UNRaf

UNIVERSIDAD NACIONAL DE RAFAELA

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar

#IC

Ingeniería en Computación

ALGORITMOS Y ESTRUCTURAS DE DATOS

ALGORITMOS Y ESTRUCTURAS DE DATOS

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

AGENDA

#IC

- Repaso TDP
- TDA
  - Listas enlazadas simples
  - Listas enlazadas dobles
  - Listas enlazadas circulares
- Ejemplos

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

Tipos de Datos

#IC

Tipos de Datos Primitivos (TDP):

- **Números (enteros, reales)**
- **Caracteres (cadenas de caracteres)**
- **Lógicos (booleanos)**

Tipos de Datos Abstractos (TDA)

Estructuras que pueden combinar distintos TDP (y otros TDA), y poseen comportamiento propio.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

¿Por qué TDA?

#IC

La memoria RAM está compuesta por bits (0s y 1s), agrupados en unidades más grandes, como los bytes (8 bits).

Al tener una enorme cantidad de bytes, la computadora guarda donde está almacenado cada dato usando direcciones de memoria.

Al saber dónde está almacenado, se puede acceder rápidamente a cada variable y las variables pueden modificar su contenido sin cambios grandes en el código.

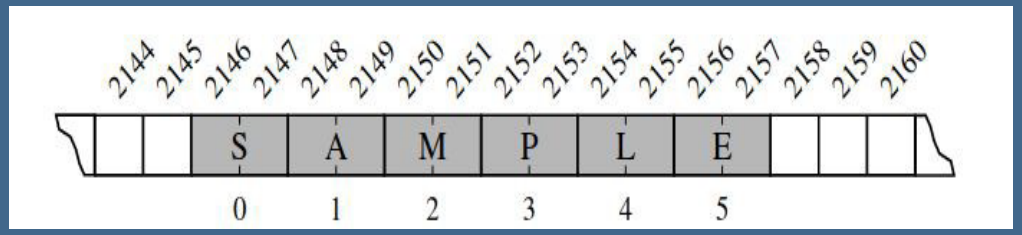
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

ARRAYS

#IC

Grupo de variables relacionadas que pueden guardarse de forma continua en la memoria.

Todos los datos de un array deben ser del mismo tipo, para calcular fácilmente cuántos bytes son necesarios para almacenar todo un array y encontrar un bloque de memoria disponible de ese tamaño.

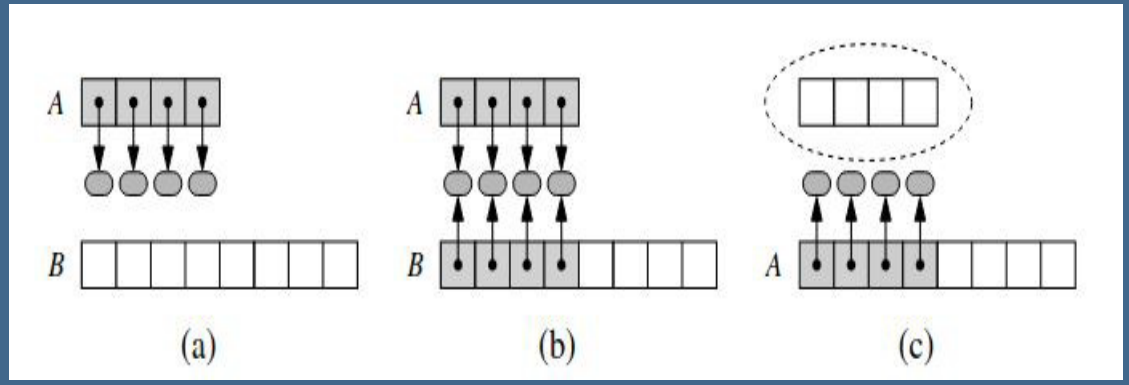


UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

ARRAYS DINÁMICOS

#IC

Por definición un array no puede cambiar de tamaño, ya que puede sobreescribir fragmentos de memoria que estén siendo usados. Para agregar elementos a un array se debe buscar un nuevo espacio en memoria que permita almacenar todos los datos, crear un nuevo array y eliminar el anterior. En Python, la clase list es un ejemplo de array dinámico.

