Anexo II .- Pilas y Colas

Un tipo de dato abstracto (**TDA**) o tipo abstracto de datos (**TAD**) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo. Un TDA es representado por su interfaz, la cual sirve como cubierta a la correspondiente implementación. La idea es que los usuarios de un TDA solo tengan que preocuparse por la interfaz, pero no por la implementación, puesto que esta puede ir cambiando a lo largo del tiempo y, si no existiera **encapsulación**, afectar los programas que usan el dato. Esto se basa en el concepto de **ocultamiento de información**, una protección para el programa de decisiones de diseño que son objeto de cambio.
Resumiendo, podemos concluir que un TDA es una estructura de datos de acceso restringido (el acceso a los datos de la estructura está limitado por el conjunto de operaciones definidas).

Algunos de los **TDA** más utilizados en programación son:

- **Pilas y Colas**: que son una implementación de los algoritmos FIFO (First In First Out) y LIFO (Last In First Out).
- **Conjuntos**: implementación de conjuntos con sus operaciones básicas (unión, intersección y diferencia), operaciones de inserción, borrado, búsqueda, etc.
- **Árboles binarios de búsqueda**: implementación de árboles de elementos, utilizados para la representación interna de un gran volumen de datos complejos donde se hace necesario que las operaciones de búsqueda sean rápidas.
- **Grafos**: implementación de grafos, una serie de vértices unidos mediante una serie de arcos o aristas. Muy utilizados para encontrar el camino más corto.

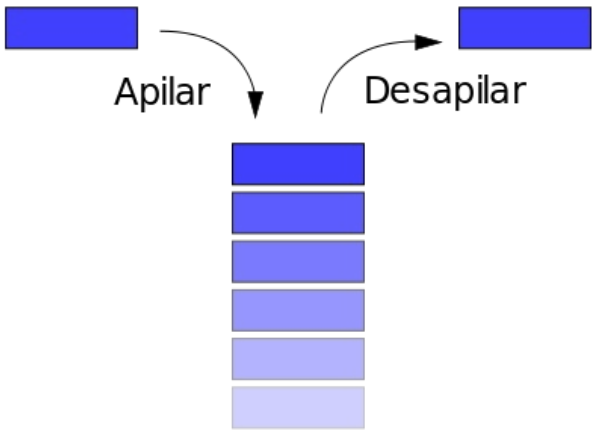
Pila

Una pila (**stack** en inglés) es un TDA que permite almacenar y recuperar datos, la forma de acceso a sus elementos es de tipos LIFO (del inglés Last In, First Out, «último en entrar, primero en salir») . Esta estructura se aplica en multitud de supuestos en el área de informática a causa de su simplicidad y capacidad de dar respuesta a numerosos procesos.

Para el manejo de los datos cuenta con dos operaciones básicas:

- apilar (**push**), que coloca un objeto en la pila
- y su operación inversa, retirar (o desapilar, **pop**), que retira el último elemento apilado.

Tan solo se tiene acceso a la parte superior de la pila (acceso restringido), es decir, al último objeto apilado (denominado **TOS**, Top of Stack en inglés). La operación retirar permite la obtención de este elemento, que es retirado de la pila permitiendo el acceso al anterior (apilado con anterioridad), que pasa a ser el último, el nuevo TOS.



Los TDA pila suelen utilizarse en los siguientes contextos:

- Evaluación de expresions en notación postfija (notación polaca inversa).
- Reconocedores sintácticos.
- Implementación de recursividad.
- Implementación de procesos en los sistemas operativos. Cada proceso tiene una zona llamada pila para almacenar valores y llamadas a funciones (paso de parámetros).

Operaciones

Habitualmente, junto a las operaciones básicas de apilar (push) y desapilar (pop), las pilas suelen disponer otras operaciones:

- Crear (**constructor**): crea una pila vacía.
- Apilar (**push**): añade un elemento a la pila.
- Desapilar (**pop**): lee y retira el elemento superior de la pila.
- Tamaño (**size**): devuelve el número de elementos que hay a la pila.
- Leer último (**top**): lee el elemento superior de la pila sin retirarlo.
- Vacía (**empty**): devuelve verdadero si la pila está vacía o falso en caso de que contenga algún elemento.

Implementación

La implementación del TDA pila depende del lenguaje de programación empleado, puesto que no todos disponen de las mismas estructuras (estáticas y dinámicas).

Con estructuras estáticas

Una de las formas de implementar el TDA pila es mediante arrays. El problema de los arrays es que son una estructura estática, es decir, su tamaño está determinado en el momento de la creación y no puede crecer una vez se ha creado el array.

Para implementar el TDA pila haciendo uso de arrays seguiremos los siguientes pasos:

- Crear un array de tamaño suficiente como para albergar inicialmente un volumen de datos suficiente.
- Definir una variable de tipo entero que contenga la posición del último elemento de la pila. En caso de estar vacía su valor será -1.
- Implementar el constructor y las operaciones push, pop, size, top y empty.

