Unidad III: Estructuras lineales

Listas

En Ciencias de la Computación, una **lista enlazada** es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.

Una lista enlazada es un tipo de dato autorreferenciado porque contienen un puntero o enlace (en inglés *link*, del mismo significado) a otro dato del mismo tipo. Las listas enlazadas permiten inserciones y eliminación de nodos en cualquier punto de la lista en tiempo constante (suponiendo que dicho punto está previamente identificado o localizado), pero no permiten un acceso aleatorio. Existen diferentes tipos de listas enlazadas: listas enlazadas simples, listas doblemente enlazadas, listas enlazadas circulares y listas enlazadas doblemente circulares.

Las listas enlazadas pueden ser implementadas en muchos lenguajes. Lenguajes tales como Lisp y Scheme tiene estructuras de datos ya construidas, junto con operaciones para acceder a las listas enlazadas. Lenguajes imperativos u orientados a objetos tales como C o C++ y Java, respectivamente, disponen de referencias para crear listas enlazadas.

Operaciones básicas con listas

Las operaciones básicas son:

- Alta, adicionar un nuevo valor a la estructura.

- Baja, borrar un valor de la estructura.

- Búsqueda, encontrar un determinado valor en la estructura para realizar una operación con este valor, en forma secuencial o binario (siempre y cuando los datos estén ordenados).

Otras operaciones que se pueden realizar son:

- Ordenamiento, de los elementos pertenecientes a la estructura.
- Apareo, dadas dos estructuras originar una nueva ordenada y que contenga a las apareadas.

Cada estructura ofrece ventajas y desventajas en relación a la simplicidad y eficiencia para la realización de cada operación. De esta forma, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.

Tipos de listas

Lista enlazada

- ***Listas doblemente enlazadas***
- **Listas enlazadas circulares**
- ***Listas enlazadas doblemente circulares***

En una lista enlazada doblemente circular, cada nodo tiene dos enlaces, similares a los de la lista doblemente enlazada, excepto que el enlace anterior del primer nodo apunta al último y el enlace siguiente del último nodo, apunta al primero. Como en una lista doblemente enlazada, las inserciones y eliminaciones pueden ser hechas desde cualquier punto con acceso a algún nodo cercano. Aunque estructuralmente una lista circular doblemente enlazada no tiene ni principio ni fin, un puntero de acceso externo puede establecer el nodo apuntado que está en la

cabeza o al nodo cola, y así mantener el orden tan bien como en una lista doblemente enlazada.

Listas simplemente enlazadas

La lista enlazada básica es la **lista enlazada simple** la cual tiene un enlace por nodo. Este enlace apunta al siguiente nodo (o indica que tiene la dirección en memoria del siguiente nodo) en la lista, o al valorNULL o a la lista vacía, si es el último nodo.

Listas doblemente enlazadas

Un tipo de lista enlazada más sofisticado es la lista doblemente enlazada o lista enlazadas de dos vías. Cada nodo tiene dos enlaces: uno apunta al nodo anterior, o apunta al valor NULL si es el primer nodo; y otro que apunta al nodo siguiente, o apunta al valor NULL si es el último nodo.

En algún lenguaje de muy bajo nivel, XOR-Linking ofrece una vía para implementar listas doblemente enlazadas, usando una sola palabra para ambos enlaces, aunque esta técnica no se suele utilizar.

- **Pilas**

Una pila, es una estructura de datos en la que el último elemento en entrar es el primero en salir, por lo que también se denominan estructuras LIFO (*Last In, First Out*) o también estructuras lineales con una política UEPS (Ultimo en entrar, primero en salir).

En esta estructura sólo se tiene acceso a la cabeza o cima de la pila, también solo se pueden insertar elementos en la pila cuando esta tiene espacio y solo se pueden extraer elementos de la pila cuando tenga valores.

Operaciones asociadas con la pila
Crear la pila
Ver si la pila esta vacía
Insertar elementos en la pila
Eliminar un elemento de la pila
Vaciar la pila

- **Representación en memoria estática y dinámica**

Memoria estática.-

Las técnicas de asignación de memoria estática son sencillas.

La asignación de memoria puede hacerse en tiempo de compilación y los objetos están vigentes desde que comienza la ejecución del programa hasta que termina.