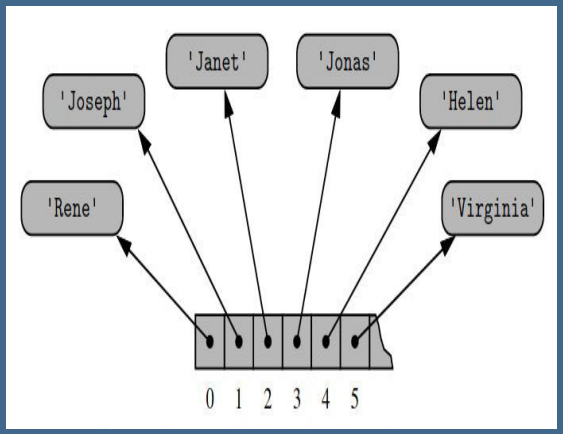


UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

ARRAYS REFERENCIADOS

#IC

Es posible almacenar dentro de un array elementos de distintos tamaños. En lugar de guardar el elemento, se almacena la dirección de memoria donde se encuentra ese elemento. Todas las direcciones de memoria tienen el mismo tamaño, por lo que se cumple la condición necesaria que define un array. Una variable que almacena una dirección de memoria se la denomina puntero.



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS EN PYTHON

#IC

Las listas de Python son una clase sumamente optimizada y, usualmente, es la mejor opción para almacenar datos. Sin embargo, posee algunas limitaciones:

- **La memoria asignada al comienzo es siempre mayor al número de elementos para facilitar el agregado de valores en los extremos.**
- **Agregar y quitar elementos en el interior de la lista es costoso**
- **Sistemas con poca memoria o memoria muy fragmentada pueden encontrarse con problemas para encontrar espacio suficiente para almacenar la lista completa.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS ENLAZADAS

#IC

Añadir o quitar elementos en una lista es una operación muy costosa

¿Qué cambios en la estructura de datos podrían facilitar estos procesos?

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

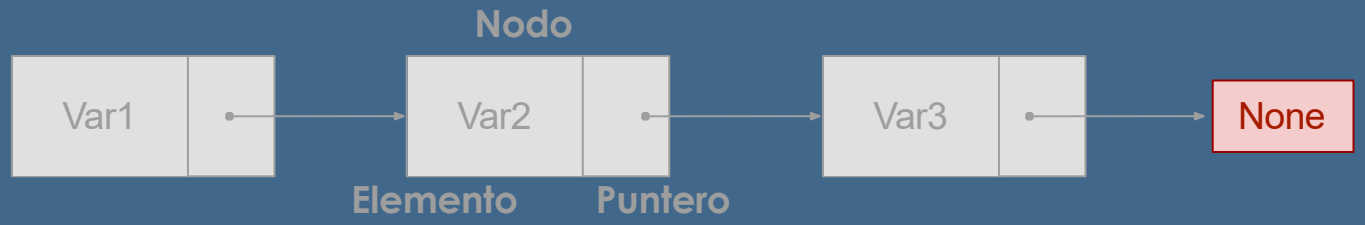
LISTAS ENLAZADAS

#IC

Los arrays responden a una representación centralizada de los datos: una gran porción de memoria capaz de almacenar todos los datos (y más).

Las listas enlazadas se basan en una representación distribuida de los datos.

Estas listas se conforman de nodos que almacenan un elemento de la lista y la dirección de memoria, un puntero, de uno o más elementos aledaños.



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS SIMPLES ENLAZADAS (LSE)

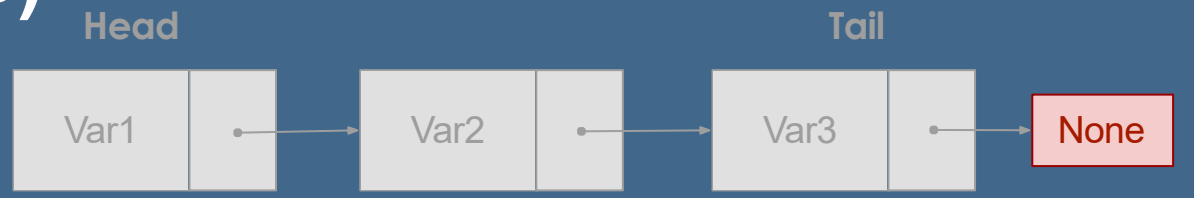
#IC

Versión más simple de las listas enlazadas.