Ventajas de la implementación estática:

- Al ser una estructura estática (array) las posiciones en memoria son consecutivas y por tanto el acceso a sus elementos es muy rápido.

Inconvenientes de la implementación estática:

- Las estructuras estáticas no pueden crecer en tiempo de ejecución. Por lo tanto, disponenmos de dos alternativas al crear una pila mediante arrays:
  - Crear inicialmente un array muy grande, desperdiciaremos memoria la mayor parte del tiempo.
  - Crear array pequeños y en la operación apilar, si no queda espacio, crear un array más grande y copiar todos los elementos del antiguo al nuevo.

Con estructuras dinámicas

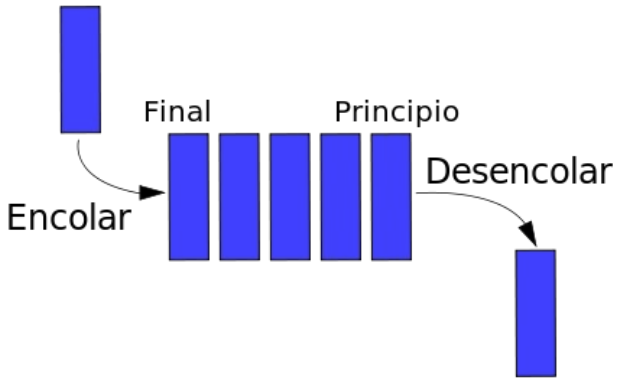
Si el lenguaje de programación que estamos utilizando dispone de estructuras dinámicas, podemos implementar el TDA pila haciendo uso de ellas.

En Java existen varias formas de representar estructuras dinámicas, pero para representar un TDA pila podemos hacer uso de ArrayList y LinkedList, lo veremos más adelante.

Colas

Una cola (en inglés queue) es un TDA caracterizado para ser una estructura FIFO (del inglés First In First Out), por el hecho de que el primer elemento a entrar, será también el primero a salir. Tiene dos operaciones básicas:

- la operación de inserción (encolar o push) se realiza por un extremo.
- y la operación de extracción (desencolar o pop) por el otro extremo.



Los TDA cola suelen utilizarse en los siguientes contextes:

- Sistemas informáticos
- Transportes y operaciones de investigación (entre otras), donde los objetos, personas o acontecimientos son tomados como datos que se almacenan y se guardan mediante colas para su posterior procesamiento.

Operaciones

Habitualmente, junto a las operaciones básicas de encolar (push) y desencolar (pop), las pilas suelen disponer otras operaciones:

- Crear (**constructor**): crea una cola vacía.
- Encolar (**add**): añade un elemento a lfinal de la cola.
- Desencolar (**remove**): lee y retira el elemento frontal de la cola, es decir, el primer elemento que entró.
- Tamaño (**size**): devuelve el número de elementos que hay a la cola.
- Frente (**peek**): lee el elemento frontal de la cola, es decir, el primer elemento que entró.
- Vacía (**empty**): devuelve verdadero si la cola está vacía o falso en caso de que contenga algún elemento.

Implementación

La implementación del TDA cola depende del lenguaje de programación empleado, puesto que no todos disponen de las mismas estructuras (estáticas y dinámicas).

Con estructuras estáticas

Una de las formas de implementar el TDA cola es mediante arrays.

Para implementar el TDA cola haciendo uso de arrays seguiremos los siguientes pasos:

- Crear un array de tamaño suficiente como para albergar inicialmente un volumen de datos suficiente.

- Definir una variable (frente) de tipo entero que contenga la posición del primero de la cola. En caso de estar vacía su valor será 0.
- Definir una variable (ultimo) de tipo entero que contenga la posición del último elemento de la cola. En caso de estar vacía su valor será 0.
- Implementar el constructor y las operaciones add, remove, size, peek y empty.

Ventajas de la implementación estática:

- Al ser una estructura estática (array) las posiciones en memoria son consecutivas y por tanto el acceso a sus elementos es muy rápido.

Inconvenientes de la implementación estática:

- Las estructuras estáticas no pueden crecer en tiempo de ejecución. Por lo tanto tenemos dos alternativas al crear una cola mediante arrays:
  - Crear inicialmente un array muy grande, desperdiciaremos memoria la mayor parte del tiempo.
  - Crear array pequeños y en la operación add, si no queda espacio, crear un array más grande y copiar todos los elementos del antiguo al nuevo.

Con estructuras dinámicas

Si el lenguaje de programación que estamos utilizando dispone de estructuras dinámicas, podemos implementar el TDA cola haciendo uso de ellas.

En Java tenemos la interfaz Queue y varías clases que la implementas como por ejemplo PriorityQueue.