Estructura de Datos Lineales

ESTRUCTURA DE DATOS

Generalidades

- Las estructuras lineales de datos se caracterizan porque sus elementos están en secuencia, relacionados en forma lineal, uno luego del otro.

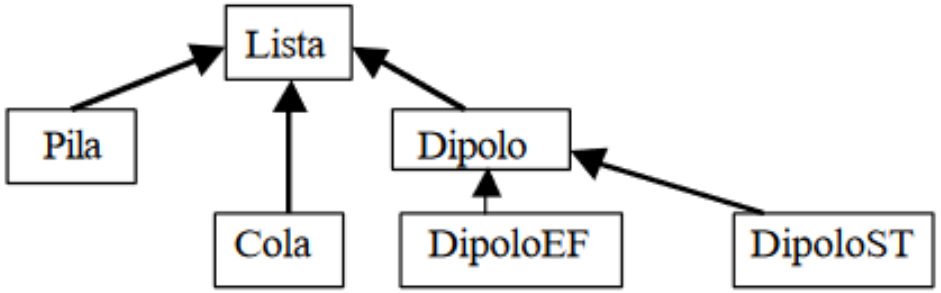
- Cada elemento de la estructura puede estar conformado por uno o varios sub-elementos o campos que pueden pertenecer a cualquier tipo de dato, pero que normalmente son tipos básicos.

- Una estructura lineal de datos esta conformada elementos que tienen una relación dónde existe un primer elemento, seguido de un segundo elemento y así sucesivamente hasta llegar al último.

- El valor contenido en los elementos pueden ser el mismo o diferente.

Jerarquía de clases para las listas

- Las listas de acceso restringido son las pilas, colas y dipolos.
  - **Pilas:** las operaciones de acceso se realizan por un único extremo de la lista, al cual normalmente se denomina tope de la pila.
  - **Colas:** estas operaciones se realizan por ambos extremos de la lista llamados generalmente, inicio y fin de la cola.
  - **Dipolos:** son colas dobles, las operaciones se realizan también por ambos extremos de la lista, en este caso todas las operaciones se pueden hacer por ambos extremos, es decir se puede insertar o eliminar elementos por el tope o por el fin.
- Las listas de acceso no restringido, denominadas listas, son el tipo más general, al cual se considera como la superclase de las otras clases de listas, en específico de las pilas, colas y dipolos.



Propiedades de las listas

- Cada elemento de la lista tiene asignado un tipo de dato. Si l es del tipo ListaDe [TipoEle: e] entonces e1, e2, …, en que conforman la lista l cuyos elementos tienen asignado un mismo tipo.

- Las propiedades de las listas son:

  1) Si $n = 0$ entonces la lista está vacía
  2) Si $n \geq 1$ entonces e_1 es el primer elemento de la lista y e_n el último.
  3) e_i es el predecesor de e_{i+1} y el sucesor de e_{i-1} con $1 \leq i \leq n$

- Las listas se pueden clasificar por varios criterios:
  - **Por el orden de sus elementos sobre la base de un subelemento:** ordenadas (ascendente o descendente), y desordenadas.
  - **Por el método de almacenamiento:** secuencial y enlazada (simple, doble, simple circular y doble circular).

Representación lógica de las estructuras lineales de datos

- Las listas se representan en forma lógica como una secuencia de elementos.
- Una lista se representa lógicamente como:

- Cada tipo de lista tiene la misma representación.

