|  |  |  |
| --- | --- | --- |
|  | **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): | 04 |
| Integrante(s): | Pineda González Rodrigo |
| No. de Equipo de cómputo empleado: | 26 |
| No. de Lista o Brigada: | 30 |
| Semestre: | 2020-II |
| Fecha de entrega: | 17/03/2020 |
| Observaciones: |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_

# Objetivo:

Revisar las definiciones, características, procedimientos y ejemplos de las estructuras lineales Lista simple y Lista circular, con la finalidad de comprender su estructura 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.
* 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 simple

También conocida como lista ligada, está constituida por un conjunto de nodos alineados de manera lineal y unidos entre sí por una referencia. A diferencia de un arreglo, el orden está determinado por una referencia y no por un índice, y el tamaño no es fijo. La unidad básica es un elemento o nodo. Cada elemento de la lista es un objeto que contiene la información que se desea almacenar, así como una referencia (NEXT) al siguiente elemento (SUCESOR). 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. Cuando se INSERTA un nuevo elemento la referencia del nuevo nodo (NEXT) apunta al mismo nodo al que apunta HEAD y HEAD 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.

## Lista circular

Es una lista ligada simple modificada, donde el apuntador del elemento que se encuentra al final de la lista (TAIL) apunta al primer elemento de la lista (HEAD). Para diseñar un algoritmo implementando una lista circular,, se consideran 2 casos para cada opción:

* 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 a 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 NEXT de cada nodo; 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. Dentro de una lista circular con elementos es posible BUSCAR una llave K. Cuando se INSERTA un elemento la referencia del nuevo nodo (NEXT) apunta al mismo nodo al que apunta el inicio de la lista (HEAD) y ahora HEAD apunta al nuevo nodo. Así mismo, el último nodo de la estructura (TAIL) apunta al primer elemento. 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.

# Desarrollo:

### Aplicaciones:

### Lista ligada

1. En diversas redes sociales, la manera de organizar las publicaciones se hace siguiendo esta estructura, pues siempre se puede agregar algo nuevo que se mostrará al inicio, y al terminar no habrá un sucesor.
2. En la vida cotidiana, esta suele ser una estructura muy utilizada: desde las compras, las tareas o deberes del día, deudas saldadas, entre otras más, se organizan en forma de lista. Si algo ya se cumplió, se puede buscar a simple vista y se puede tachar o borrar ese elemento, o bien, escribir un dato nuevo en la lista, la cual puede ser tan amplia como se requiera.
3. Si se quisiera realizar un sistema de registro de entrada y salida de pacientes de algún hospital, se puede implementar una lista, ya que a las mismas siempre se pueden agregar elementos nuevos y después de un tiempo (ya sea al final de la jornada laboral, el día, etc.), la lista tiene un final, y no seguirán apareciendo más nombres en ella, aunque claro está que en un hospital no se sabe si habrá alguna emergencia y se tenga que agregar algún nombre más a la lista. Del mismo modo, no todos los pacientes se recuperarán en el orden en el que entraron, por lo que la flexibilidad de las listas permite eliminar a un paciente buscándolo en la lista.

### Lista circular

1. Para realizar una lista de reproducción de diversos tipos de contenidos multimedia, y al seleccionar la función de repetirse automáticamente se genera una lista circular, ya que el sucesor del último elemento es el primero 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 de la lista se pueden agregar los elementos de acuerdo a diversos criterios, y debido a que pueden agregarse n elementos nuevos, es mejor para así insertar o eliminar fácilmente el contenido de la lista.
3. En los videojuegos que cuentan con un sistema de juego multijugador masivo en línea, al conectarse al servidor se les asigna dentro de “salas”, algunas de las cuales necesitan albergar una cantidad dinámica de usuarios, por lo que el listado de las personas en línea se puede hacer con una estructura de pila circular, donde si un usuario sale de la sala, facilmente se puede retirar de la lista, pero se puede navegar fácilmente entre las personas restantes en la sala.

# 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 la estructura lista, que permiten almacenar información en posiciones de memoria que no sean contiguas, 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.

Además, las mejoras que brindan ambos tipos de listas son una gran implementación para resolver más problemas y hacerlos programables,pues su manera de operar con los métodos de buscar e insertar son más flexibles, o las limitaciones de memoria al ser dinámico. Claro está que se tiene que cuidar el equipo en el cual se utilice, pues con cantidades muy grandes de elementos podría ser que se desborde la memoria.

# 

# 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.