Cada nodo posee dos valores:

- **una referencia a un elemento de la secuencia**
- **un puntero al nodo siguiente**

El primer elemento de la lista se conoce como cabeza (o head) mientras que el último se denomina cola (o tail). El elemento tail siempre apunta a un elemento nulo (None en Python o Null en otros lenguajes)



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

RECORRIDO LSE

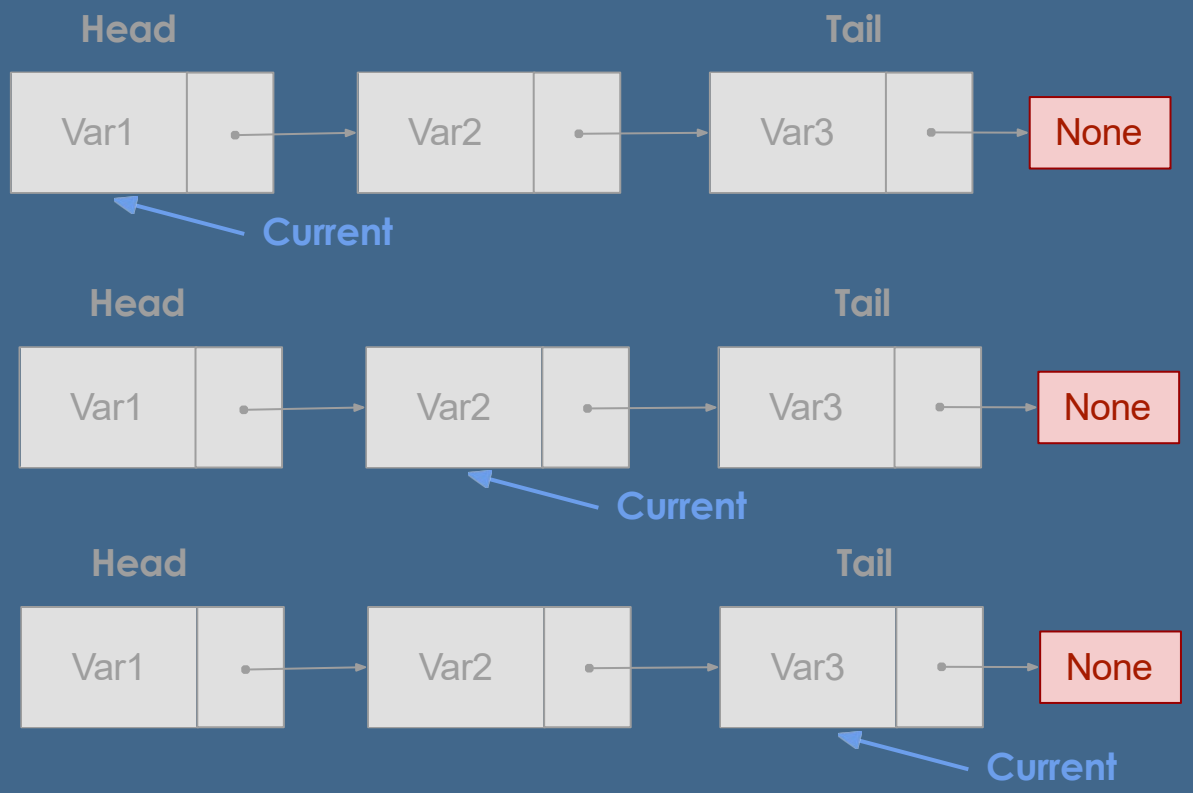
#IC

- Para recorrer una lista completa, siempre (y solo) se comienza desde el elemento head.
- Moviéndose de un nodo al siguiente es posible recorrer toda la lista, hasta alcanzar el valor None.
- Los distintos nodos de la lista se visitan usando el puntero current.
- Se debe definir un método que mueva el puntero current a la posición siguiente.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