Pilas

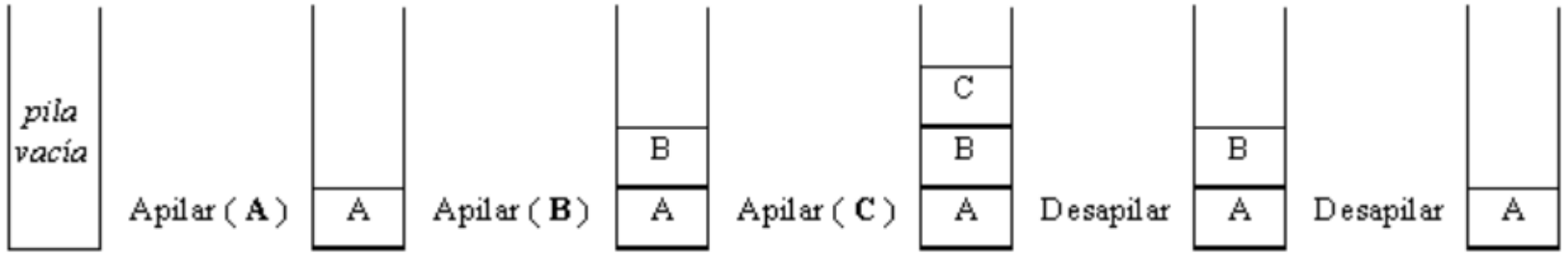
- Una pila es un subtipo de las listas donde el acceso está restringido a un solo extremos de la lista, en este caso al tope de la misma.

- Las operaciones básicas sobre una pila son: crearla, destruirla, agregar un nuevo elemento, suprimir un elemento, consultar el elemento del tope y verificar si está vacía.

- Sobre la base de estas operaciones se especifíca el TAD Pila como se muestra en la siguiente figura. Esta especificación incluye operaciones que pueden ser extendidas en la implementación para soportar otras operaciones útiles de acuerdo a las aplicaciones que la puedan utilizar.

- En particular, toda implementación debe contener las operaciones básicas definidas para el tipo y puede ser ampliada con otras adicionales.

También llamadas LIFO: Last In First Out (Último en Entrar Primero en Salir).



http://webdiis.unizar.es/~elvira/eda/addeda/teruel/Pilaconcepto2.swf

Pilas

Especificación Pila[TipoEle]		
1	**Especificación sintáctica:** creaPila()⟶ Pila, meterElePila(Pila,TipoEle)⟶ Pila, sacarElePila(Pila)⟶ Pila, conTopePila(Pila)⟶ TipoEle, vacíaPila(Pila)⟶ Lógico, destruyePila(Pila)⟶ .	**-creaPila()**: Crea una pila vacía. **-meterElePila()**: Ingresa un nuevo elemento a la pila por el tope de la misma. **-sacarElePila()**: Elimina el elemento que está actualmente en el tope de la pila. Si la pila está vacía no hace ninguna eliminación. **-vacíaPila()**: Regresa Verdadero si la pila está vacía. **-destruyePila()**: Destruye la pila. **-conTopePila()**: Devuelve el elemento que se encuentra actualmente en el tope de la pila. Si la pila está vacía devuelve un valor especial que lo indica.
2	**Declaraciones:** TipoEle: e, {TipoEleNoDef}	
3	**Especificación semántica:** vacíaPila(creaPila())=Verdadero vacíaPila(meterElePila(creaPila(),e))=Falso conTopePila(creaPila())={TipoEleNoDef} sacarElePila(creaPila())=creaPila()	

