Annex II.- Piles i Cues

Un tipus de dada abstracta (**TDA**) o tipus abstracte de dades (**TAD**) és un model matemàtic compost per una col·lecció d'operacions definides sobre un conjunt de dades per al model. Un TDA és representat per la seua interfície, la qual serveix com a coberta a la corresponent implementació. La idea és que els usuaris d'un TDA hagen de preocupar-se només per la interfície, però no per la implementació, ja que aquesta pot anar canviant amb el temps i, si no existira **encapsulació**, afectar els programes que usen la dada. Això es basa en el concepte d'**ocultació d'informació**, una protecció per al programa de decisions de disseny que són objecte de canvi.
Resumint, podem concloure que un TDA és una estructura de dades d'accés restringit (l'accés a les dades de l'estructura està llimitat per el conjunt d'operacions definides.

Alguns dels **TDA** més utilitzats en programació són:

- **Piles i Cues**: que són una implementació dels algorismes FIFO (First In First Out) i LIFO (Last In First Out).
- **Conjunts**: implementació de conjunts amb les seues operacions bàsiques (unió, intersecció i diferència), operacions de inserció, esborrat, cerca, etc.
- **Arbres binaris de cerca**: implementació d'arbres d'elements, utilitzats per a la representació interna d'un gran volum de dades complexes on es fa necessari que les operacions de cerca siguen ràpides.
- **Grafs**: implementació de grafs, una sèrie de vèrtex units mitjançant una sèrie d'arcs o arestes. Molt utilitzats per trobar el camí més curt.

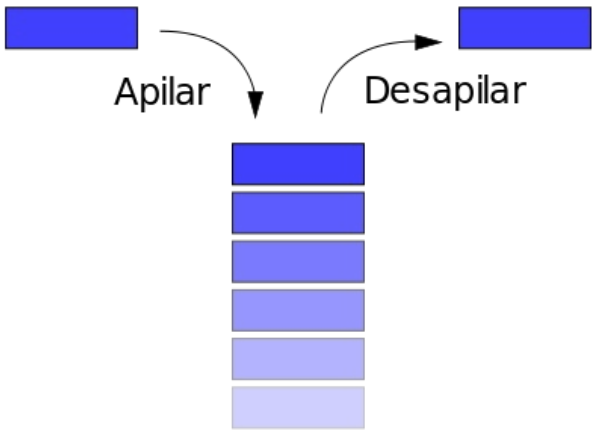
Pila

Una pila (**stack** en anglès) és un TDA que permet emmagatzemar i recuperar dades, la manera d'accés als seus elements és de tipus LIFO (de l'anglès Last In, First Out, «últim a entrar, primer a sortir») . Aquesta estructura s'aplica en multitud de supòsits en l'àrea d'informàtica a causa de la seva simplicitat i capacitat de donar resposta a nombrosos processos.

Per al maneig de les dades compta amb dues operacions bàsiques:

- apilar (**push**), que col·loca un objecte en la pila
- i la seva operació inversa, retirar (o desapilar, **pop**), que retira l'últim element apilat.

Només es té accés a la part superior de la pila (accés restringit), és a dir, a l'últim objecte apilat (denominat **TOS**, Top of Stack en anglès). L'operació retirar permet l'obtenció d'aquest element, que és retirat de la pila permetent l'accés a l'anterior (apilat amb anterioritat), que passa a ser l'últim, el nou TOS.



Els TDA pila solen utilitzar-se en els següents contextes:

- Avaluació d'expresions en notació postfixa (notació polaca inversa).
- Reconeixedors sintàctics
- Implementació de recursivitat.
- Implementació de processos en els sistemes operatius. Cada procés té una zona anomenada pila per a emmagatzemar valors i cridades a funcions (pas de paràmetres).

Operacions

Habitualment, junt a les operacions bàsiques d'apilar (push) i desapilar (pop), les piles solen disposar d'altres operacions:

- Crear (**constructor**): crea una pila buida.
- Apilar (**push**): afegix un element a la pila.
- Desapilar (**pop**): llig i retira l'element superior de la pila.
- Grandària (**size**): retorna el nombre d'elements que hi ha a la pila.
- Llegir últim (**top**): llig l'element superior de la pila sense retirar-lo.
- Buida (**empty**): retorna vertader si la pila està buida o fals en cas que continga algún element.

Implementació

La implementació del TDA pila depén del llenguatge de programació emprat, ja que no tots disposen de les mateixes estructures (estàtiques i dinàmiques).

Amb estructures estàtiques

Una de les formes d'implementar el TDA pila es mitjançant arrays. El problema dels arrays és que són una estructura estàtica, es a dir, la seua grandària està determinada en el moment de la creació i no pot creixer una vegada s'ha creat l'array.

