UNIVERSIDAD DEL VALLE DE ORIZABA

UNIVO

ALUMNO: TRESS RUIZ ARTURO

MATERIA: ESTRUCTURA DE DATOS

DOCENTE: FLOR DENISSE ARENAS CORTÉS

UNIVERSIDAD DEL VALLE DE ORIZABA

INVESTIGACIÓN SOBRE EL FUNCIONAMIENTO DE PUNTEROS NEXT Y PREV EN LISTAS ENLAZADAS

Introducción

Las **listas enlazadas** son una estructura de datos dinámica fundamental que permite almacenar y organizar elementos (nodos) de forma secuencial, pero no necesariamente contigua en memoria. A diferencia de los arreglos o vectores, las listas enlazadas permiten una gestión eficiente de memoria y operaciones flexibles, como inserciones y eliminaciones en cualquier posición, sin necesidad de mover grandes bloques de datos.

El elemento clave que permite esta flexibilidad es el uso de **punteros**, principalmente next y prev, que conectan los nodos entre sí y definen el orden y dirección del recorrido dentro de la lista.

Definición y Estructura de una Lista Enlazada

Una lista enlazada está compuesta por nodos, donde cada nodo generalmente contiene:

- **Dato o información**: El valor que se almacena en el nodo (por ejemplo, un número, un objeto, etc.).
- **Puntero(s)**: Referencias a otros nodos dentro de la lista.

Tipos básicos de listas enlazadas:

- **Lista simplemente enlazada (Singly Linked List):** Cada nodo tiene un puntero next que apunta al siguiente nodo.
- **Lista doblemente enlazada (Doubly Linked List):** Cada nodo tiene dos punteros: next (siguiente nodo) y prev (nodo anterior).
- **Lista circular:** El último nodo apunta al primer nodo, formando un ciclo, y puede ser simple o doblemente enlazada.

Funcionamiento de los Punteros **next** y **prev**

Puntero next

En una lista enlazada, el puntero next dentro de un nodo almacena la dirección del siguiente nodo en la secuencia. Esto permite **navegar hacia adelante** en la lista desde cualquier nodo dado.

- En una lista simplemente enlazada, el puntero next es el único medio para avanzar.
- El último nodo tiene next apuntando a NULL o nullptr para indicar el fin de la lista.

Puntero prev

El puntero prev aparece en las listas doblemente enlazadas y apunta al nodo anterior en la secuencia.

- Permite **navegar hacia atrás**.
- El primer nodo tiene prev apuntando a NULL o nullptr.
- Facilita operaciones eficientes, como eliminación o inserción, sin necesidad de recorrer la lista desde el inicio.

Ejemplo de estructura de un nodo en C/C++

```c
typedef struct Nodo {
    int dato;
    struct Nodo* next;  // Apunta al siguiente nodo
    struct Nodo* prev;  // Apunta al nodo anterior (solo en listas doblemente enlazadas)
} Nodo;
```

Operaciones básicas utilizando next y prev

1. Recorrido (traversal)

- **Lista simplemente enlazada:**
  Se comienza en el nodo cabeza y se sigue el puntero next hasta encontrar NULL.
- **Lista doblemente enlazada:**
  Se puede recorrer hacia adelante usando next o hacia atrás usando prev.

2. Inserción

- **Al inicio:**
  Se crea un nuevo nodo, se apunta su next al nodo cabeza, y si es doblemente enlazada, el nodo cabeza cambia su prev al nuevo nodo.
- **Al final:**
  Se navega hasta el último nodo (que tiene next igual a NULL), y se apunta su next al nuevo nodo, además de ajustar prev en listas dobles.
- **En medio:**
  Se ajustan punteros next y prev de los nodos adyacentes para incluir el nuevo nodo.

3. Eliminación

- Se actualizan los punteros next y prev de los nodos vecinos para “saltar” el nodo que se elimina, liberando su memoria.

Ventajas del uso de next y prev

- **Flexibilidad:** Permite insertar y eliminar nodos sin necesidad de mover elementos como en un arreglo.
- **Doble navegación:** El puntero prev permite moverse hacia atrás, lo que no es posible en listas simplemente enlazadas.
- **Eficiencia:** Operaciones como eliminación o inserción en listas doblemente enlazadas pueden realizarse en tiempo constante (O(1)) cuando se conoce el nodo, porque no se requiere recorrer desde el inicio.

Consideraciones y desventajas

- **Mayor uso de memoria:** Las listas doblemente enlazadas requieren más memoria para almacenar el puntero adicional prev.
- **Complejidad de implementación:** Manejar correctamente los punteros next y prev para evitar errores como punteros colgantes o pérdida de nodos requiere atención.
- **Sobrecarga en operaciones simples:** En casos donde solo se necesita navegación unidireccional, las listas simples son más ligeras.

Conclusión

Los punteros next y prev son fundamentales para la estructura y operación de las listas enlazadas. Mientras que next establece la conexión hacia adelante, prev habilita el recorrido hacia atrás y facilita operaciones más eficientes en listas doblemente enlazadas. La correcta manipulación de estos punteros es clave para garantizar la integridad de la lista, evitar pérdidas de datos y mantener la eficiencia en operaciones de inserción, eliminación y búsqueda.

El estudio profundo de cómo funcionan y se manejan estos punteros es esencial para cualquier programador o ingeniero de software que desee comprender las bases de las estructuras de datos dinámicas y su aplicación en el desarrollo de software eficiente y robusto.