Algoritmos y Estructuras de Datos I

Universidad Católica del Uruguay

Listas, Pilas y Colas

Listas

- Secuencia de cero o más elementos de un tipo determinado
- Sucesión de elementos separados por comas:

  $a_1, a_2, ..., a_n$ donde n >= 0 y a_i es del tipo de elemento
- n = 0 significa lista vacía
- a_1 es el primer elemento y a_n es el último
- a_i está en la posición i
- a_{i-1} precede a a_i , y a_i sucede a a_{i-1}
- los elementos pueden estar ordenados en forma lineal de acuerdo a sus posiciones en la lista

Listas – operaciones (1)

- EsVacia. Devuelve TRUE si la lista tiene 0 elementos, y FALSE en caso contrario.
- ObtenerPrimero: Tipo Elemento - devuelve el primer elemento de una lista; si la lista está vacía, devuelve NULO.
- ObtenerUltimo: Tipo Elemento - devuelve el último elemento de una lista; si la lista está vacía, devuelve NULO.
- InsertarAlPrincipio (unElemento). Inserta unElemento al principio de la lista
- InsertarAlFinal (unElemento). Inserta unElemento al final de la lista
- BuscarEtiqueta(unaEtiqueta) Busca el TElemento que tenga la etiqueta y si lo encuentra, retorna el elemento; de lo contrario retorna NULO. Si hay más de un elemento con esa etiqueta, devuelve el primero que encuentra.
- BuscarElemento (unElemento). Retorna si unElemento existe o no en la lista
- Eliminar (unElemento). Elimina el elemento indicado de la lista, devolviendo TRUE si la eliminación fue exitosa o FALSE en caso contrario.
- Eliminar (unaEtiqueta). Elimina el elemento que tiene esa etiqueta, devolviendo el elemento si la eliminación fue exitosa o NULO en caso contrario. Si hay más de un elemento con esa etiqueta, elimina el primero que encuentra.
- QuitarPrimero. Quita y devuelve el primer elemento de la lista. Si la lista está vacía, devuelve NULO.

Listas – operaciones (2)

- Siguiente(unElemento). Devuleve el elemento que está a continuación de unElemento; si unElemento es el último de la lista, devuelve NULO.
- Anterior(unElemento). Devuleve el elemento que está inmediatamente antes de unElemento; si unElemento es el primero de la lista, devuelve NULO.
- ObtenerCantidadElementos. Devuelve la cantidad de elementos que tiene la lista.
- Vaciar. Deja la lista vacía
- ImprimirEtiquetas. Imprime las etiquetas de los elemento, una a continuación de la otra, del primero al último elemento.
- Invertir. Devuelve una nueva Lista invertida.
- Concatenar(otraLista). Pone la otraLista a continuación de la lista.
- EliminarDuplicados. Si hay elememmto con la misma etiqueta, deja solamente el primero que aparece.

Listas – operaciones (3)