Per a implementar el TDA pila fent ús d'arrays seguirem els següents passos:

- Crear un array de grandària suficient com per a albergar inicialment un volum de dades suficient.
- Definir una variable de tipus enter que continga la posició de l'últim element de la pila. En cas de estar buida el seu valor serà -1.
- Implementar el constructor i les operacions push, pop, size, top i empty.

Avantatges de la implementació estàtica:

- Al ser una estructura estàtica (array) les posicions en memòria són consecutives i per tant l'accés als seus elements és molt ràpid.

Inconvenients de la implementació estàtica:

- Les estructures estàtiques no poden creixer en temps d'execució. Per tant tenim dos alternatives al crear una pila mitjançant arrays:
  - Crear inicialment un array molt gran, malgastarem memòria la major part del temps.
  - Crear array menuts i en l'operació apilar, si no queda lloc, crear un array més gran i copiar tots els elements de l'antic al nou.

Amb estructures dinàmiques

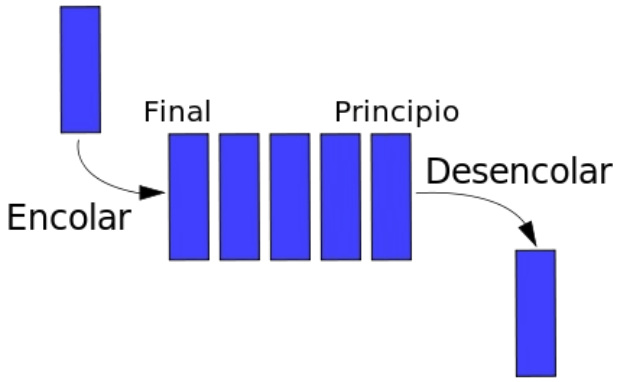
Si el llenguatge de programació que estem utilitzant disposa d'estructures dinàmiques, podem implementar el TDA pila fent ús d'elles.

En Java hi han diverses formes de representar estructures dinàmiques, però per a representar un TDA pila podem fer ús de ArrayList i LinkedList, ho vorem més endavant.

Cues

Una cua (en anglés queue) és un TDA caracteritzat per ser una estructura FIFO (de l'anglès First In First Out), pel fet que el primer element a entrar serà també el primer a sortir. Te dos operacions bàsiques:

- l'operació d'inserció (encolar o **push**) es realitza per un extrem.
- i l'operació d'extracció (desencolar o **pop**) per l'altre extrem.



Els TDA cua solen utilitzar-se en els següents contextes:

- Sistemes informàtics
- Transports i operacions de recerca (entre altres), on els objectes, persones o esdeveniments són presos com a dades que s'emmagatzemen i es guarden mitjançant cues per al seu posterior processament.

Operacions

Habitualment, junt a les operacions bàsiques d'encolar (push) i desencolar (pop), les piles solen disposar d'altres operacions:

- Crear (**constructor**): crea una cua buida.
- Encolar (**add**): afegix un element al final de la cua.
- Desencolar (**remove**): llig i retira l'element frontal de la cua, es a dir, el primer element que va entrar.
- Grandària (**size**): retorna el nombre d'elements que hi ha a la cua.
- Front (**peek**): llig l'element frontal de la cua, es a dir, el primer element que va entrar.
- Buida (**empty**): retorna vertader si la cua està buida o fals en cas que continga algún element.

Implementació

La implementació del TDA cua depén del llenguatge de programació emprat, ja que no tots disposen de les mateixes estructures (estàtiques i dinàmiques).

Amb estructures estàtiques

Una de les formes d'implementar el TDA cua es mitjançant arrays.

Per a implementar el TDA cua fent ús d'arrays seguirem els següents passos:

- Crear un array de grandària suficient com per a albergar inicialment un volum de dades suficient.
- Definir una variable (front) de tipus enter que continga la posició del primer de la cua. En cas

d'estar buida el seu valor serà 0.

- Definir una variable (ultim) de tipus enter que continga la posició de l'últim element de la cua. En cas d'estar buida el seu valor serà 0.
- Implementar el constructor i les operacions add, remove, size, peek i empty.

Avantatges de la implementació estàtica:

- Al ser una estructura estàtica (array) les posicions en memòria són consecutives i per tant l'accés als seus elements és molt ràpid.

Inconvenients de la implementació estàtica:

- Les estructures estàtiques no poden creixer en temps d'execució. Per tant tenim dos alternatives al crear una cua mitjançant arrays:
  - Crear inicialment un array molt gran, malgastarem memòria la major part del temps.
  - Crear array menuts i en l'operació add, si no queda lloc, crear un array més gran i copiar tots els elements de l'antic al nou.

Amb estructures dinàmiques

Si el llenguatge de programació que estem utilitzant disposa d'estructures dinàmiques, podem implementar el TDA cua fent ús d'elles.

En Java tenim la interfície Queue i varies classes que la implementes com per exemple PriorityQueue.