Practica 04

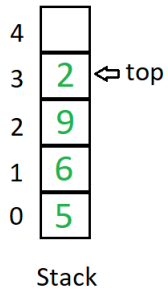
Estructura de Datos y Algoritmos 1

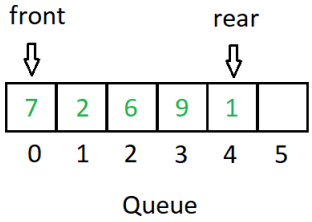
Josué Alexis Campos Negrón
josue.campos@cimat.mx
Universidad de Guanajuato
27 de febrero del 2023

Fecha de entrega: Lunes 06 de marzo.

Problema

Implementar la estructura de *stack* y *queue* utilizando clases y listas enlazadas, es decir, nodos con apuntadores y tamaño dinámico.





Las clase de la **stack** debe contener las siguientes funciones:

- `top()`: Imprime y devuelve el elemento que se encuentra en el *top* de la stack, si el stack está vació, retorna la macro `INT_MIN`.
- `push()`: Inserta un elemento al stack.
- `pop()`: Elimina el elemento que se encuentra en el *top* de la *stack*.
- `size()`: Imprime y devuelve el tamaño de la stack.
- `empty()`: Retorna true en caso que la pila esté vacía y false en caso contrario.
- `last()`: Imprime el último elemento del stack, si el stack está vació, retorna la macro `INT_MIN`..

Las clase de la **queue** debe contener las siguientes funciones:

- `front()`: Imprime y devuelve el elemento que se encuentra en el *front* de la queue, si la queue está vació, retorna la macro `INT_MIN`..
- `push()`: Inserta un elemento a la queue.
- `pop()`: Elimina el elemento que se encuentra en el *front* de la *queue*.
- `size()`: Imprime y devuelve el tamaño de la queue.
- `empty()`: Retorna true en caso que la queue esté vacía y false en caso contrario.
- `last()`: Imprime el último elemento del queue, si la queue está vació, retorna la macro `INT_MIN`..

Ambas clases deben contener dos constructores, una sin parámetros de entrada y otra con una variable de tipo `int` como parámetro el cual será el primer elemento a insertar al igual que el destructor que libere memoria.

Entregable

Se entrega un archivo comprimido `.zip` con el siguiente formato `Apellido1Apellido2_Practica01.zip` el cual contiene un folder con los siguientes documentos.

- Un reporte tipo `pdf` con nombre `Apellido1Apellido2_Practica01.pdf`. En la sección **Reporte** se detalla los requisitos del `pdf`.
- Dos archivos `.cpp` con nombre `stack.cpp` y `queue.cpp` en donde vendrán las implementaciones de la clase *stack* y la clase *queue* respectivamente.
- Imagenes de evidencia del `output` que demuestre el correcto funcionamiento de las implementaciones. Las imagenes deben tener nombres del estilo `stackIMGn.png` y `queueIMGn.png` donde `n` es el número de imagen.

Reporte

El reporte consta de cinco secciones las cuales son:

1. **Stack:** Explica la lógica de la estructura *stack*. Además explica la lógica de la implementación de la estructura. Por último, explica un ejemplo en donde puedas utilizar la estructura de datos *stack* y no una *queue*.
2. **Queue:** Explica la lógica de la estructura *queue*. Además explica la lógica de la implementación de la estructura. Por último, explica un ejemplo en donde puedas utilizar la estructura de datos *queue* y no una *stack*.
3. **Problemas encontrado**: Mención de los problemas encontrados (si es que hubieron) al momento de implementar los algoritmos o comprenderlos.
4. **Conclusión**: Resumen breve de lo aprendido y posibles aplicaciones en la vida cotidiana.
5. **Referencias**: Enunciar las referencias utilizadas para la realización de la practica.

Código

Para el caso de la implementación de la clase **stack** tenemos el siguiente input y output.

Input: Recibe un entero Q, seguido de ello tenemos Q líneas las cuales representan operaciones a realizar en el stack las cuales estan representadas de la siguiente forma:

- **T**: Imprime el elemento *top* del stack si no está vacía.
- **H**: Inserta un elemento al stack.
- **P**: Elimina el elemento *top* del stack si no está vacía.
- **S**: Imprime el tamaño del stack.
- **E**: Imprime 1 si la pila está vacía y 0 si no.
- **L**: Imprime el último elemento del stack.

Output: El output es el esperado deacuerdo al tipo de *operación* indicado en el input, cada una separada por un salto de línea.

Para el caso de la implementación de la clase **queue** tenemos el siguiente input y output.

Input: Recibe un entero Q, seguido de ello tenemos Q líneas las cuales representan operaciones a realizar en la queue las cuales estan representadas de la siguiente forma:

- **F**: Imprime el elemento *front* de la queue si no está vacía.
- **H**: Inserta un elemento a la queue.
- **P**: Elimina el elemento *front* de la queue si no está vacía.
- **S**: Imprime el tamaño de la queue.
- **E**: Imprime 1 si la queue está vacía y 0 si no.
- **L**: Imprime el último elemento de la queue.

Output: El output es el esperado deacuerdo al tipo de *operación* indicado en el input, cada una separada por un salto de línea.

Comentar las piezas fundamentales del código.