DOCUMENTO TÉCNICO

# Requisitos

## Especificación

#RESUMEN DEL PROBLEMA (opcional)

Implementar método delete, el valor anterior y estructura de pila y cola a listas enlazadas

### Entrada:

### Una lista x.

### Salida:

# Dicha lista en estructura de lista enlazada con aplicaciones de varios métodos como delete, update, search…

# Diseño

## Estrategia

### Para implementar el método delete se tuvieron en cuenta 3 casos principales.

### Si el elemento a buscar era la cabeza, se debía actualizar la cabeza al elemento siguiente y si no lo había, la lista era vacía (su cabeza y cola es None)

### Si el elemento era la cola, se tenía que actualizar la cola al elemento anterior a este; si no lo había la cabeza y cola pasarían a ser None

### Si el elemento no estaba en la cola ni en la cabeza, se buscaba el elemento hasta encontrarlo y simplemente se eliminaba tomando el dato anterior al elemento y conectándolo al dato posterior al elemento

### Tener el nodo previo afecta de manera directa a la función delete, ya que es mucho más fácil realizar la eliminación. Por ejemplo, si se está eliminando la cola, simplemente toca actualizar a la cola y que tome el valor del elemento anterior a esta y ya está. Si hablamos de las demás funciones no tiene mucho impacto, ya que la función update y search se tenga o no se tenga el valor del elemento anterior, igualmente toca recorrer toda la lista enlazada.

### Para realizar la implementación de la pila y la cola se deben tener claros los conceptos de estas. La pila elimina los elementos que se agregaron de primeras y la cola elimina los elementos que se agregaron de ultimas. Ya teniendo esto claro solo es implementar los métodos de agregar y eliminar.

### Para el método de eliminar se debe tener en cuenta que si se elimina la cabeza, toca actualizarla con el elemento siguiente a esta, como es el caso de la cola, al contrario de la pila , donde si se elimina la cola se debe actualizar la cola con el elemento anterior a esta.

### #Estructuras de datos

### Listas

#ALGORITMO

Programa adjunto

## Casos de prueba

|  |  |  |
| --- | --- | --- |
| **Entrada** | **Justificación** | **Salida** |
| Eliminar un elemento al inicio en una lista de mas de 1 elemento | Si está en la cabeza se debe actualizar la cabeza con el elemento siguiente a esta | Los datos – el dato eliminado |
| Eliminar un elemento al inicio en una lista de 1 elemento | Si está en la cabeza se debe actualizar la cabeza y la cola con None | None |
| Eliminar un elemento al final en una lista de 1 elemento | Si está en la cola se debe actualizar la cabeza y la cola con None | None |
| Eliminar un elemento al final en una lista de mas de 1 elemento | Si está en la cola se debe actualizar la cola con el elemento anterior a esta | None |

# Análisis

# #programa adjunto

# 

## Temporal

La complejidad de este código es aproximadamente y de manera general de n ya que tiene varios procesos que si tienen costo constante, como es el caso de eliminar un elemento al inicio de la lista o agregar un elemento, pero también tiene otros procesos de costo n como es el de acceder al valor de un elemento o eliminar un elemento en una posición x. Por lo tanto, tiene un costo general de n

# Código

*Al igual que el punto anterior pueden guiarse con este ejemplo*

## Documentación

Dentro del código.

## Fuentes

*Las fuentes son el código sobre el cual escribieron el documento*

/arena\_1\_a