Aplicables a Lista Ordenada

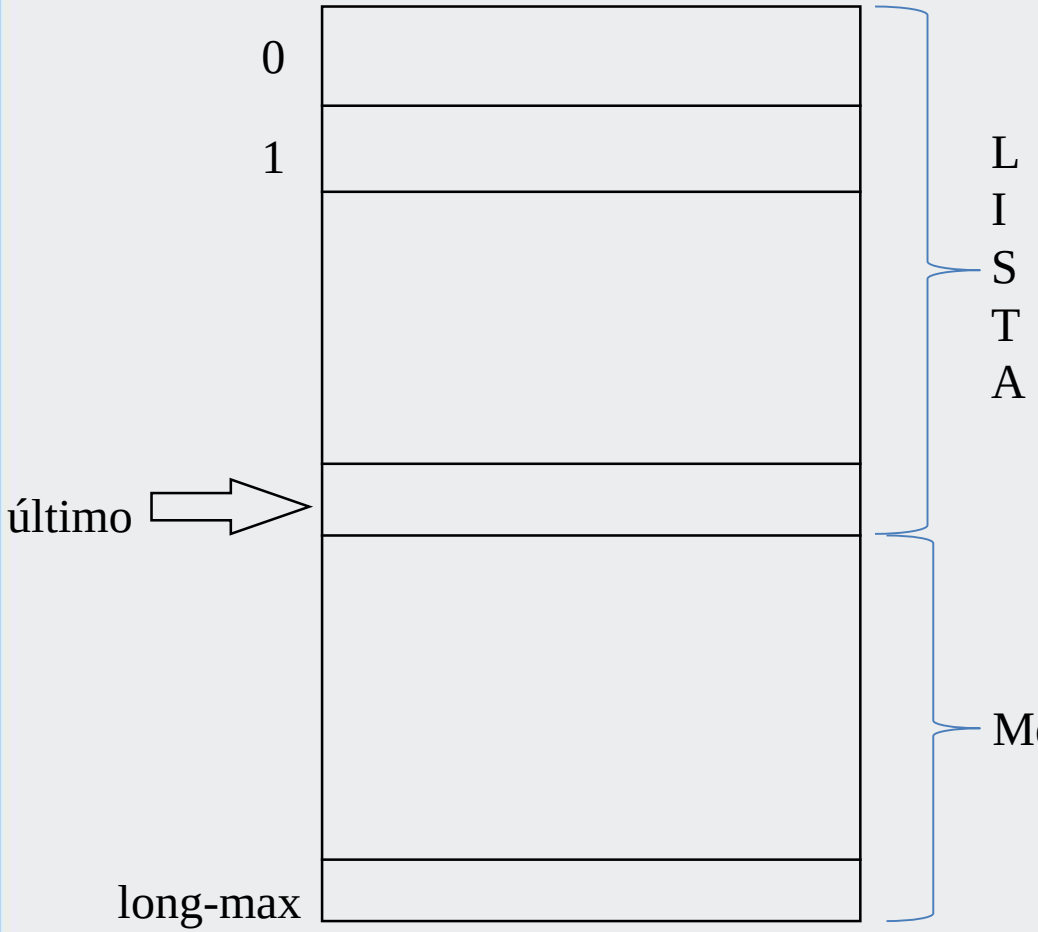
- Mezclar (otraLista). Devuelve una nueva lista mezclada (intercalada por el valor de la etiqueta de cada elemento) con la otraLista.
- InsertarOrdenado (unElemento). Inserta el elemento en el lugar que le corresponde de acuerdo al valor de su etiqueta. Si ya hay elementos con esa etiqueta, se inserta a continuación de ellas.

TDA: Tipo de datos abstracto

Universidad Católica del Uruguay

- Un modelo matemático con un conjunto de operaciones definidas sobre él se llama Tipo de datos abstracto, o TDA.
- De esta forma, definida la lista y su conjunto de operaciones, se puede hablar del TDA Lista.
- Un mismo TDA puede implementarse (programarse) con diferentes estructuras de datos.
- La implementación (programación) de las operaciones dependerá de las estructuras de datos elegidas para implementarlo.

Implementación del TDA Lista con arreglo




```
TDA Lista
        elementos : array [0..longMax]
                                de  TipoElemento;
        último : integer;

TDA TipoElemento
                etiqueta : TipoEtiqueta;
```

Implementación con arreglo

- Emplea memoria contigua.
- Se reserva memoria que eventualmente no se usa.
- Puede requerir redimensionamiento en tiempo de ejecución.
- Requiere desplazamientos de elementos al insertar o eliminar (excepto si es al final).
- Se puede acceder a una posición determinada en forma directa.