RECORRIDO LSE

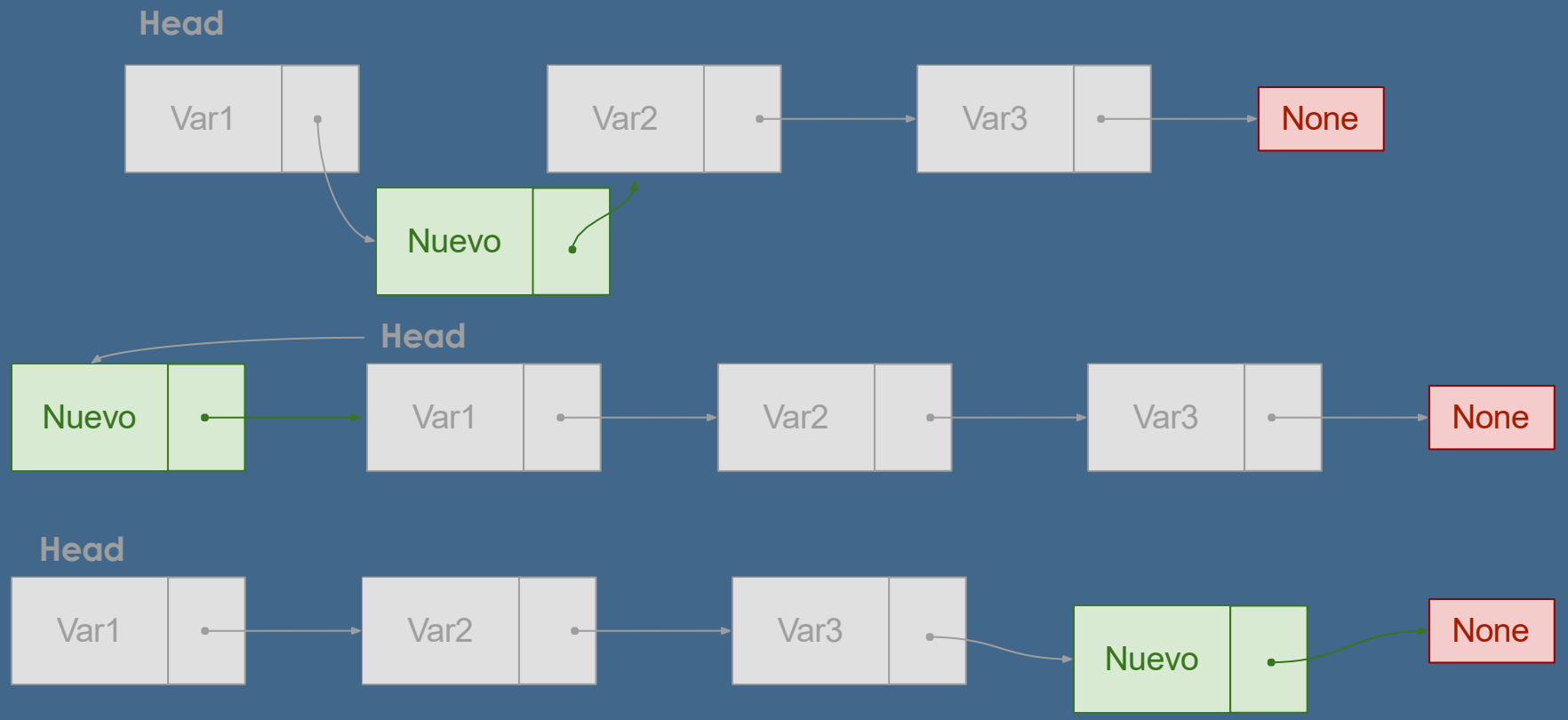
#IC



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

AÑADIR ELEMENTO A LSE

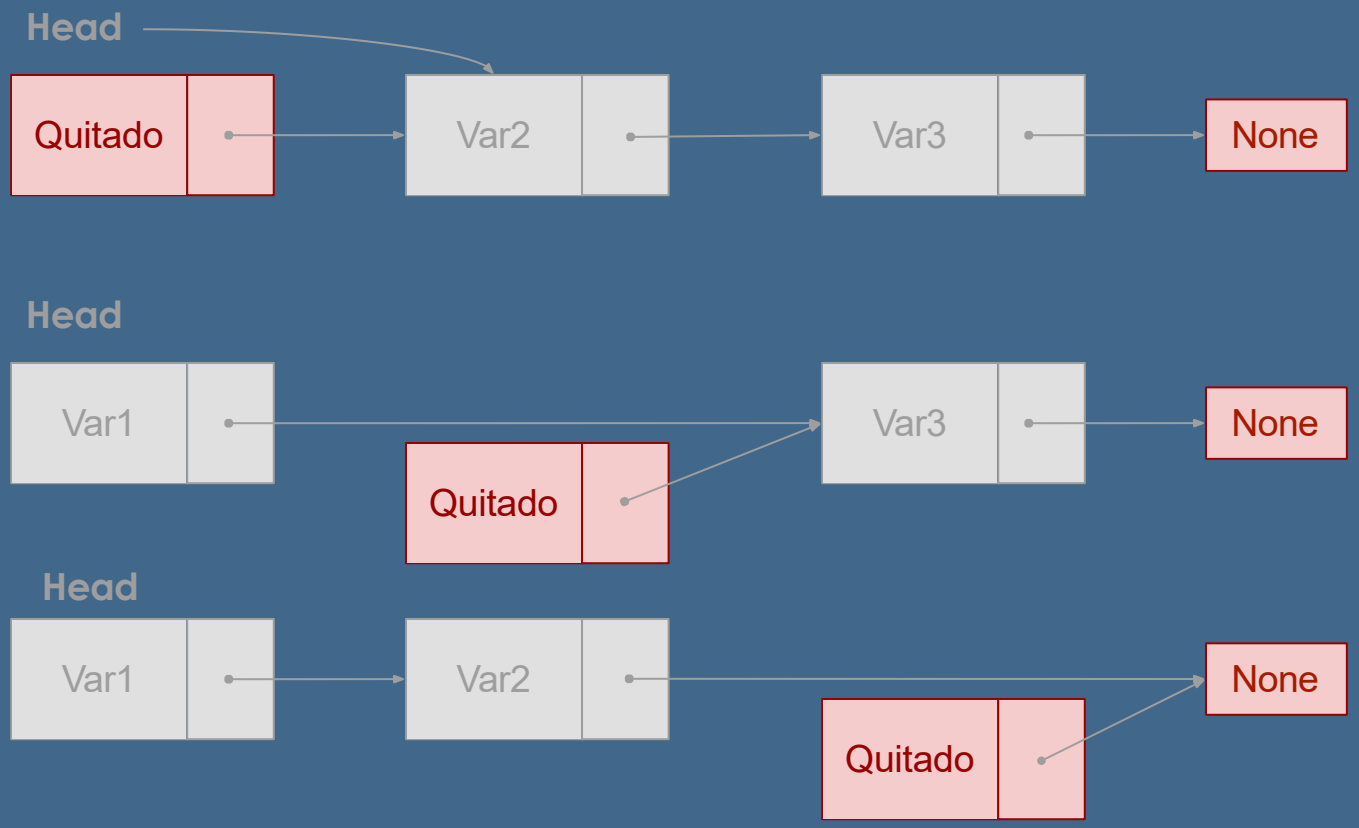
#IC



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

QUITAR ELEMENTO A LSE

#IC



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS SIMPLES ENLAZADAS

#IC

```
class Nodo:
    def __init__(self, dato):
        self.dato = dato
        self.siguiente = None
```

```
class ListaEnlazadaSimple:
    def __init__(self):
        self.cabeza = None

    def insertar(self, dato):
        nuevo_nodo = Nodo(dato)
        nuevo_nodo.siguiente = self.cabeza
        self.cabeza = nuevo_nodo

    def __len__(self):
        return self._len

    def vacia(self):
        if self._len == 0:
            return True
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS SIMPLES ENLAZADAS

#IC

```
def buscar(self, clave):
        actual = self.cabeza
        while actual:
            if actual.dato == clave:
                return True
            actual = actual.siguiente
        return False
```

