TAREA 1

Desarrollo de aplicaciones avanzadas de ciencias computacionales
(Gpo 502)

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Leonardo González Guerra - A01721434

Octubre 10, 2024

Implementación y Pruebas de Estructuras de Datos en C++

Este documento describe la implementación de tres estructuras de datos básicas en C++: Stack (LIFO), Queue (FIFO) y Dictionary (mapa ordenado). Estas estructuras fueron desarrolladas para soportar las principales operaciones de acceso y manipulación de datos. Para validar su funcionamiento, se diseñaron varios casos de prueba que fueron ejecutados utilizando archivos de entrada .txt, los cuales contienen una serie de comandos a realizar sobre cada estructura. La salida esperada de cada operación fue analizada para verificar la correcta implementación de cada estructura.

Implementación de Estructura de Datos

Stack (LIFO)

La clase Stack implementa una pila (LIFO: Last In, First Out) usando std::vector como contenedor subyacente. Las operaciones principales son:

- push(item): Agrega un elemento al final de la pila.
- pop(): Remueve y devuelve el elemento más reciente agregado.
- peek(): Devuelve el último elemento agregado sin removerlo.
- size(): Retorna el número de elementos en la pila.

Queue (FIFO)

La clase Queue implementa una cola (FIFO: First In, First Out) usando std::queue de la biblioteca estándar. Las operaciones principales son:

- enqueue(item): Agrega un elemento al final de la cola.
- dequeue(): Remueve y devuelve el elemento al frente de la cola.
- front(): Devuelve el elemento al frente de la cola sin removerlo.
- size(): Retorna el número de elementos en la cola.

Dictionary (Ordenado)

La clase Dictionary implementa un mapa ordenado usando std::map, que permite almacenar pares clave-valor y mantener las claves ordenadas automáticamente. Las operaciones principales son:

- set(key, value): Inserta o actualiza un valor asociado a una clave.
- get(key): Devuelve el valor asociado a una clave.
- remove(key): Elimina la clave y su valor asociado.
- size(): Retorna el número de pares clave-valor en el diccionario.

Pruebas y Validación de Funcionalidad

Pruebas para la Clase Stack

Archivo de Prueba: **stack_test1.txt**

Descripción:

1. push 5, push 10, push 15: Agrega los valores 5, 10, y 15 a la pila.
2. pop: Remueve y muestra 15, que es el último elemento agregado.
3. peek: Muestra 10 sin removerlo.
4. size: Muestra 2, indicando que la pila tiene dos elementos (5 y 10).

Objetivo: Validar que las operaciones sigan el comportamiento LIFO, es decir, el último elemento agregado es el primero en ser removido.

Pruebas para la Clase Queue

Archivo de Prueba: **queue_test1.txt**

Descripción:

1. enqueue 3, enqueue 6: Agrega los valores 3 y 6 a la cola.
2. dequeue: Remueve y muestra 3, que es el primer elemento agregado.
3. front: Muestra 6 sin removerlo.
4. size: Muestra 1, indicando que la cola tiene un elemento (6).

Objetivo: Validar que las operaciones sigan el comportamiento FIFO, es decir, el primer elemento agregado es el primero en ser removido.

Pruebas para la Clase Dictionary

Archivo de Prueba: **dictionary_test1.txt**

Descripción:

1. set number 5: Agrega el par clave-valor ("number", 5).
2. set string 10: Agrega el par clave-valor ("string", 10).
3. get number: Muestra 5, el valor asociado a la clave number.
4. remove number: Elimina la clave number del diccionario.
5. size: Muestra 1, indicando que el diccionario tiene una clave (string).

Objetivo: Validar que las operaciones gestionen correctamente la inserción, recuperación, y eliminación de pares clave-valor, asegurando que el diccionario mantenga el comportamiento esperado cuando se manejan claves y valores.