InstitutoSuperiordeFormaciónTécnicaNº151

Carrera:AnalistaenSistemas

3Año.Algoritmos y Estructuras de Datos III.

TrabajoPrácticoNº4	Unidad4
Modalidad: Semi -Presencial	**EstratégicaDidáctica:**Trabajoindividual.
MetodologíadeDesarrollo: Det. docente	**MetodologíadeCorrección:**ViaClassroom.
CarácterdeTrabajo: Obligatorio – ConNota	**FechaEntrega:**AconfirmarporelDocente.

PILAS Y COLAS.

MarcoTeórico:

Responder el siguiente cuestionario en función de la bibliografía Obligatoria.

1. Describir el concepto de Pila y sus Operaciones Básicas

Una pila es un tipo especial de lista abierta en la que sólo se pueden insertar y eliminar nodos en uno de los extremos de la lista. Estas operaciones se conocen como "push" y "pop", respectivamente "empujar" y "tirar". Además, las escrituras de datos siempre son inserciones de nodos, y las lecturas siempre eliminan el nodo leído.Estas características implican un comportamiento de lista LIFO (Last In First Out), el último en entrar es el primero en salir.

El símil del que deriva el nombre de la estructura es una pila de platos. Sólo es posible añadir platos en la parte superior de la pila, y sólo pueden tomarse del mismo extremo.

El nodo típico para construir pilas es el mismo que vimos en el capítulo anterior para la construcción de listas:

```
struct nodo {
    int dato;
    struct nodo *siguiente;
};
```

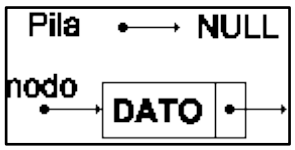
Las pilas tienen un conjunto de operaciones muy limitado, sólo permiten las operaciones de "push" y "pop":

- Push: Añadir un elemento al final de la pila.
- Pop: Leer y eliminar un elemento del final de la pila.

Push, insertar elemento

Las operaciones con pilas son muy simples, no hay casos especiales, salvo que la pila esté vacía.

Push en una pila vacía



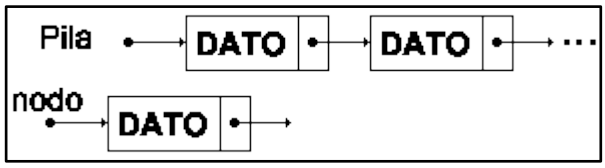
Push en vacía

Partiremos de que ya tenemos el nodo a insertar y, por supuesto un puntero que apunte a él, además el puntero a la pila valdrá NULL:

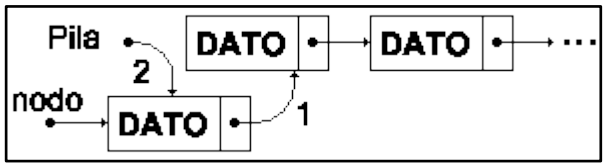
El proceso es muy simple, bastará con que:

1. nodo->siguiente apunte a NULL.
2. Pilaa apunte a nodo.

Push en una pila no vacía



Push en pila no vacía



Resultado

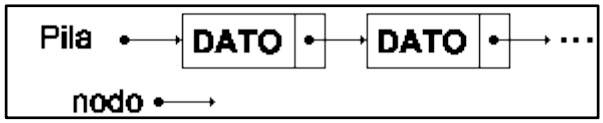
Podemos considerar el caso anterior como un caso particular de éste, la única diferencia es que podemos y debemos trabajar con una pila vacía como con una pila normal.

De nuevo partiremos de un nodo a insertar, con un puntero que apunte a él, y de una pila, en este caso no vacía:

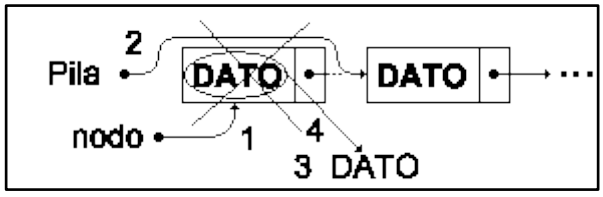
El proceso sigue siendo muy sencillo:

1. Hacemos que nodo->siguiente apunte a Pila.
2. Hacemos que Pila apunte a nodo.

Pop, leer y eliminar un elemento



Pop



Resultado

Ahora sólo existe un caso posible, ya que sólo podemos leer desde un extremo de la pila.

Partiremos de una pila con uno o más nodos, y usaremos un puntero auxiliar, nodo:

1. Hacemos que nodo apunte al primer elemento de la pila, es decir a Pila.
2. Asignamos a Pila la dirección del segundo nodo de la pila: Pila->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación pop equivale a leer y borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.

