Tecnológico de Monterrey
Escuela de Ingeniería y Ciencias

Campus Sonora Norte

Act 2.1 - Implementación de un ADT de estructura de datos lineales

Curso:

Programación de estructuras de datos y algoritmos fundamentales

(TC1031)

Estudiante:

Daniel Alfredo Barreras Meraz A01254805

Docente:

Baldomero Olvera Villanueva

Fecha de entrega:

26 de septiembre de 2023

Código fuente

- Adjunto en el archivo zip.

Introducción

En el código proporcionado, se presenta una implementación de un Tipo de Dato Abstracto (ADT) en la forma de una lista enlazada. Esta lista enlazada es un ADT que almacena una secuencia de elementos en un orden lineal. Cada elemento, conocido como nodo, alberga un dato y una referencia al siguiente nodo en la lista.

Se define una clase Nodo para representar los elementos individuales de la lista, donde cada Nodo contiene un valor y una referencia al siguiente Nodo. Además, se establece una clase Lista que sirve como el ADT. Esta clase mantiene una referencia al primer Nodo en la lista y ofrece métodos para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

Además de las operaciones CRUD estándar, la implementación también incluye métodos para cargar y guardar los datos de la lista a partir de archivos de texto. Esto simula el comportamiento de una base de datos, permitiendo que los datos persistan entre diferentes ejecuciones del programa.

En términos generales, el código proporciona una implementación efectiva de un ADT en forma de lista enlazada, ilustrando cómo los ADTs pueden simplificar el manejo de datos complejos y mejorar la eficiencia del código.

Casos de prueba

Create

Número de Caso de Prueba	Entrada	Salida Esperada	Salida Real
Caso 1	-	1	1
Caso 2	1	1 2	1 2
Caso 3	1 2 3	1 2 3	1 2 3
Caso 4	1 2 3 4	1 2 3 4	1 2 3 4
Caso 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

Read

Número de Caso de Prueba	Entrada	Salida Esperada	Salida Real

Caso 1	1	El valor 1 fue encontrado en la posición: 0x563422cae4a0	El valor 1 fue encontrado en la posición: 0x563422cae4a0
Caso 2	2	El valor 2 fue encontrado en la posición: 0x563422cae8f0	El valor 2 fue encontrado en la posición: 0x563422cae8f0
Caso 3	3	El valor 3 fue encontrado en la posición: 0x563422cae930	El valor 3 fue encontrado en la posición: 0x563422cae930
Caso 4	4	El valor 4 fue encontrado en la posición: 0x563422cae970	El valor 4 fue encontrado en la posición: 0x563422cae970
Caso 5	5	El valor 5 fue encontrado en la posición: 0x563422cae9b0	El valor 5 fue encontrado en la posición: 0x563422cae9b0

Update

Número de Caso de Prueba	Entrada	Salida Esperada	Salida Real
Caso 1	1	6 2 3 4 5	6 2 3 4 5
Caso 2	2	6 7 3 4 5	6 7 3 4 5
Caso 3	3	6 7 8 4 5	6 7 8 4 5
Caso 4	4	6 7 8 9 5	6 7 8 9 5
Caso 5	5	6 7 8 9 10	6 7 8 9 10

Delete

Número de Caso de Prueba	Entrada	Salida Esperada	Salida Real
Caso 1	-	1	1

Número de Caso de Prueba	Entrada	Salida Esperada	Salida Real
Caso 1	6	7 8 9 10	7 8 9 10
Caso 2	7	8 9 10	8 9 10
Caso 3	8	9 10	9 10

Comprobación casos de prueba

```
❯ make
g++ -c main.cpp
g++ -c LinkedList.cpp
g++ -o programa main.o LinkedList.o
./programa
Lista después de crear nodos: 1
Lista después de crear nodos: 1 2
Lista después de crear nodos: 1 2       3
Lista después de crear nodos: 1 2       3       4
Lista después de crear nodos: 1 2       3       4       5
El valor 1 fue encontrado en la posición: 0x563422cae4a0
El valor 2 fue encontrado en la posición: 0x563422cae8f0
El valor 3 fue encontrado en la posición: 0x563422cae930
El valor 4 fue encontrado en la posición: 0x563422cae970
El valor 5 fue encontrado en la posición: 0x563422cae9b0
Lista después de actualizar el nodo con valor 1: 6      2       3       4       5
Lista después de actualizar el nodo con valor 2: 6      7       3       4       5
Lista después de actualizar el nodo con valor 3: 6      7       8       4       5
Lista después de actualizar el nodo con valor 4: 6      7       8       9       5
Lista después de actualizar el nodo con valor 5: 6      7       8       9       10
Lista después de eliminar el nodo con valor 6: 7        8       9       10
Lista después de eliminar el nodo con valor 7: 8        9       10
Lista después de eliminar el nodo con valor 8: 9        10
Lista después de eliminar el nodo con valor 9: 10
Lista después de eliminar el nodo con valor 10:
```

Justificación casos de prueba

Los casos de prueba seleccionados son fundamentales, ya que permiten verificar todas las funcionalidades de la aplicación. Se han definido pasos y entradas específicas para asegurar que todas las funcionalidades operan como se espera. Los cinco casos de prueba mencionados anteriormente cubren las ocho funcionalidades a las que el usuario puede acceder mediante comandos.

Además, existen funcionalidades adicionales, como la lectura y escritura de archivos txt, que aunque no son visibles para el usuario en forma de opciones, su funcionamiento se puede observar a través de las demás funcionalidades implementadas.

Al realizar las pruebas siguiendo los pasos establecidos, se ha confirmado que la aplicación funciona según lo esperado, tanto en términos de inputs como de outputs.