Laboratorio #2: Listas y Colas

Jenny Johanna Vargas Oliveros

Escuela Superior de Ingeniería y Tecnología, Fundación Universitaria Internacional de la Rioja

Ingeniería informática

Algoritmia y Complejidad

Rafael Lizcano Reyes

02 de octubre 2024

Introducción:

Este laboratorio está destinado a poner en práctica la implementación de las estructuras de datos de Pilas y Colas, junto con funciones específicas que permiten interactuar con estas estructuras.

Desarrollo del Laboratorio

Resolveremos dos problemas de mediana complejidad utilizando listas y colas el IDE usado es Visual Studio Code y su visualización se hará con la herramienta en línea de GitHub.
El enunciado del problema es el siguiente:

1. Diseño de las clases Pila y cola con los atributos y métodos necesarios para realizar las siguientes operaciones básicas:
- Para la Pila: crear (), desapilar(), esta_vacia(), imprimir().
- Para la Cola: crear(), encolar(), desencolar(), esta_vacia(), imprimir().
2. Implementación de la función adicional modificar_estructural() que reciba como parámetros una estructura (Pila o Cola) y un número X. La función elimina todos los elementos de la estructura hasta encontrar el valor X, sin eliminar X. Al final muestra el estado final de la estructura después de la modificación.
3. Presentación de menú al usuario que permite elegir entre las diferentes operaciones para la Pila y la Cola, incluye la opción de modificar la estructura.

A continuación se da respuesta a las preguntas planteadas:

A. Diferencia clave entre Pila y Cola.
   - En las pilas **estructura LIFO** (Last in First Out), el último elemento que entra es el primero en salir (borra), como una pila de libros o una de platos.
   - En las Colas **estructura FIFO (First in First Out)**, el primer elemento que entra es el primero en salir, como una cola de banco o cola de vehículos en el semáforo.

- La principal diferencia radica en el orden en que se insertan y se extraen los elementos. Las pilas siguen un orden LIFO, mientras que las colas siguen un orden FIFO.

B. Explicación de las clases y funciones del programa.

La estructura Pila proporciona las siguientes operaciones:

- Push: poner un elemento en la cima.
- Pop: retirar un elemento de la cima.
- Peek: para consultar el elemento en la parte superior sin quitarlo.
- Empty: indica si la pila está vacía.

Las pilas se pueden implementar mediante arrays y mediante listas enlazadas.

La estructura Cola proporciona tres operaciones:

- Enqueue: insertar un elemento al final de la cola.
- Dequeue o desencolar: elimina un elemento de la cabeza de la cola.
- Empty: indica si la cola está vacía.

C. Descripción de la implementación de la función Pilar() para la clase Pila.

- Debemos definir una clase que contenga un array o lista (dependiendo del lenguaje) para almacenar los elementos, junto con las operaciones fundamentales de la pila.
- Luego, debemos inicializar un contenedor vacío en el constructor de la clase, seguidamente implementamos el método push() para agregar elementos al final de la lista y pop() para eliminar y retornar el último elemento ingresado. También definimos el método peek() para consultar el elemento en la parte superior sin quitarlo y Empty() para verificar si la pila está vacía.
- De este modo se gestiona la lógica de la estructura LIFO (Last In, First Out) de la pila.

D. Métodos para la implementación de la función desencolar() en la clase Cola.

- Esta función se implementa para eliminar y retornar el primer elemento, respetando la estructura FIFO (First In, First Out). Para implementarlo en la clase Colas debemos inicializar una estructura de datos, como una lista o una lista enlazada, en el constructor para almacenar los elementos. La función desencolar() o dequeque() verifica primero si la cola está vacía utilizando el método isEmpty(); si está vacía, se puede lanzar una excepción o devolver un valor indicativo como

(noone). En caso de no estar vacía, se elimina el elemento de la primera posición de la estructura y se retorna, manteniendo el orden de llegada. Esta operación es eficiente si usamos una lista enlazada, donde la eliminación del primer nodo es una operación de tiempo constante.

E. Aplicación de la función modificar_estructura() con X = 4 en una Pila, estado final de la Pila en la lista de números [5,3,9,2,8,4,6].
F. Estructura resultante aplicando la función modificar_estructura() con X =7 en una Cola que contiene [1,2,3,4,5,6,7,8,9].
G. Exploración de la eficiencia de la función modificar_estructura() en términos de tiempo de ejecución en relación con el tamaño de la estructura.
    - ¿Cómo se mejora su eficiencia?

Entregables:

- Código fuente del programa con la implementación de la solución, incluye las clases y funciones requeridas, enlace GitHub https://github.com/Johavaroli/Algoritmia_Lab2_Pilas_Colas/blob/main/main.py
- Informe de la actividad.

Bibliografía

- Temarios de la clase de Algoritmia y complejidad Tema6_PilasyColas y Tema8_HeapsyColas.
- Sesiones de clase de Algoritmia y complejidad Tema_ListasEnlazadas.
- *Algoritmos y Estructuras de Datos*. (n.d.). ALGORITMIA ALGO+ - Algoritmos y Estructuras de Datos. Retrieved October 1, 2024, from http://www.algoritmia.net/articles.php?id=15
- *Cover image for Pilas y Colas*. (2022, March 16). DEV Community. Retrieved October 1, 2024, from https://dev.to/szalimben/pilas-y-colas-b3
- *Pilas vs Colas*. (2024, 05 10). Pilas vs Colas FIFO en Java. Retrieved 10 01, 2024, from https://academiasanroque.com/pilas-vs-colas-fifo-en-java-conceptos-y-ejemplos-practicos/