|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B



|  |  |
| --- | --- |
| Profesor(a): | M.I. Marco Antonio Martinez Quintana. |
| Asignatura: | Estructura de Datos y Algoritmos I |
| Grupo: | 17 |
| No de Práctica(s): | 08 |
| Integrante(s): | Pineda González Rodrigo |
| No. de Equipo de cómputo empleado: | 26 |
| No. de Lista o Brigada: | 30 |
| Semestre: | 2020-II |
| Fecha de entrega: | 24/03/2020 |
| Observaciones: |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

# Objetivo:

Revisar las definiciones, características, procedimientos y ejemplos de las estructuras lineales “Lista doblemente ligada” y “Lista doblemente ligada circular”, con la finalidad de comprender sus estructuras y poder implementarlas.

# Introducción:

Las listas son un tipo de estructura de datos lineal y dinámica. Las operaciones básicas dentro de una lista son BUSCAR, INSERTAR Y ELIMINAR.

* Para BUSCAR el método debe encontrar el primer elemento que coincida con la llave “K” dentro de la lista “L”, a través de una búsqueda lineal simple, regresando un apuntador a dicho elemento si éste se encuentra en la lista o nulo en caso contrario. Se puede buscar desde el HEAD o desde TAIL.
* Para INSERTAR, dado un nodo “X” que contenga una llave “K” previamente establecida, el método agrega el elemento “X” al inicio de la lista.
* Para ELIMINAR un elemento el método lo elimina de la lista “L” (si es que éste se encuentra en la estructura), aunque primero es necesario saber la ubicación del nodo a eliminar, por lo tanto, se debe realizar una búsqueda del elemento.

## Lista doblemente ligada

Simplificada a lista doble, está constituida por un conjunto de nodos lineales y unidos entre sí por dos referencias, una al sucesor (NEXT) y una al predecesor (PREV). La unidad básica es un nodo que contiene la información que se desea almacenar, así como dos referencias, una al siguiente elemento (NEXT) y otra al elemento anterior (PREV).

Dado un elemento x, NEXT[x] apunta al sucesor de x y PREV[x] apunta al predecesor de x. Sí PREV[x] es igual a NULL, el elemento x no tiene predecesor y, por ende, es el primer elemento (o HEAD) de la lista. Si NEXT[x] es igual NULL, el elemento x no tiene sucesor y, por ende, es el último elemento (o TAIL) de la lista. El atributo HEAD[L] apunta al primer elemento de la lista, si HEAD[L] es NULL entonces se puede afirmar que la lista está vacía.

Para diseñar un algoritmo implementando una lista simple, se consideran 2 casos para cada operación:

* Estructura vacía: no contiene elementos, por lo que la referencia a HEAD apunta a nulo, y no es posible BUSCAR elementos. Cuando se INSERTA un nuevo elemento la referencia a HEAD apunta al nodo insertado. No es posible ELIMINAR, pues no se contienen elementos.
* Estructura con elemento(s): la referencia HEAD apunta al primer elemento. Para BUSCAR es posible recorrer la lista a través de la referencia NEXT de cada nodo hasta llegar al que apunta a nulo, el cuál será el último elemento; también, si se posee una referencia al final de la lista TAIL, es posible recorrer la lisa a través de la referencia anterior PREV de cada nodo hasta llegar al que apunta a nulo, el cual será el primer elemento. Cuando se INSERTA un nuevo elemento la referencia del nuevo nodo (NEXT) apunta al mismo nodo al que apunta el inicio de la lista (HEAD), la referencia anterior (PREV) del nodo siguiente (NEXT) del inicio de la lista apunta al nuevo nodo, y HEAD también apunta al nuevo nodo. Para ELIMINAR, primero se debe buscar el elemento a eliminar, una vez encontrado, se deben mover las referencias de tal manera de que el antecesor del nodo a eliminar apunte al sucesor del mismo y el predecesor del nodo sucesor apunte al predecesor del nodo

## Lista doblemente ligada circular

Es una lista doblemente ligada modificada, donde la referencia siguiente (NEXT) del elemento que se encuentra al final de la lista (TAIL) en lugar de apuntar a nulo, apunta al primer elemento de la lista (HEAD).

* Estructura vacía: la referencia a HEAD apunta a nulo, por lo tanto, en una lista vacía no es posible BUSCAR elementos. Cuando se INSERTA un elemento la referencia al HEAD apunta al nodo insertado y la referencia a NEXT del nodo apunta a sí mismo. No es posible ELIMINAR pues no contiene elementos.
* Estructura con elemento(s): la referencia al inicio HEAD apunta al primer elemento de la lista y la referencia a NEXT del último elemento apunta al primer elemento. Es posible recorrer la lista a través de la referencia al sucesor NEXT de cada nodo (o su proceso contrario por TAIL a PREV). Hay que tener en cuenta el número de elementos de la lista, ya que el último elemento apunta al inicio de la estructura y, por tanto, se puede recorrer de manera infinita. Es posible BUSCAR una llave K. Cuando se INSERTA un elemento el sucesor del nuevo nodo NEXT apunta al mismo nodo al que apunta HEAD, la referencia al predecesor del nodo apunta a TAIL y ahora HEAD apunta al nuevo nodo. Así mismo, TAIL apunta al primer elemento (nuevo nodo). Para ELIMINAR se deben mover las referencias de la estructura de tal manera de que el antecesor del nodo a eliminar apunte al sucesor del mismo y viceversa.

