**UNIVERSIDAD NACIONAL DE HURLINGHAM**

****

Materia: Estructura de Datos

Docente : González, Sergio

Tema: Informe, TP N°3 Pilas / Colas

Alumnos: Mariano A. Salto, Santiago Smith

Año: 2020, Primer Cuatrimestre

**Introducción:**

En este informe explicaremos de forma breve y concisa las estructuras de datos cola (queue) y pila (stack) que son utilizadas en la implementación de los algoritmos. Incluimos una conclusión al final comentando como hicimos para resolver el trabajo y los problemas que se nos presentaron.

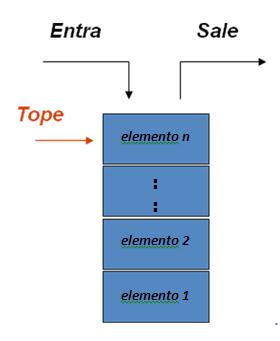
**Estructura de Datos: Pila (Stack)**

La pila es una estructura para almacenar datos y recuperarlos de forma ordenada, es de tipo **LIFO** (Last In, First Out ), esto quiere decir que el último elemento que entra en esta es el primer elemento en salir de la misma.

Sus funciones principales son push(), que añade un elemento que recibe por parámetro y lo apila a la parte superior, pop() que devuelve el último elemento de la pila y lo elimina de la misma, top() devuelve el elemento que está arriba de la pila sin eliminarlo. Además cuenta con las operaciones empty() y size(), la primera devuelve si la pila está vacía (o sea , no contiene ningún elemento) y la segunda devuelve el numero de elementos que contiene (cantidad).

En nuestro trabajo al implementar estas funciones le asignamos los nombres de:

* push = apilarElemento(elemento)
* pop = desapilarElemento()
* top = topePila()
* empty = estaVacia()
* size = tamanioPila()
* clonarPila(), que como dice el nombre clona una pila ya existente.

****

**Fig1. Stack (Pila)- Ejemplo gráfico del funcionamiento de la estructura donde el ultimo elemento en ser apilado es el primero en salir.**

**Estructura de Datos: Cola (Queue)**

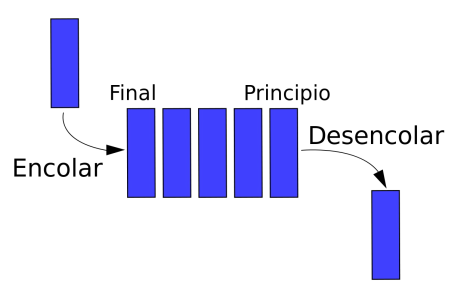
La cola es una estructura que almacena los datos “encolándolos” por un extremo y “desencolándolos” por el otro, es de tipo **FIFO** (First In First Out – primero en entrar, primero en salir). Se Podría pensar y representar como una cola de supermercado, es decir, el primer cliente que llega a la cola, es el primero en ser atendido, luego sigue el siguiente y así hasta que quede completamente vacía.

Cuenta con las operaciones queue(), que añade el elemento en la posición 0 (que a su vez es el final de la cola), y a medida que se van agregando elementos estos se van corriendo. deQueue() que devuelve el primer elemento que entró y lo elimina de la cola Top() o front() que devuelve el primer elemento de la cola, o sea el primero que entró. También cuenta con empty(), len() o size(), que devuelven si está vacía, y el tamaño de la misma respectivamente

En nuestra implementación también agregamos clonarCola(), que clona una cola ya creada y vaciarCola() que elimina todo los elementos que hay en ella.

También le asignamos diferentes nombres para poder hacer un seguimiento del código y sea legible y más práctico.

* queue = encolar(elemento)
* dequeue = desencolar()
* top = primerElementoFila()
* empty , size() = devuelve si la cola esta vacía.

****

**Fig2. Queue (Cola)- Ejemplo gráfico del funcionamiento de la estructura donde el primer elemento en entrar a la fila es el primero en salir.**

**Otras estructuras usadas:**

Para la implementación del problema, utilizamos otras estructuras como TDA’s (Fueros, Edificio Tribunales, etc) los cuales tienen su funcionamiento interno independiente. Para crear el “edificio de Tribunales” utilizamos un arreglo bidimensional (matriz) el cual fuimos recorriendo para obtener lo que se encuentra en cada una de las coordenadas, las mismas representar una oficina que contiene un juzgado con el nombre del juez y sus respectivas “filas” de expedientes a tratar, ya sea urgentes o normales. Toda esta implementación y explicación de los algoritmos se encuentra comentada en el código adjunto a este pequeño informe.

**Conclusión:**

La manera que aplicamos para ir desarrollando el código fue, dividir los problemas en otros más más pequeños, de esta manera, ir testeando y verificando cada uno, más aun, en las funciones que nosotros consideramos un poco más complejas. Si el problema que planteábamos no funcionaba, partíamos desde la base del mismo (con implementaciones más simples), hasta llegar a la solución completa, realizando lo que se solicita en el enunciado.

También al implementar las estructuras, les asignamos a las operaciones y funciones

nombres para mejor entendimiento y legibilidad. Somos conscientes que a la hora de implementar algunas de las mismas (si bien el código hace lo solicitado), la resolución no es “elegante” (como por ejemplo, en varias de las operaciones del edificio de Tribunales.

El código podría haber quedado mucho mejor sin repeticiones, más estilizado en los casos cuando los algoritmos son parecidos y hacen lo mismo (cosa que si hicimos en estructuras anteriores) pero se nos complicó y no supimos como implementarlo por ejemplo, al recorrer el edificio (matriz), la cual utilizamos los “For” varias veces ,

La función que debía ser recursiva, tampoco la pudimos implementar porque no encontramos la manera de que funcione, aunque igual fue realizada pero de forma iterativa. Nos queda pendiente al momento de entrega del TP la consulta sobre la resolución de dicha función (la cual nos surgió al final) y de esta manera poder entender mejor como encarar el problema para resolverlo recursivamente).

Cada estructura, algoritmo, e implementación se encuentran comentadas en el código. Y también los archivos “Main” donde íbamos probando lo que hacíamos.

Debemos reconocer que como experiencia personal nos encontramos con un desafío que realmente desde el principio creímos no poder resolver, pero con insistencia y practica logramos hacer la mayoría de los requerimientos. Hacemos hincapié que hay algoritmos y estructuras que podrían mejorarse, pero de la mayoría de las funciones se obtiene lo solicitado. Agradecemos al docente el tiempo invertido en aclarar dudas, donde el dictado de clases y demás cuestiones en la realidad que vivimos no es a lo que estamos acostumbrados.