Implementación del TDA Lista con referencias

- TDA TipoElemento

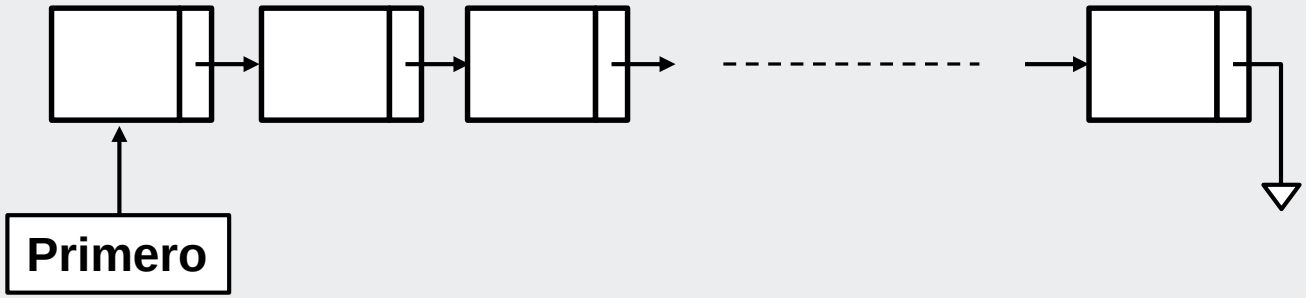
  etiqueta : TipoEtiqueta;

  siguiente : TipoElemento;

- TDA Lista

  Primero: TipoElemento;

Implementación del TDA Lista con referencias



Tipo Elemento y Tipo Etiqueta

- Tipo Elemento
  - Atributos:
    - Etiqueta: Tipo Etiqueta
    - Siguiente : TipoElemento
  - Operaciones
    - ImprimirEtiqueta
- Tipo Etiqueta :
  - Atributos
    - Etiqueta
  - Operaciones:
    - Imprimir

Implementación con referencias

- Evita el empleo de memoria contigua.
- Evita los desplazamientos de elementos al insertar o eliminar.
- Precio adicional : espacio requerido por las referencias.
- La lista está formada por nodos o celdas; cada celda contiene un elemento de la lista y una referencia a la siguiente celda.
- La última celda contiene una referencia nula o vacía
- Existe una celda de encabezamiento que apunta a la primer celda de la lista, y no tiene ningún elemento.
- Opcionalmente podríamos tener una referencia al último nodo o celda- resulta útil

Listas: Comparación de métodos.

- La realización con arreglos exige especificar el tamaño máximo de la lista: se desperdicia espacio de almacenamiento.
- Ciertas realizaciones son más lentas en una representación que en otra.
- La realización con referencias utiliza para los elementos sólo el espacio real requerido por ellos, pero debe agregarse el espacio necesario para cada referencia.

Listas doblemente encadenadas

- En algunos casos es necesario recorrer la lista en ambos sentidos.
- Utilizamos entonces dos punteros: uno a la celda siguiente y otro a la anterior.
- Algunas operaciones son más complicadas.
- Espacio adicional para el nuevo puntero.

Listas: Planteo de un problema 1

- Se necesita imprimir la etiqueta de cada elemento de una lista en el orden en el que aparecen del primero al último elemento.
- Primer paso: elaborar una solución en lenguaje natural sobre el modelo matemático lista. Escribir la solución.
- Segundo paso: refinar la solución del primer paso y elaborar un algoritmo en seudocódigo sobre el TDA LISTA.
- Tercer paso: implantar el TDA con estructuras de datos y procedimientos en un lenguaje de programación.

Resolución del Problema: PASO 1

- Sobre la base del modelo matemático LISTA (secuencia de elementos), una solución puede ser la siguiente:
- Voy al primer elemento, y si existe, imprimo su etiqueta.
- Avanzo hasta el segundo elemento repitiendo la operación.
- Sigo avanzando e imprimiendo hasta que llego al último elemento.

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

- Lista.ImprimirEtiquetas

  //imprime las etiquetas de todos los elemento, del primero al último.

  //se define la operación TEtiqueta.Imprimir que imprime el contenido de una etiqueta.