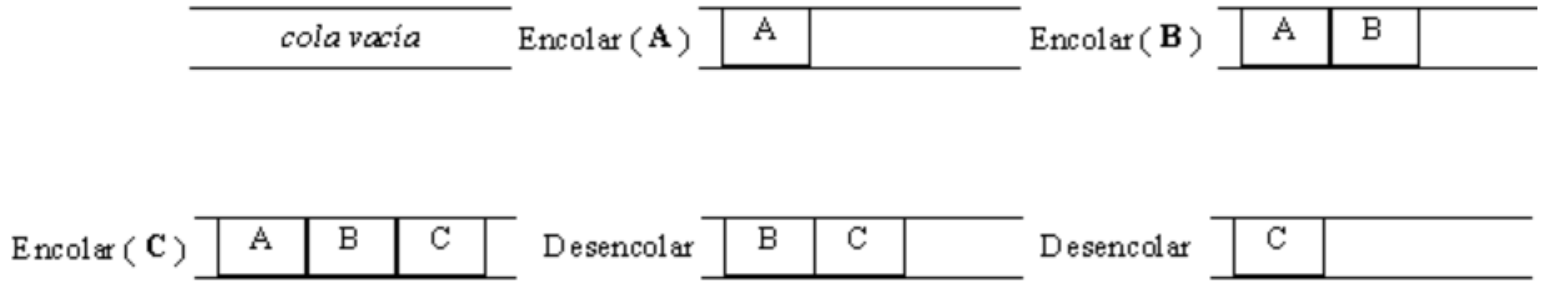
Colas

- Una cola es otro subtipo de las listas donde el acceso está restringido a los extremos de la lista, es decir al inicio y al fin de la misma.

- Similar al subtipo pila, en la cola las operaciones básicas son: creación, destrucción, inserción al final de un nuevo elemento, eliminación del inicio de un elemento, consultar que elemento está al inicio y cual al final, y verificar si la cola está vacía.

- Según estas operaciones se especifica el TAD Cola que se muestra en la siguiente figura.

También llamadas FIFO: First In First Out (Primero en Entrar, Primero en Salir).



http://webdiis.unizar.es/~elvira/eda/addeda/teruel/Cola.swf

Colas

Especificación Cola[TipoEle]		
1	**Especificación sintáctica:** creaCola()⟶ Cola, entrarCola(Cola,TipoEle)⟶ Cola, salirCola(Cola)⟶ Cola, conEleCola(Cola)⟶ TipoEle, vacíaCola(Cola)⟶ Lógico, destruyeCola(Cola)⟶ .	**-creaCola()**: Crea una cola vacía. **-entrarCola()**: Ingresa un nuevo elemento a la cola por el fin de la misma. **-salirCola()**: Elimina el elemento que está actualmente en el inicio de la cola. Si la cola está vacía no hace ninguna eliminación. **-destruyeCola()**: Destruye la cola. **-vacíaCola()**: Regresa Verdadero si la cola está vacía. **-conEleCola()**: Devuelve el elemento que se encuentra actualmente en el inicio de la cola. Si la cola está vacía devuelve un valor especial que lo indica.
2	**Declaraciones:** TipoEle: e, {TipoEleNoDef}	
3	**Especificación semántica:** vacíaCola(creaCola())=Verdadero vacíaCola(meterEleCola(creaCola(),e))=Falso conEleCola(creaCola())={TipoEleNoDef} sacarEleCola(creaCola())=creaCola()	

Dipolo

- Esta estructura equivale a dos colas colocadas una en un sentido y la otra en sentido contrario, por ello las operaciones de inserción y eliminación se pueden realizar por ambos extremos.
- Dos casos especiales se pueden tener
  - **Dipolo de entrada restringida:** donde sólo se puede insertar por un extremo y eliminar por ambos.
  - **Dipolo de salida restringida:** donde se puede insertar por ambos extremos y sólo se puede suprimir por un extremo.

  Se llamará a estos extremos como izquierdo (izq) y derecho (der).

- Sus operaciones básicas son: creación, destrucción, verificación de dipolo vacío, inserción de un nuevo elemento por la izquierda, inserción por la derecha, eliminación por la izquierda, eliminación por la derecha, consulta del elemento que está más a la izquierda y del que está más a la derecha.

- La siguiente figura se presenta la especificación de este subtipo de lista.