```
def eliminar(self, clave):
    actual = self.cabeza
    anterior = None
    while actual:
        if actual.dato == clave:
            if anterior:
                anterior.siguiente = actual.siguiente
            else:
                self.cabeza = actual.siguiente
            return True
        anterior = actual
        actual = actual.siguiente
    return False
```

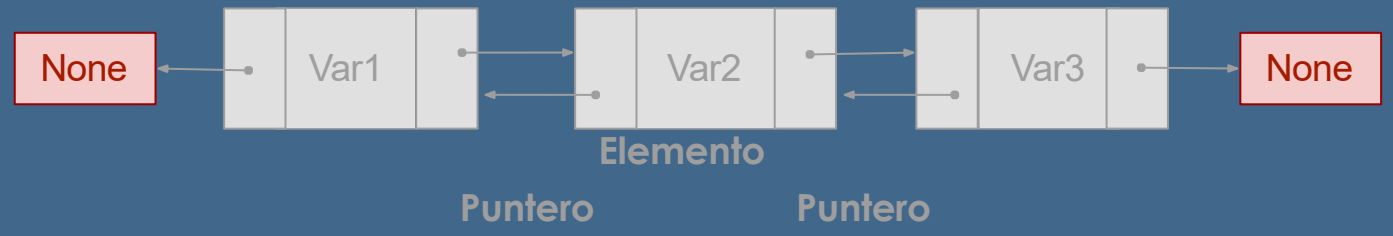
UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS DOBLES ENLAZADAS (LDE)

#IC

Las LSE no permiten el recorrido inverso de una lista y la eliminación de elementos en el interior de la lista es complejo.

Para solucionar estos problemas es posible agregar un segundo puntero en cada nodo que apunte al elemento anterior en la lista. Estas estructuras se denominan listas doblemente enlazadas.



UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS DOBLES ENLAZADAS

#IC

```python
class NodoDoble:
    def __init__(self, dato):
        self.dato = dato
        self.anterior = None
        self.siguiente = None
```

```python
class ListaDoble:
    def __init__(self):
        self.cabeza = None
        self.cola = None

    def insertar_al_final(self, dato):
        nuevo_nodo = NodoDoble(dato)
        if not self.cola:
            self.cabeza = nuevo_nodo
            self.cola = nuevo_nodo
        else:
            self.cola.siguiente = nuevo_nodo
            nuevo_nodo.anterior = self.cola
            self.cola = nuevo_nodo
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS DOBLES ENLAZADAS

#IC

```
def eliminar_del_principio(self):
    if not self.cabeza:
        return None
    dato = self.cabeza.dato
    self.cabeza = self.cabeza.siguiente
    if self.cabeza:
        self.cabeza.anterior = None
    else:
        self.cola = None
    return dato
```

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS CIRCULARES (LC)

#IC

- **Son similares a las LSE, pero el último elemento apunta al nodo head.**
- **No tienen un comienzo y fin como las LSE.**
- **Las LC no poseen ni inicio ni fin estrictos, por lo que es necesario mantener una referencia a algún nodo para poder recorrer la lista. Este puntero puede llamarse head.**

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

LISTAS CIRCULARES (LC)

#IC

- El recorrido es similar al de las LSE, pero se finaliza al llegar al primer elemento recorrido (nodo head).

- Añadir o quitar elementos sigue la misma lógica que para las LSE, actualizando los nodos a los que apunta cada puntero.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

ARRAYS vs LISTAS ENLAZADAS

#IC

Arrays	Listas Enlazadas
Permiten acceder rápidamente al n-ésimo elemento.	Se debe recorrer la lista hasta el n-ésimo elemento.
Agregar elementos en los extremos requiere solo ubicar la posición en memoria (operación aritmética).	Agregar elementos en los extremos requiere redirigir punteros.
Requieren solo memoria para los elementos de la lista.	Requieren memoria para los elementos de la lista y los punteros.
Es costoso mover todos los elementos para insertar/quitar uno del medio.	Una inserción/deleción en el medio de la lista solo requiere cambiar punteros.
Es posible necesitar mover todos los elementos al agregar/quitar elementos.	Una vez definido un nodo, no es necesario moverlo.

UNRaf
UNIVERSIDAD NACIONAL DE RAFAELA

#IC

UNRaf

UNIVERSIDAD
NACIONAL DE
RAFAELA

Bv. J.A. Roca 989 / CP: 2300
Rafaela - Santa Fe - Argentina

T: +54 (03492) **501155**

info@unraf.edu.ar
www.unraf.edu.ar