  UnElemento de Tipo Elemento

  COMIENZO

  UnElemento ← UnaLista.Primero

  Mientras UnElemento <> nulo hacer

  UnElemento.Etiqueta.Imprimir

  UnElemento← UnElemento.Siguiente

  Fin mientras

  FIN

Seudocódigo

- PRECONDICIONES
  - **Estado** en que ***debe*** encontrarse el objeto ***antes*** de comenzar el algoritmo
  - Lenguaje natural
  - Especificación rigurosa
  - Mapeo con el código!!!
  - Ayudan a escribir los casos de prueba

- POSTCONDICIONES
  - **Estado** en que ***ha de quedar*** el objeto ***después*** de terminar el algoritmo
  - **Facilitan** la escritura de los casos de prueba.

LISTAS: Planteo de un problema 2

- Se necesita diseñar un algoritmo que elimine los elementos con etiqueta duplicada que puedan existir en una lista.
- Primer paso: elaborar una solución en lenguaje natural sobre el modelo matemático lista. Escribir la solución.
- Segundo paso: refinar la solución del primer paso y elaborar un algoritmo en seudocódigo sobre el TDA LISTA.
- Tercer paso: implantar el TDA con estructuras de datos y procedimientos en un lenguaje de programación.

Resolución del Problema: PASO 1

- Sobre la base del modelo matemático LISTA (secuencia de elementos), una solución puede ser la siguiente:
- Veo cual es el primer elemento y me voy fijando a partir del siguiente elemento y hasta el último a ver si hay alguno que tenga la misma etiqueta; en caso afirmativo elimino ese duplicado.
- Avanzo hasta el segundo elemento repitiendo la operación.
- Sigo avanzando y verificando hasta que llego al último elemento. (Comparamos cada uno con todos los demás)

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

Universidad Católica del Uruguay

- UnaLista.EliminaDuplicados

```
//elimina los elementos con etiqueta duplicada de la lista
UnElemento, ElOtroElemento de Tipo Elemento

COMIENZO
UnElemento ← UnaLista.Primero
Mientras UnElemento <> nulo hacer
      ElOtroElemento ← UnElemento.Siguiente
     mientras ElOtroElemento <> nulo hacer
        si ElOtroElemento.Etiqueta = UnElemento.Etiqueta
                        entonces  UnaLista.Elimina(ElOtroElemento)
                        si no ElOtroElemento ← ElOtroElemento.Siguiente
        fin si
     fin mientras
     UnElemento← UnElemento.Siguiente
Fin mientras
FIN
```

