➢ OBJETIVO

Revisarás las definiciones, características, procedimientos y ejemplos de las estructuras lineales Pila y Cola, con la finalidad de que comprendas sus estructuras y puedas implementarlas.

➢ INTRODUCCIÓN

Una **pila** es una estructura de datos en la cual los elementos almacenados en la misma se agregan y se sacan del mismo lugar, llamado el tope de la pila. El tope es el único lugar a partir del cual se pueden acceder a los elementos de la estructura. Esta característica hace que el último elemento en ser insertado en la pila es el primero en salir. Este tipo de estructuras se denominan LIFO (Last In First Out).

Para utilizar el Tipo de Dato Abstracto (TDA) Pila, el mismo nos proveerá de una serie de procedimientos que nos permitirán acceder o agregar elementos.

Los siguientes son los procedimientos básicos que debe contener una pila:

- P_Crear
- P_Vaciar
- P_Vacia
- P_Agregar
- P_Sacar

El usuario de una pila utilizara estos procedimientos, sin tener en cuenta como están implementados por dentro. Lo único que deberá conocer es los procedimientos y sus parámetros.

Una **cola** es una estructura de datos en la cual los elementos almacenados en la misma se agregan al final y se sacan del principio de la cola. Esta característica hace que el primer elemento insertado en la cola es el primero en salir, como en cualquier cola de la realidad (en un banco, en el cine, en el colectivo). Este tipo de estructuras se denominan FIFO (First In First Out).

Para utilizar el Tipo de Dato Abstracto (TDA) Pila, el mismo nos proveerá de una serie de procedimientos que nos permitirán acceder o agregar elementos.

Los siguientes son los procedimientos básicos que debe contener una pila:

- C_Crear
- C_Vaciar
- C_Vacia
- C_Agregar
- C_Sacar

- **DESARROLLO**

• Describa 3 ejemplos donde se apliquen colas y 3 donde se apliquen listas.

Pilas:

1. En la verificación de sintaxis del compilador, un proceso es verificar si varios corchetes coinciden, como ([]), que coincide, pero {[}] no coincide. Puede utilizar la pila para hacer coincidir los paréntesis.
   El algoritmo específico es el siguiente:
   Crea una pila vacía.
   mientras (el archivo no termina) {
   Leer un carácter.
   si encuentra un paréntesis de apertura, colóquelo en la pila.
   de lo contrario, si se encuentra el paréntesis de cierre, verifique la pila, {
   si la pila está vacía, informe un error y finalice el programa (los corchetes no coinciden).
   de lo contrario, si la pila no está vacía, entonces {
   si la parte superior de la pila no es el paréntesis izquierdo correspondiente, se informa de un error y se termina el programa.
   Abre la parte superior de la pila.
   }
   }
   si la pila no está vacía, se informa de un error.
2. Si queremos calcular 6 + 4 * 8, tenemos que tener en cuenta el tema de la prioridad, entonces se puede usar la pila.
   Primero, debemos construir la fórmula algebraica en 6 4 8 * + (el método de construcción también usa la pila, que se discutirá en el próximo artículo). Lea los datos uno por uno, cuando se lea el número, ponga el número en la pila,
   Cuando se lee el operador, se abren dos elementos en la pila (porque aquí hay un operador binario, por lo que se abren dos, si es un operador unario como sin, se abre uno), Según lectura
   El operador de realiza la operación, inserta el resultado en la pila y luego continúa leyendo los datos. Una vez completada la lectura, el elemento superior de la pila es el resultado.
   Por ejemplo, si lee 6, 4 y 8, porque son números, se introducen en la pila uno por uno.
   Cuando se lea "*", saque 4 y 8, multiplique para obtener 32 y ponga 32 en la pila. Cuando lees "+", muestra 6 y 32, ejecuta el cálculo para obtener 38, lo coloca en la pila y luego finaliza la lectura. El 38 en la parte superior de la pila es el resultado.

3. Debido a que la CPU solo puede ejecutar un comando a la vez, y los registros también son públicos,
   Cuando la función actual current () se está ejecutando, los datos se almacenan en el registro. Si se va a llamar a otra función target (), y target () también requiere el uso de registros, para evitar la pérdida de datos y después de ejecutar target ()
   puede volver a current () para continuar con la ejecución. En este momento, los datos importantes de la función actual deben almacenarse y enviarse a la pila en la memoria (incluidos los valores de las variables y las direcciones de las funciones) . De esta forma, la función target () puede utilizar registros sin escrúpulos.
   Cuando se ejecuta la función target (), tome la dirección de retorno en la parte superior de la pila y continúe ejecutando current (). Si se llama a otra función en target (), la operación correspondiente es la misma.
   Este mecanismo es similar a la coincidencia de corchetes: una llamada de función es como un corchete de apertura y un retorno de función es como un corchete de cierre.
   Este mecanismo es el principio de recursividad. La dirección de retorno recursiva es usted mismo. (Esta frase puede ser problemática, así es como la entiendo).
   El espacio de la pila es limitado, si la recursividad no tiene una condición final, continuará empujando la pila y luego la pila se desbordará y el programa saldrá mal.

Colas:

1. Cuando se asignan varias tareas a la impresora, para evitar conflictos, se crea una cola, las tareas se ponen en cola y las tareas se procesan de acuerdo con el principio de primero en entrar, primero en salir.
   Cuando varios usuarios desean acceder a archivos en el servidor remoto, las colas también se utilizan para cumplir con el principio de primero en llegar, primero en ser atendido.
2. Hay una rama de las matemáticas llamada teoría de las colas. Se utiliza para calcular y predecir el tiempo de espera de los usuarios en la cola, la longitud de la cola, etc.
   La respuesta depende de la frecuencia con la que el usuario llega a la cola y el tiempo de procesamiento de la tarea del usuario. Si la situación es relativamente simple, se puede resolver únicamente mediante análisis. Un ejemplo simple es
   Un teléfono, un operador. Cuando ingrese una llamada, si el operador está ocupado, coloque la llamada detrás de la línea de espera. El operador respondió desde el principio de la línea.

3. Operación de suma polinomial de una variable
   Idea de algoritmo:
   usa una lista enlazada individualmente sin un nodo principal y organiza los elementos en el orden de exponente decreciente. Se suman los coeficientes de los términos del mismo índice y se copia el resto.
   Los dos punteros P1 y P2 apuntan respectivamente al primer nodo de los dos polinomios y se repiten continuamente.
   (1) P1-> expon == P2-> expon: suma de coeficiente, si el resultado no es 0, se utilizará como el coeficiente del término correspondiente del polinomio de resultado. Al mismo tiempo, tanto P1 como P2 apuntan al siguiente elemento respectivamente;
   (2) P1-> expon> P2-> expon: guarda el término actual de P1 en el polinomio resultante y haz que P1 apunte al siguiente término;
   (3) P1-> expon <P2-> expon: guarda el término actual de P2 en el polinomio de resultado y haz que P2 apunte al siguiente término;
   Cuando se procesa un determinado polinomio, todos los nodos de otro polinomio se copian secuencialmente al polinomio resultante.

➢ CONCLUSIÓN

Se cumplió con el objetivo de esta práctica ya que comprendimos y nos acercamos más a las pilas y colas lineales a través de sus definiciones, característica y ejemplos, por lo cual aprendimos a implementarlas en diferentes situaciones y problemas

➢ BIBLIOGRAFÍA CONSULTADA

Carolo, G. (2005). *Algoritmos y Programación II.* Recuperado el 06 de agosto de 2021, de https://algo2.files.wordpress.com/2008/08/ayp2_listas_pilas_y_colas.pdf