Dipolo

Especificación Dipolo[TipoEle]		
1	**Especificación sintáctica:** creaDipolo()⟶ Dipolo, insIzqDipolo(Dipolo,TipoEle)⟶ Dipolo, insDerDipolo(Dipolo,TipoEle)⟶ Dipolo, eliIzqDipolo(Dipolo)⟶ Dipolo, eliDerDipolo(Dipolo)⟶ Dipolo, conIzqDipolo(Dipolo)⟶ TipoEle, conDerDipolo(Dipolo)⟶ TipoEle, vacíoDipolo(Dipolo)⟶ Lógico, destruyeDipolo(Dipolo)⟶ .	**-creaDipolo()**: Crea un Dipolo vacío. **-insIzqDipolo(), insDerDipolo()**: Ingresa un nuevo elemento al Dipolo por la izquierda o por la derecha del mismo, respectivamente. **-eliIzqDipolo(), eliDerDipolo()**: Elimina el elemento actual a la izquierda o a la derecha del Dipolo. Si el Dipolo está vacío no hace ninguna eliminación. **-destruyeDipolo()**: Destruye el Dipolo. **-vacíoDipolo()**: Regresa Verdadero si el Dipolo está vacío. **-conIzqDipolo(), conDerDipolo()**: Devuelven el elemento actual a la izquierda y a la derecha del Dipolo, respectivamente. Si el Dipolo está vacío, devuelve un valor especial que lo indica.
2	**Declaraciones:** TipoEle: e, {TipoEleNoDef}	
3	**Especificación semántica:** vacíoDipolo(creaDipolo())=Verdadero vacíoDipolo(insIzqDipolo(creaDipolo(),e))=F vacíoDipolo(insDerDipolo(creaDipolo(),e))=F conIzqDipolo(creaDipolo())={TipoEleNoDef} conDerDipolo(creaDipolo())={TipoEleNoDef} eliIzqDipolo(creaDipolo())=creaDipolo() eliDerDipolo(creaDipolo())=creaDipolo()	

Lista

- La lista es el tipo más general de estructura lineal donde las inserciones y eliminaciones se hacen en cualquier punto de la lista, por ello se debe especificar donde se requiere que se haga la operación.

- Sus operaciones básicas son: creación, destrucción, inserción, eliminación, consulta y verificación de lista vacía. a la derecha.

- La especificación de este TAD se muestra en la siguiente figura.