# Desarrollo:

### Aplicaciones:

### Lista doblemente ligada

1. En servicios de correo electrónico, la forma de administrar los correos que se almacenan en la nube del usuario y su muestra por páginas se utiliza una lista doblemente ligada, donde cada correo tiene la opción de ir al siguiente o anterior, al igual que las páginas. por otro lado, si se está al inicio o al final, no hay más que mostrar, por lo que no se hará nada.
2. En la vida cotidiana, y de manera más abstracta, se puede ver como un libro, libreta o cuaderno, en el cual puedes “buscar” una página, por su portada o por su contraportada, puedes escribir sobre él y (dependiendo del material con el que escribas) borrar lo que escribes. Antes de su portada no tiene nada que le pertenezca al libro, de la misma forma que tras su contraportada.
3. Para programas o plataformas que proyecten el contenido de sus archivos en orden, en este caso enfocándonos con los que manejan proyección de diapositivas, como los servicios de Office (PowerPoint), Prezi, entre otros. La primera diapositiva no tiene un antecesor, pero tras ella puedes avanzar o retroceder a la diapositiva correspondiente. En la última diapositiva, al no haber sucesor, acaba la presentación.

### Lista doblemente ligada circular

1. En servicios de video bajo demanda en los cuales se puedan crear listas de reproducción que se puedan repetir hasta que el usuario lo desee, en el video que se reproduce hay una referencia para el siguiente video y el anterior. Si se está en el primer video, la referencia al anterior es el último video de la lista.
2. Se puede implementar en sistemas de selección en los menús, por ejemplo, de canales de televisión, del contenido de servicios de streaming, de un catálogo de libros, entre otros, para permitir que el usuario, tras recorrer todo el catálogo, no tenga que regresar manualmente hasta el inicio y en cambio empiece desde el inicio al moverse tras el último elemento. Aprovechando la estructura se pueden además navegar hacia “adelante y atrás” respecto al elemento en el que el usuario esté posicionado, para así no tenga que darse “una vuelta completa” a todos los elementos para regresar a uno específico.
3. Se puede apreciar en algunos videojuegos para mostrar diferentes elementos al jugador, entre ellos objetos, piezas, cartas, entre otros. Por ejemplo, en el juego “Solitario” incluído desde hace muchas versiones de Windows, cuando el usuario selecciona el mazo de cartas, estas se muestran de una manera definida (a pesar de ser aleatoria en cada partida, tras iniciar se mantienen estáticas y no cambian el resto de la partida) con las cuales el usuario puede jugar. Se puede seguir sorteando la siguiente carta o deshacer la jugada para ver la carta anterior, y tras “acabar” de barajar el mazo, su sucesor es con el que empezó el mazo.

# Conclusión:

Las estructuras de datos lineales resultan de gran utilidad para optimizar el uso de recursos y la eficiencia de los programas, y surgen de la abstracción del concepto que podemos ver en la vida cotidiana desde la antigüedad. En este caso, se vió una mejora de la previamente vista estructura lista, que permiten almacenar información en posiciones de memoria que puede ser operada no necesariamente de manera contigua, dándoles un rango más amplio de aplicación que otras estructuras de datos, como las pilas y las colas las cuales se ven limitadas a que las operaciones que se pueden ejecutar en ellas forzosamente tiene que ser un elemento sucesivo o anterior ya sea al último ingresado o al primero, dependiendo el caso. En estas listas también hay un elemento sucesivo y anterior para los nodos, sin embargo estos sirven para navegar en la búsqueda de los elementos que el usuario requiera. Retomando lo visto con las estructuras pasadas, cabe mencionar que las estructuras de datos no están definidas de manera puntual, como lo pueden ser las estructuras repetitivas o las estructuras de selección, sino que la programación se deja libre al programador para hacerlo tan complejo como le sea posible.

Además, las mejoras que brindan ambos tipos de listas son una gran implementación para resolver más problemas u optimizar los hechos con alguna estructura de menos nivel,pues su manera de operar con los métodos de buscar e insertar son más flexibles, y las limitaciones de memoria al ser dinámico únicamente están dadas por el equipo en el que se ejecute, pues con cantidades muy grandes de elementos podría ser que se desborde la memoria de equipos con recursos limitados.

# Referencias:

* Brian W. Kernighan, Dennis M. Ritchie. (1991). “*El lenguaje de programación C*”. Segunda edición, USA, Pearson Educación.
* Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, McGraw-Hill.