INSTITUTO KHIPU

Semestre III

Sesión 13

CARRERA PROFESIONAL

DESARROLLO DE SISTEMAS DE INFORMACION

HERRAMIENTAS DE PROGRAMACION C#

Tema

IMPLEMENTACION DE UNA LISTA ENLAZADA

LISTA ENLAZADA

Las listas enlazadas son una de las estructuras de datos más fundamentales en informática. Se utilizan para almacenar una secuencia de elementos de manera eficiente, sobre todo cuando se necesita flexibilidad para insertar y eliminar elementos dinámicamente. A continuación, te proporciono un análisis detallado sobre las listas enlazadas, sus tipos, ventajas y desventajas, así como los métodos más comunes utilizados para trabajar con ellas.

¿Qué es una Lista Enlazada?

Una lista enlazada es una estructura de datos dinámica que consiste en una colección de elementos llamados nodos. Cada nodo contiene dos componentes principales:

1. **Dato:** Almacena el valor que contiene el nodo.
2. **Referencia:** Almacena la dirección o referencia al siguiente nodo de la lista.

A diferencia de los arreglos o vectores, en los que los elementos están almacenados en posiciones consecutivas de memoria, en una lista enlazada los nodos no necesitan estar contiguos en la memoria. Cada nodo tiene un "puntero" que apunta al siguiente nodo en la secuencia, creando una cadena de nodos que pueden estar dispersos en la memoria.

Componentes de una Lista Enlazada

- **Nodo:** Unidad básica que compone la lista. Cada nodo contiene:
    - **Dato:** El valor almacenado.
    - **Referencia o Puntero:** Un enlace al siguiente nodo en la lista.

- **Cabeza (head):** El primer nodo de la lista enlazada.
- **Cola (tail):** El último nodo, cuya referencia es null en una lista enlazada simple.

Tipos de Listas Enlazadas

Existen varios tipos de listas enlazadas, dependiendo de cómo se gestionan los enlaces entre nodos:

1. Lista Enlazada Simple

En una lista enlazada simple, cada nodo contiene un enlace al siguiente nodo de la lista. El último nodo apunta a null, indicando que no hay más nodos después de él.

- **Ventajas:** Es la implementación más sencilla. Su operación de inserción al inicio es muy eficiente.
- **Desventajas:** El acceso a elementos no es inmediato, ya que es necesario recorrer la lista desde la cabeza hasta el nodo deseado.

```
[Dato] -> [Dato] -> [Dato] -> null
```

Lista simplemente enlazada.

2. Lista Doblemente Enlazada

En una lista doblemente enlazada, cada nodo tiene dos enlaces: uno al siguiente nodo y otro al nodo anterior. Esto permite recorrer la lista en ambas direcciones (hacia adelante y hacia atrás).

- **Ventajas:** Facilita las operaciones de inserción y eliminación de nodos en cualquier parte de la lista, ya que se tiene acceso tanto al nodo anterior como al siguiente.
- **Desventajas:** Requiere más memoria, ya que cada nodo debe almacenar dos punteros.

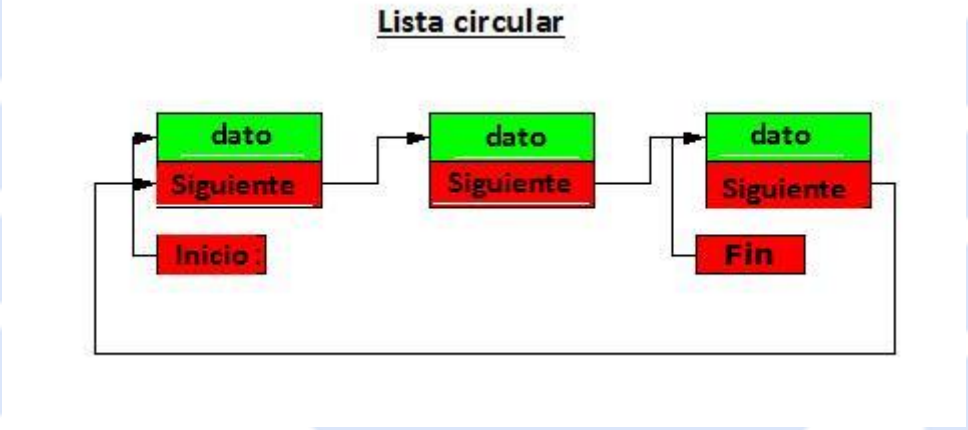
```
null <- [Dato] <-> [Dato] <-> [Dato] -> null
```

3. Lista Enlazada Circular

En una lista enlazada circular, el último nodo apunta de nuevo al primer nodo, formando un ciclo. Puede ser simple o doblemente enlazada, dependiendo de si los nodos tienen uno o dos enlaces.

- **Ventajas:** Útil para representar estructuras circulares, como colas circulares.
- **Desventajas:** Puede ser más complejo de gestionar debido a la posibilidad de ciclos infinitos si no se maneja adecuadamente.

```
[Dato] -> [Dato] -> [Dato] -> [Dato]
  ^                             |
  |-----------------------------|
```



Ventajas y Desventajas de las Listas Enlazadas

Ventajas

1. **Tamaño Dinámico:** Las listas enlazadas pueden crecer o reducirse dinámicamente a medida que se agregan o eliminan nodos, sin necesidad de reestructurar bloques de memoria.
2. **Inserciones y Eliminaciones Eficientes:** Añadir o eliminar elementos al inicio de la lista es rápido (en tiempo constante), ya que no se requiere mover otros elementos.

3. **Flexibilidad:** Se pueden implementar diversas variaciones de listas enlazadas según las necesidades (simples, dobles, circulares).

Desventajas

1. **Acceso Secuencial:** Para acceder a un elemento, es necesario recorrer la lista desde el inicio hasta el nodo deseado. Esto puede ser lento en listas largas, ya que el acceso es O(n), a diferencia de los arreglos, donde el acceso es O(1).
2. **Uso de Memoria Adicional:** Cada nodo requiere almacenamiento adicional para el puntero al siguiente nodo (y al anterior en listas doblemente enlazadas), lo que aumenta el consumo de memoria en comparación con estructuras como los arreglos.

INSTITUTO

KHIPU