En los lenguajes que permiten la existencia de subprogramas, y siempre que todos los objetos de estos subprogramas puedan almacenarse estáticamente se aloja en la memoria estática un registro de activación correspondiente a cada uno de los subprogramas.

Estos registros de activación contendrán las variables locales, parámetros formales y valor devuelto por la función.

Consideraciones

ü Error en tiempo de ejecución de índice fuera del rango.

ü Se debe conocer con anticipación el tamaño de la estructura.

ü Se guardan en memorias adyacentes.

ü Vectores, matrices, cubos, registros, archivos.

Ventajas

· La velocidad de acceso es alta.

· Para retener los datos solo necesita estar energizada.

· Lógica simple.

Son más fáciles de diseñar.

Desventajas:

ü No se puede modificar el tamaño de la estructura en tiempo de ejecucion.

ü No es óptimo con grandes cantidades de datos.

ü Desperdicio de memoria cuando no se utiliza en su totalidad del tamaño v[100] .

ü Menor capacidad, debido a que cada celda de almacenamiento requiere más transistores.

ü Mayor costo por *bit.*

ü Mayor consumo de Potencia

MEMORIA DINÁMICA

La memoria dinámica es un espacio de almacenamiento que se solicita en tiempo de ejecución. De esa manera, a medida que el proceso va necesitando espacio para más líneas, va solicitando más memoria al sistema operativo para guardarlas. El medio para manejar la memoria que otorga el sistema operativo, es el puntero, puesto que no podemos saber en tiempo de compilación dónde nos dará huecos el sistema operativo (en la memoria de nuestro PC).

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va

dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definirla cantidad de memoria a utilizar en el momento de compilar el programa.

Cuando se crea un programa en el que es necesario manejar memoria dinámica el sistema operativo divide el programa en cuatro partes que son: texto, datos (estáticos), pila y una zona libre o heap. En el momento de la ejecución habrá tanto partes libres como partes asignadas al proceso por lo cual si no se liberan las partes utilizadas de la memoria y que han quedado inservibles es posible que se “agote” esta parte y por lo tanto la fuente de la memoria dinámica. También la pila cambia su tamaño dinámicamente, pero esto no depende del programador sino del sistema operativo.

VENTAJAS:

ü Es posible disponer de un espacio de memoria arbitrario que dependa de información dinámica (disponible sólo en ejecución): Toda esa memoria que maneja es implementada por el programador cuando fuese necesario.

ü Otra ventaja de la memoria dinámica es que se puede ir incrementando durante la ejecución del programa. Esto permite, por ejemplo, trabajar con arreglos dinámicos.

ü Es memoria que se reserva en tiempo de ejecución. Su tamaño puede variar durante la ejecución del programa y puede ser liberado mediante la función free.

DESVENTAJAS:

ü Es difícil de implementar en el desarrollo de un programa o aplicación.

ü Es difícil implementar estructuras de datos complejas como son los tipos recursivos (árboles, grafos, etc.).Por ello necesitamos una forma para solicitar y

liberar memoria para nuevas variables que puedan ser necesarias durante la ejecución de nuestros programas: Heap.

ü Una desventaja de la memoria dinámica es que es más difícil de manejar.

ü La memoria dinámica puede afectar el rendimiento. Puesto que con la memoria estática el tamaño de las variables se conoce en tiempo de compilación, esta información está incluida en el código objeto generado. Cuando se reserva memoria de manera dinámica,

ü Se tienen que llevar a cabo varias tareas, como buscar un bloque de memoria libre y almacenar la posición y tamaño de la memoria asignada, de manera que pueda ser liberada más adelante. Todo esto representa una carga adicional, aunque esto depende de la implementación y hay técnicas para reducir su impacto.

- **Operaciones básicas con pilas**

Una pila cuenta con 2 operaciones imprescindibles: apilar y desapilar, a las que en las implementaciones modernas de las pilas se suelen añadir más de uso habitual.

- **Crear:** se crea la pila vacía. (constructor)
- **Tamaño:** regresa el número de elementos de la pila. (size)
- **Apilar:** se añade un elemento a la pila.(push)
- **Desapilar:** se elimina el elemento frontal de la pila.(pop)
- **Cima:** devuelve el elemento que esta en la cima de la pila. (top o peek)
- **Vacía:** devuelve cierto si la pila está vacía o falso en caso contrario (empty).

