Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 2.2 - Verificación de las funcionalidades de una estructura de datos lineal

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

24 de septiembre de 2023

Código fuente

- Adjunto en el archivo zip.

Introducción

En el código proporcionado, se presenta una implementación de dos Tipos de Datos Abstractos (ADT): una pila y una cola. Ambas son estructuras de datos lineales, pero difieren en cómo se accede a los elementos.

Una pila es como una pila de libros que has apilado uno encima del otro. El último libro que colocaste en la pila será el primero que tomarás cuando empieces a leer. Este es el principio LIFO (Last In, First Out). En nuestra implementación de pila, cada libro es un nodo que tiene un valor y una referencia al libro que está debajo de él en la pila. Y siempre trabajamos con el libro que está en la cima de la pila.

Por otro lado, una cola es como una fila de personas esperando para comprar entradas para un concierto. La primera persona que llegó a la fila será la primera en obtener su entrada. Este es el principio FIFO (First In, First Out). En nuestra implementación de cola, cada persona es un nodo con un valor y una referencia a la siguiente persona en la fila. Y siempre trabajamos con la persona que está al frente de la fila.

Casos de prueba

Create

Número de Caso de Prueba para las Filas	**Entrada**	**Salida Filas**	**Salida Pilas**
Caso 1	1	1	1

Caso 2	2	1 2	2 1
Caso 3	3	1 2 3	3 2 1
Caso 4	4	1 2 3 4	4 3 2 1
Caso 5	5	1 2 3 4 5	5 4 3 2 1

Read

Número de Caso de Prueba para las Filas	**Entrada**	**Salida Filas**	**Salida Pilas**
Caso 1	-	1	5

Update

Número de Caso de Prueba	**Entrada**	**Salida Filas**	**Salida Pilas**
Caso 1	6	6 2 3 4 5	6 4 3 2 1

Caso 2	7	7 2 3 4 5	7 4 3 2 1
Caso 3	8	8 2 3 4 5	8 4 3 2 1
Caso 4	9	9 7 8 9 5	9 7 8 9 5
Caso 5	10	9 7 8 9 10	9 7 8 9 10

Delete

Número de Caso de Prueba	**Entrada**	**Salida Filas**	**Salida Pilas**
Caso 1	-	2 3 4 5	4 3 2 1
Caso 2	-	3 4 5	3 2 1
Caso 3	-	4 5	2 1
Caso 4	-	5	1

Caso 5	-	-	-

Justificación casos de prueba

Los números 1, 2, 3, 4 y 5 se seleccionaron como datos de prueba para las funciones CRUD de las pilas y colas debido a su simplicidad y diversidad. Son simples y fáciles de entender, lo que facilita la verificación de los resultados de las pruebas. Aunque son simples, estos cinco números son suficientemente diversos para probar la funcionalidad básica. Por ejemplo, nos permiten verificar que los elementos se añaden y se eliminan en el orden correcto. Además, con sólo cinco elementos, estas pruebas se pueden ejecutar rápidamente, lo que permite realizar iteraciones más rápidas durante el desarrollo. Aunque estas pruebas son pequeñas, el enfoque utilizado para diseñarlas se puede escalar a conjuntos de datos más grandes si es necesario. En resumen, estos casos de prueba proporcionan una forma eficiente y efectiva de verificar que las operaciones CRUD de las pilas y colas funcionen como se espera.

Comprobación casos de prueba

```
Creando nodos en la Fila y la Pila:
Fila después de crear nodos: 1
Pila después de crear nodos: 1
Fila después de crear nodos: 1 2
Pila después de crear nodos: 2 1
Fila después de crear nodos: 1 2 3
Pila después de crear nodos: 3 2 1
Fila después de crear nodos: 1 2 3 4
Pila después de crear nodos: 4 3 2 1
Fila después de crear nodos: 1 2 3 4 5
Pila después de crear nodos: 5 4 3 2 1

Leyendo valores en la Fila y la Pila:
El primer valor en la cola es: 1
El valor superior de la pila es: 5


Actualizando valores en la Fila y la Pila:
Fila después de actualizar el nodo con valor 6: 6 2 3 4 5
Pila después de actualizar el nodo con valor 6: 6 4 3 2 1
Fila después de actualizar el nodo con valor 7: 7 2 3 4 5
Pila después de actualizar el nodo con valor 7: 7 4 3 2 1
Fila después de actualizar el nodo con valor 8: 8 2 3 4 5
Pila después de actualizar el nodo con valor 8: 8 4 3 2 1
Fila después de actualizar el nodo con valor 9: 9 2 3 4 5
Pila después de actualizar el nodo con valor 9: 9 4 3 2 1
Fila después de actualizar el nodo con valor 10: 10 2 3 4 5
Pila después de actualizar el nodo con valor 10: 10 4 3 2 1

Eliminando valores en la Fila y la Pila:
Fila después de eliminar el nodo con valor 6: 2 3 4 5
Pila después de eliminar el nodo con valor 6: 4 3 2 1
Fila después de eliminar el nodo con valor 7: 3 4 5
Pila después de eliminar el nodo con valor 7: 3 2 1
Fila después de eliminar el nodo con valor 8: 4 5
Pila después de eliminar el nodo con valor 8: 2 1
Fila después de eliminar el nodo con valor 9: 5
Pila después de eliminar el nodo con valor 9: 1
Fila después de eliminar el nodo con valor 10:
Pila después de eliminar el nodo con valor 10:
```

Para complementar lo anterior, es posible agregar que la calidad del software es un aspecto esencial en el desarrollo de cualquier sistema de software. En el caso de los Tipos de Datos Abstractos (ADT), como las pilas y las colas, hay varios factores que contribuyen a su calidad.

La indentación es uno de estos factores. Es como el lenguaje corporal del código: puede no decir nada por sí misma, pero puede hacer que el mensaje general sea mucho más claro. Una buena indentación puede hacer que un bloque de código sea

mucho más fácil de entender a primera vista, permitiendo a los desarrolladores entender rápidamente la estructura y el flujo del código. Además, la documentación es otro factor crucial. Es como un mapa para los desarrolladores: les muestra cómo funciona el código, cómo usarlo y qué esperar de él. Una buena documentación puede ser la diferencia entre un proyecto fácil de mantener y uno que es prácticamente inentendible para cualquier persona excepto su autor original.

La mantenibilidad es la medida de cuán fácil es para los desarrolladores hacer cambios en el código. Un código altamente mantenible es como un coche con un motor fácilmente accesible: cuando algo necesita ser cambiado o arreglado, se puede hacer de manera eficiente sin tener que desmontar todo el coche. Por último, está la facilidad de lectura y entendimiento. Un código fácil de leer es como un libro bien escrito: los desarrolladores pueden seguir su flujo y entender su propósito sin tener que luchar contra una sintaxis confusa o una estructura complicada.

En conclusión, sin duda es posible afirmar que estos cuatro conceptos son indispensables para medir la calidad del software en los ADT. Al prestar atención a estos aspectos durante el desarrollo del software, podemos asegurarnos de que nuestro código no sólo funciona correctamente, sino que también es robusto, eficiente y fácil de mantener en el futuro.