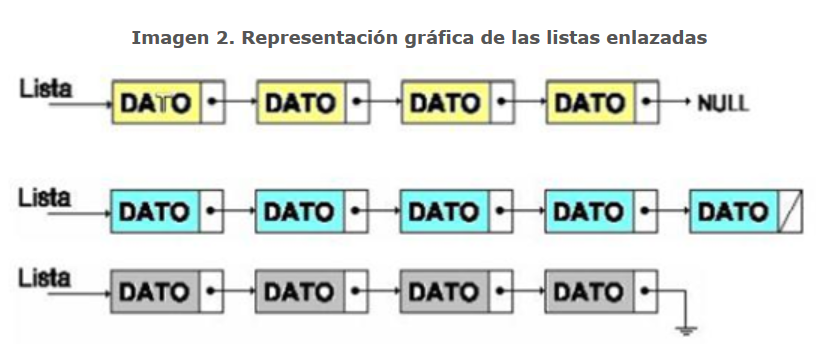
**Listas Dinamicas**

Prof. Sergio Gonzalez

Prof. Ariel Clocchiatti

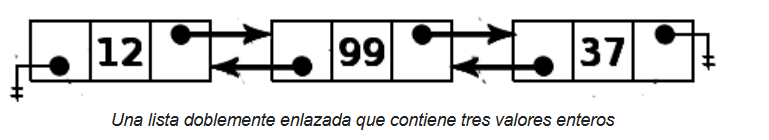
**Lista Enlazada:**

Consiste en una secuencia de [nodos](https://es.wikipedia.org/wiki/Nodo_(inform%C3%A1tica)), en los que se guardan campos de datos arbitrarios y una referencia o [puntero](https://es.wikipedia.org/wiki/Puntero_(inform%C3%A1tica)) al nodo siguiente. El principal beneficio de las listas enlazadas respecto a los [vectores](https://es.wikipedia.org/wiki/Vector_(inform%C3%A1tica)) o arrays convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.



Las listas enlazadas permiten inserciones y eliminaciones de nodos en cualquier punto de la lista, pero el acceso es secuencial no aleatorio. Una lista enlazada es un tipo de dato autorreferencial porque contiene punteros o enlaces a otros datos del mismo tipo (nodos).

Las listas cuyos nodos tienen punteros al siguiente nodo se llaman simplemente enlazadas. Aquellas listas cuyos nodos tienen punteros al nodo siguiente y nodo anterior, se llaman doblemente enlazada.



También existen las listas circulares, tal como se muestra a continuación:

[Circularly-linked-list.svg](https://commons.wikimedia.org/wiki/File:Circularly-linked-list.svg)  
*Una lista enlazada circular que contiene tres valores enteros*

**Operaciones de la estructura de datos Lista:**

* **Crear:** Crea una lista vacía.
* **Insertar:** Inserta el elemento en la posición dada, desplazando los elementos restantes una posición adelante.
* **EliminarElemento:** Elimina el elemento en la posición dada, desplazando los restantes una posición hacia atrás.
* **VaciarLista:** Vacía la lista, eliminando todos los elementos.
* **EstaVacia:** DevuelveTrue, si la lista no tiene elementos, de lo contrario False.
* **Tamaño:** Devuelve la cantidad de elementos en la lista.
* **AgregarElemento:** Agrega un elemento al final de la lista.
* **ObtenerElemento:** Devuelve el elemento de la posición dada.

**Veamos algunas implementaciones en Python:**

Primero definimos el Nodo para la lista:

class NodoLista:

def \_\_init\_\_(self, dato = None):

self.dato = dato

self.siguiente = None

**Crear la lista:**

class Lista:

def \_\_init\_\_(self):

self.primero = None

**EstaVacia**:

def isEmpty(self):

return self.primero == None

**Tamaño:**

def getLen(self):

cant = 0

aux = self.primero

while aux != None:

cant += 1

aux = aux.siguiente

return cant

'''

La lista podría tener un contador interno, que se incrementa y decrementa en uno, cuando se inserta, elimina un nodo respectivamente.

'''

**ObtenerElemento:**

class Lista:

def getDato(self, pos = 0):

if pos < self.getLen() and pos >= 0:

i = 0

aux = self.primero

while i < pos:

i += 1

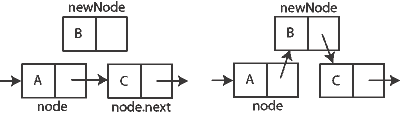
aux = aux.siguiente

return aux.dato

else:

raise Exception("La posicion no existe en la lista.")

**Insertar:** El siguiente código inserta un elemento a continuación de otro en una lista simple. El diagrama muestra cómo funciona.

[](https://commons.wikimedia.org/wiki/File:Singly_linked_list_insert_after.png)

def insert(self, dato = None, pos = 0):

if pos >= 0:

nuevo = NodoLista(dato)

if self.isEmpty():

self.primero = nuevo

else:

if pos == 0:

nuevo.siguiente = self.primero

self.primero = nuevo

else:

i = 0

aux = self.primero

while i < pos-1 and aux.siguiente != None:

i += 1

aux = aux.siguiente

nuevo.siguiente = aux.siguiente

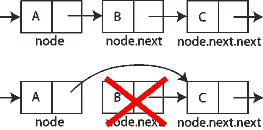
aux.siguiente = nuevo

else:

raise Exception("Posicion incorrrecta.")

**EliminarElemento:**

De forma similar, también tenemos funciones para borrar un nodo dado ó para borrar un nodo del principio de la lista. Ver diagrama.

[](https://commons.wikimedia.org/wiki/File:Singly_linked_list_delete_after.png)

def delete(self, pos = 0):

if not self.isEmpty() and pos >= 0:

if pos == 0:

self.primero = self.primero.siguiente

else:

i = 0

aux = self.primero

while i < pos-1 and aux.siguiente != None:

i += 1

aux = aux.siguiente

if aux.siguiente != None:

aux.siguiente = aux.siguiente.siguiente

else:

if self.isEmpty():

raise Exception("Lista vacia.")

else:

raise Exception("Posición incorrecta.")