- **Notación infija y postfija**

PreFija:

La Expresión o Notación PreFija nos indica que el operador va antes de los operandos sus características principales son:

-Los operandos conservan el mismo orden que la notación infija equivalente.
-No requiere de paréntesis para indicar el orden de precedencia de operadores ya que el es una operación.
-Se evalúa de izquierda a derecha hasta que encontrémosle primer operador seguido inmediatamente de un par de operandos.
-Se evalúa la expresión binaria y el resultado se cambia como un nuevo operando. Se repite este hasta que nos quede un solo resultado.

(A+B)*C

Notación prefija: El orden es operador, primer operando, segundo operando

InFija:

La Expresión o Notación InFija es la forma mas común que utilizamos para escribir expresiones matemáticas, estas notaciones se refiere a que el operador esta entre los operandos. La notación infija puede estar completamente parentizada o puede basarse en un esquema de precedencia de operadores así como el uso de paréntesis para invalidar los arreglos al expresar el orden de evaluación de una expresión:

3*4=12
3*4+2=14
3*(4+2)=18

Notación infija: La notación habitual. El orden es primer operando, operador, segundo operando.

PosFija:

Como su nombre lo indica se refiere a que el operador ocupa la posición después de los operandos sus características principales son:
-El orden de los operandos se conserva igual que la expresión infija equivalente no utiliza paréntesis ya que no es una operación ambigua.
-La operación posfija no es exactamente lo inverso a la operación prefija equivalente:
(A+B)*C AB+C*

- **Recursividad con ayuda de pilas**
- **Colas**
- La particularidad de una estructura de datos de cola es el hecho de que sólo podemos acceder al primer y al último elemento de la estructura. Así mismo, los elementos sólo se pueden eliminar por el principio y sólo se pueden añadir por el final de la cola.

15	20	9	.....	18	19

1.Ejemplo de Cola

15	20	9	.....	18	19

2.Vamos a Insertar el 13 en la Cola.

15	20	9	.....	18	19	13

3.Sacamos el frente de la Cola (15)

20	9	.....	18	19	13

Ejemplos de colas en la vida real serían: personas comprando en un supermercado, esperando para entrar a ver un partido de béisbol, esperando en el cine para ver una película, una pequeña peluquería, etc. La idea esencial es que son todos líneas de espera.

- **Representación en memoria estática y dinámica.**
- **Operaciones básicas con colas**

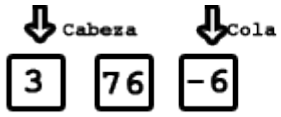
Pues en las colas como en toda estructura de datos las operaciones principales son insertar y eliminar, aunque en varias implementaciones de colas puedan recibir nombres diferentes.

Insertar

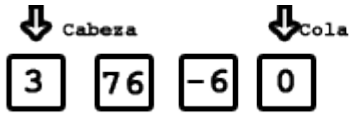
La inserción en las colas se realiza por la cola de las mismas, es decir, se inserta al final de la estructura.

Para llevar a cabo esta operación únicamente hay que reestructurar un par de punteros, el último nodo debe pasar a apuntar al nuevo nodo (que pasará a ser el último) y el nuevo nodo pasa a ser la nueva cola de la cola.

Vamos a verlo gráficamente sobre la siguiente cola:



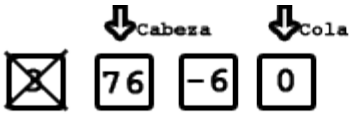
Si a esta cola le añadimos el elemento 0, la cola resultante sería:



Borrar

El borrado es una operación muy simple en las colas. Esta operación supone extraer la cabeza de la cola, ya que es el elemento que más tiempo lleva en la estructura. Para llevar a cabo esta operación únicamente hay que extraer el elemento situado en la cabeza de la cola y avanzar el puntero cabeza una posición, para que de esta forma la nueva cabeza sea el segundo elemento que más tiempo lleva en la cola.

Si realizamos la operación eliminar sobre la colas de 4 elementos del último gráfico el resultado sería el siguiente:



- **Tipos de colas: Cola simple, Cola circular y Colas dobles**
- **Aplicaciones: Colas de prioridad**