Si la pila sólo tiene un nodo, el proceso sigue siendo válido, ya que el valor de Pila->siguiente es NULL, y después de eliminar el último nodo la pila quedará vacía, y el valor de Pila será NULL.

2. Explicar el concepto de "StakOverFlow" y como evitarelo?

El desbordamiento de pila es un error de software que ocurre cuando un programa intenta acceder a más memoria que el tamaño de pila disponible, lo que provoca el bloqueo del programa. Stack es la estructura de datos Last in First Out (LIFO). Se utiliza para almacenar las variables locales, los parámetros/argumentos pasados en la función y sus direcciones de retorno. Cuando la función se ejecuta por completo, todas sus variables locales y otros datos se eliminan de la pila y se libera la memoria. Sin embargo, como la memoria de la pila es limitada en la computadora, si el programa accede a más memoria que la disponible, surge la condición de desbordamiento de la pila.Funciona de manera similar a los platos llanos colocados uno encima del otro. Entonces, cuando se requieren estas placas, la última placa se usa primero y el primer conjunto de placas se usa al final. Lo mismo sucede en el caso de la memoria de pila, ya que la pila es la región de la memoria del proceso y está limitada en la computadora.

3. Buscar describir las diferencias de Implementar Pilas en Clase so viasTL?

Las clases para pilas son versiones simplificadas de las mismas clases que usamos para listas.Para empezar, necesitaremos dos clases, una para nodo y otra para pila. Además la clase para nodo debe ser amiga de la clase pila, ya que ésta debe acceder a los miembros privados de nodo.

El contenedor de pila en STL es un tipo de adaptadores de contenedor. Se utiliza para replicar una estructura de datos de pila en C ++. El contenedor de pila es un conjunto de elementos en el que los elementos se insertan en un extremo y también se eliminan en el mismo extremo.Este punto común de adición y eliminación se conoce como 'Top of the stack'.

4. Que es una Cola y cuales son las operaciones Basicas?

La cola es un tipo de estructura de datos que opera en forma de primero en entrar, primero en salir (FIFO), lo que significa que el elemento se ingresará desde atrás y se eliminará desde el frente. Como si tuviéramos un sistema general de colas en el mundo práctico. La cola es un adaptador de contenedor que contiene datos del mismo tipo. El adaptador de contenedor no contiene iteradores, por lo que no podemos manipular nuestros datos. La cola en c ++ solo nos proporciona dos métodos para insertar elementos y eliminar elementos, es decir, push() y pop().

```
/*            Estructura de los nodos de la cola
--------------------------------------------------------------------------*/
struct nodo {
  int dato;
  struct nodo *siguiente;
};
/*                Estructura de la cola
--------------------------------------------------------------------------*/
struct cola
{
   nodo *delante;
   nodo *atras  ;
};
```

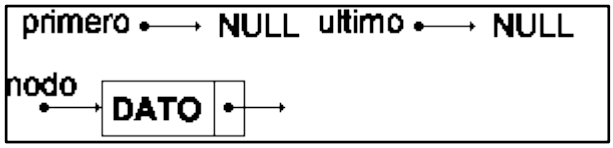
De nuevo nos encontramos ante una estructura con muy pocas operaciones disponibles. Las colas sólo permiten añadir y leer elementos:

- Añadir: Inserta un elemento al final de la cola.
- Leer: Lee y elimina un elemento del principio de la cola.

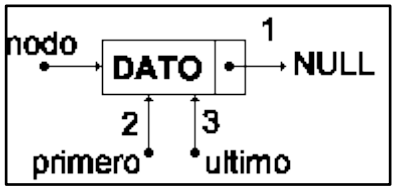
Añadir un elemento

Las operaciones con colas son muy sencillas, prácticamente no hay casos especiales, salvo que la cola esté vacía.

Añadir elemento en una cola vacía



Cola vacía



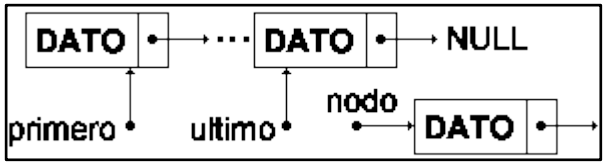
Elemento encolado

Partiremos de que ya tenemos el nodo a insertar y, por supuesto un puntero que apunte a él, además los punteros que definen la cola, primero y ultimo que valdrán NULL:

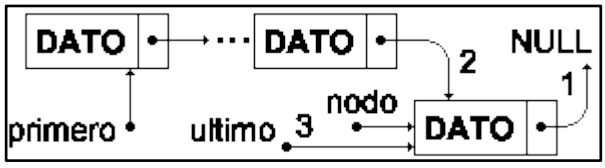
El proceso es muy simple, bastará con que:

1. Hacer que nodo->siguiente apunte a NULL.
2. Que el puntero primero apunte a nodo.
3. Y que el puntero último también apunte a nodo.