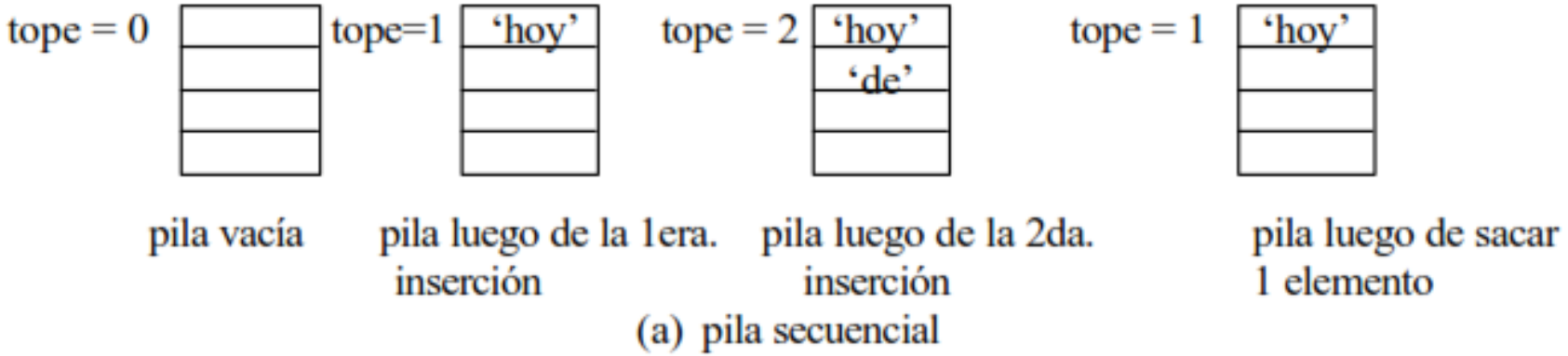
Lista

Especificación Lista[TipoEle]		
1	**Especificación sintáctica:** creaLista()⟶ Lista, insLista(Lista,TipoEle)⟶ Lista, eliLista(Lista)⟶ Lista, conLista(Lista)⟶ TipoEle, vacíaLista(Lista)⟶ Lógico, destruyeLista(Lista)⟶ .	**-creaLista()**: Crea una lista vacía. **-insLista()**: Ingresa un nuevo elemento a la lista según una posición especificada. **-eliLista()**: Elimina el elemento en la lista según una posición especificada. Si la lista está vacía no hace ninguna eliminación. **-destruyeLista()**: Destruye la lista. **-vacíaLista()**: Regresa Verdadero si la lista está vacía. **-conLista()**: Devuelve el elemento en la lista seguún una posición especificada. Si la lista está vacía devuelve un valor especial que lo indica.
2	**Declaraciones:** TipoEle: e, {TipoEleNoDef}	
3	**Especificación semántica:** vacíaLista(creaLista())=Verdadero vacíaLista(insLista(creaLista(),e))=Falso conLista(creaLista())={TipoEleNoDef} eliLista(creaLista())=creaLista()	

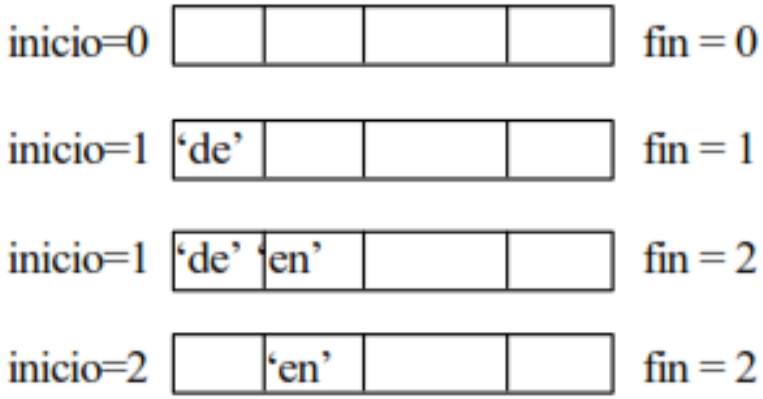
Estructuras de almacenamiento

- Para almacenar en memoria una lista se utilizan dos métodos: el secuencial y el enlazado.
- En el primero los elementos de la lista están contiguos físicamente en la memoria y para su soporte se utiliza el tipo Arreglo, teniendo la ventaja de efectuar los accesos a los elementos en forma directa, solamente indicando el subíndice o posición del elemento en la estructura.
- Por ello, el acceso es más rápido pero con la desventaja de restricción del crecimiento de la lista que está sujeta al tamaño máximo del arreglo.
- Así también, esto conlleva a otra desventaja que es la pérdida de espacio usado cuando la lista no ocupa más de la mitad del espacio supuesto para la lista.
- Ambas desventajas tienen su origen en que los arreglos se declaran o crean con un tamaño máximo fijo.
- La representación física de las listas según este método está ilustrada en la siguiente figura.

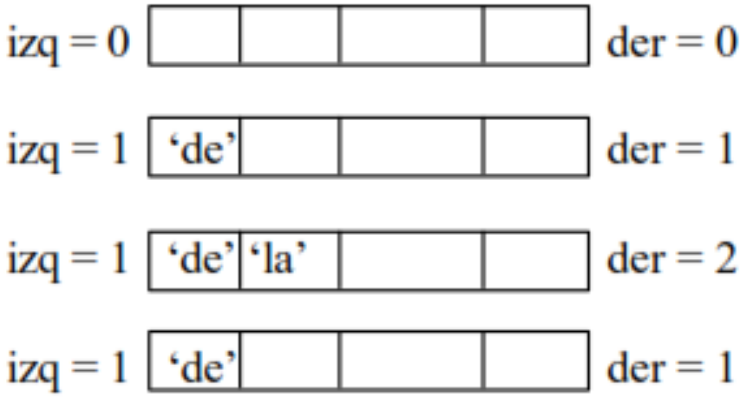
Estructuras de almacenamiento



(a) pila secuencial



(b) cola secuencial



(c) dipolo secuencial

GRACIAS