Añadir elemento en una cola no vacía



Cola no vacía



Elemento encolado

De nuevo partiremos de un nodo a insertar, con un puntero que apunte a él, y de una cola, en este caso, al no estar vacía, los punteros primero y ultimo no serán nulos:

El proceso sigue siendo muy sencillo:

1. Hacemos que nodo->siguiente apunte a NULL.
2. Después que ultimo->siguiente apunte a nodo.

3. Y actualizamos ultimo, haciendo que apunte a nodo.

Añadir elemento en una cola, caso general

Para generalizar el caso anterior, sólo necesitamos añadir una operación:

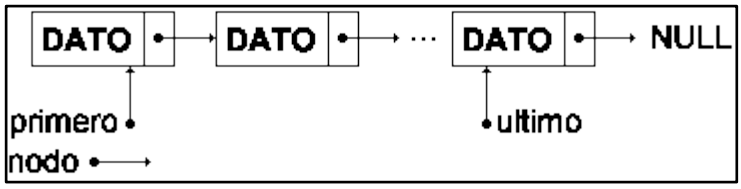
1. Hacemos que nodo->siguiente apunte a NULL.
2. Si ultimo no es NULL, hacemos que ultimo->siguiente apunte a nodo.
3. Y actualizamos ultimo, haciendo que apunte a nodo.
4. Si primero es NULL, significa que la cola estaba vacía, así que haremos que primero apunte también a nodo.

Leer un elemento de una cola, implica eliminarlo

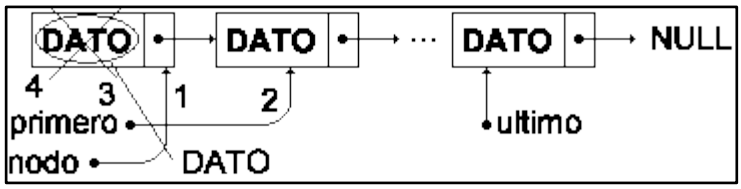
Ahora también existen dos casos, que la cola tenga un solo elemento o que tenga más de uno.

Leer un elemento en una cola con más de un elemento

Usaremos un puntero a un nodo auxiliar:



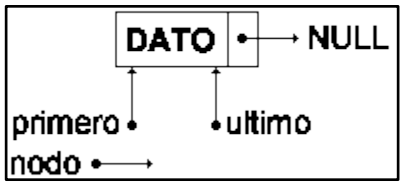
Cola con más de un elemento



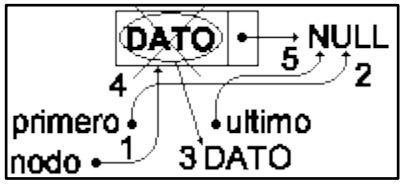
Elemento desencolado

1. Hacemos que nodo apunte al primer elemento de la cola, es decir a primero.
2. Asignamos a primero la dirección del segundo nodo de la pila: primero->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.

Leer un elemento en una cola con un solo elemento



Cola con un elemento



Elemento desencolado

También necesitamos un puntero a un nodo auxiliar:

1. Hacemos que nodo apunte al primer elemento de la pila, es decir a primero.
2. Asignamos NULL a primero, que es la dirección del segundo nodo teórico de la cola: primero->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
5. Hacemos que ultimo apunte a NULL, ya que la lectura ha dejado la cola vacía.

Leer un elemento en una cola caso general

1. Hacemos que nodo apunte al primer elemento de la pila, es decir a primero.
2. Asignamos a primero la dirección del segundo nodo de la pila: primero->siguiente.
3. Guardamos el contenido del nodo para devolverlo como retorno, recuerda que la operación de lectura en colas implican también borrar.
4. Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
5. Si primero es NULL, hacemos que ultimo también apunte a NULL, ya que la lectura ha dejado la cola vacía.

Las siguientes operaciones son necesarias para administrar correctamente una cola:

■ clear()::Borrar la cola.

■ isEmpty():compruebe si la cola está vacía.

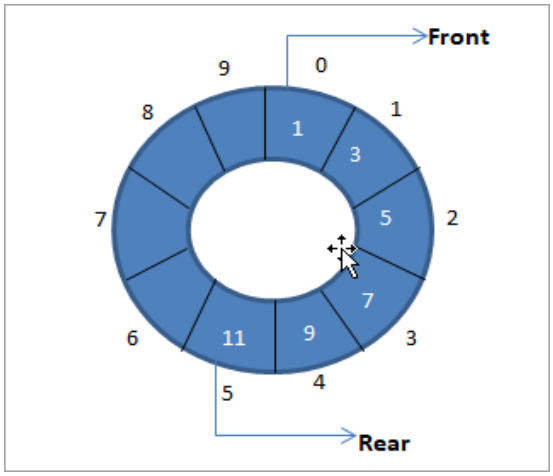
■ enqueue(el):permite colocar el elemento el al final de la cola.

■ dequeue():permite tomar el primer elemento de la cola.

■ firstEl(): permite devolver el primer elemento de la cola sin quitarlo

5. Describir: Cola circular, Multicola, Deque y cola prioritaria.

Cola circular: Es una estructura de datos lineal que se utiliza para almacenar elementos de datos. Realiza operaciones siguiendo el enfoque FIFO (primero en entrar, primero en salir) y la última posición en la cola se conecta de nuevo a la primera posición para formar un círculo.



Multicola:Si se requiere implementar más número de colas,se requiere una estructura de datos eficiente para manejar múltiples colas. Es posible utilizar todos los espacios disponibles en una sola array. Cuando más de dos colas, digamos n, se representan secuencialmente, podemos dividir la memoria disponible A[size] en n segmentos y asignar estos segmentos a n colas, una a cada una. Para cada cola i usaremos Front[i] y Rear[i]. Usaremos la

condición Front[i] = Rear[i] si y sólo si la i ésima cola está vacía, y la condición Rear[i] = Front[i] si y sólo si la i ésima cola está llena. Si queremos cinco colas, entonces podemos dividir la array A[100] en partes iguales de 20 e inicializar delante y detrás para cada cola, es decir, Front[0] = Rear[0] = 0 y Front[1] = Rear[1] = 20, y así sucesivamente para otras colas.

Deque: La palabra deque es una forma corta de cola de doble extremo. Se pronuncia como 'deck'. Deque define una estructura de datos en la que los elementos se pueden agregar o eliminar en el extremo frontal o en el extremo posterior, pero no se pueden realizar cambios en ninguna otra parte de la lista. Por lo tanto, deque es una generalización tanto de una pila como de una cola. Es compatible con capacidades similares a pilas y colas. Es un contenedor secuencial que está optimizado para un acceso rápido basado en índices y una inserción eficiente en cualquiera de sus extremos. Deque se puede implementar como un deque continuo o como un deque vinculado.

Cola prioritaria: Es similar a la cola en ciertos aspectos y, sin embargo, se diferencia de la cola ordinaria en los siguientes puntos:

- Cada elemento de la cola de prioridad está asociado con una prioridad.
- El elemento con la prioridad más alta es el primer elemento que se elimina de la cola.
- Si más de un elemento tiene la misma prioridad, se considera su orden en la cola.

Estado inicial - Cola de prioridad (PQ) - {} => vacío

Operation	Priority Queue	Return value
Insert(G)	G	
Insert(O)	G O	
Insert(M)	G O M	
deleteHighestPriority()	G M	O
Insert(A)	G M A	
deleteHighestPriority()	G A	M

En la ilustración anterior, vemos que la operación de inserción es similar a una cola normal. Pero cuando llamamos a 'deleteHighestPriority' para la cola de prioridad, el elemento con la prioridad más alta se elimina primero.

6. Para Pilas y Colas mas subtipos, Dar ejemplos de Utilización de cada uno.

Pilas: Una pila es una estructura de datos en la que los datos similar a una lista en la que los datos se introducen y se sacan por el mismo lugar. Un ejemplo de la vida real puede ser una caja de CD tipo torre, en la que los CD solamente se pueden introducir o sacar por arriba.

colas: Las colas se utilizan en sistemas informáticos, transportes y operaciones de investigación (entre otros), donde los objetos, personas o eventos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento

Marco Práctico:

1. Un Negocio necesita gestionar la atención a sus cliente, el mismo recibe los clientes en un Box de Atencion que registra los clientes y los carga para su atención según su llegada.
   Nos piden desarrollar un sistema de “Turnos” que se muestren en una pantalla.
   La aplicación deberá registrar los clientes, mostrarlos en orden de llegada, llamarlos según ese orden por pantalla con una opción “próximo cliente – Box nro”

2. Ídem 1 pero Implementando Listas en STL

3. Una empresa necesita implementar un Historial de llamadas a sus cliente, la aplicación deberá ir cargando los números telefónicos en un historial y el usuario podrá ir recorriendo el histórico de llamadas (similar al botón Back del navegador).
   Implementar una app que realice esta función.

Es una pila el último en llamar es el primero en mostrarse.

Lic.OemigJoséLuis.