TDA y LISTAS EN LA WEB

Universidad Católica del Uruguay

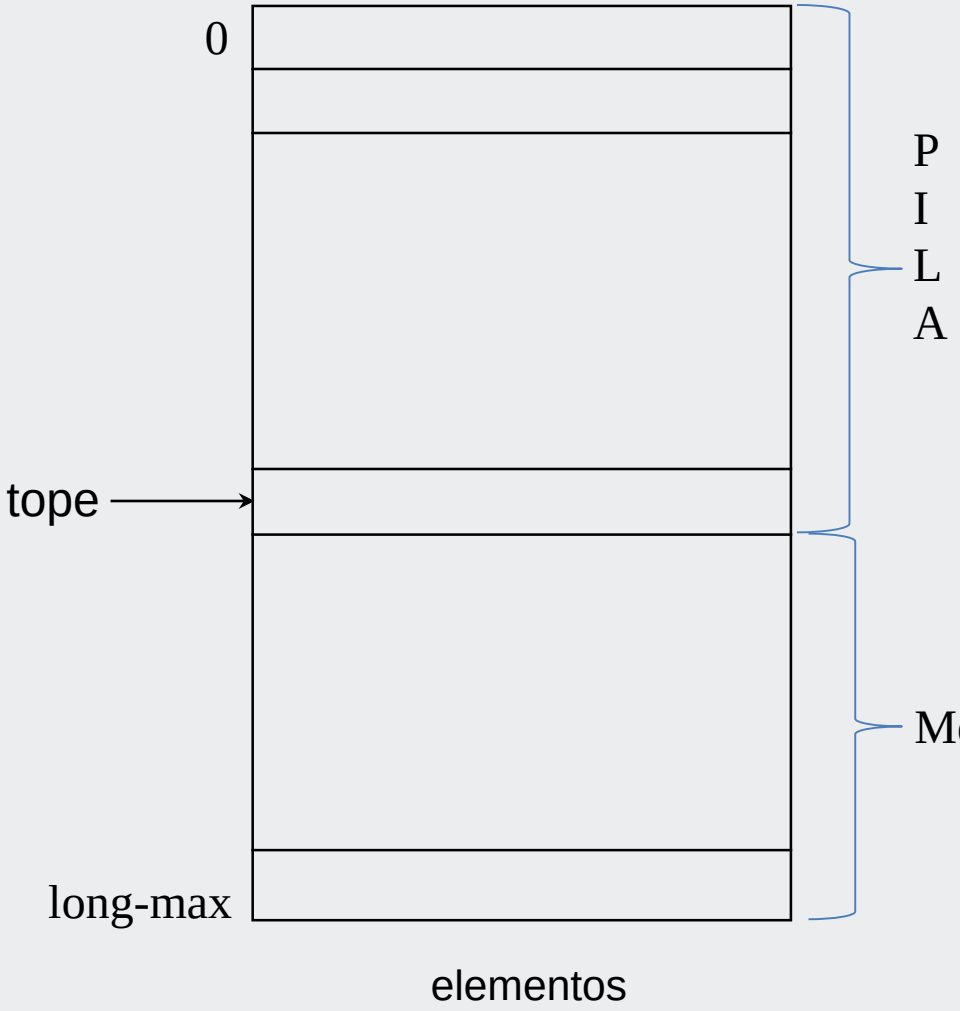
- http://en.wikipedia.org/wiki/Abstract_data_type#Defining_an_ADT
- http://www.answers.com/topic/abstract-data-type#Typical_operations
- http://es.wikipedia.org/wiki/Tipo_de_dato_abstracto

- http://en.wikipedia.org/wiki/Linked_list
- http://es.wikipedia.org/wiki/Lista_(estructura_de_datos)
- http://en.wikipedia.org/wiki/List_(computing)

Pilas

- Tipo especial de lista en que todas las inserciones y eliminaciones se hacen en el mismo extremo, denominado tope.
- Listas LIFO (Last In First Out).
- Operaciones normales para el TDA Pila:
  - Anula
  - Tope
  - Saca – “pop”
  - Mete(unElemento) – “push”
  - Vacía

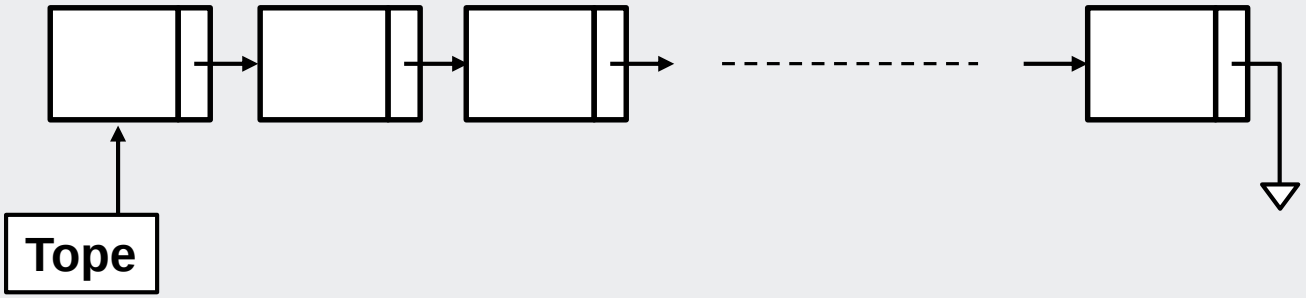
Realización de pila con arreglo




```
TDA   Pila
        tope integer;
        elementos  array [0..longMax]
                      de tipoElemento;
```

Realización de pila con referencias

- Se puede reusar la lista.
- ¿El tope conviene que sea el primero o el último de la lista?



Colas

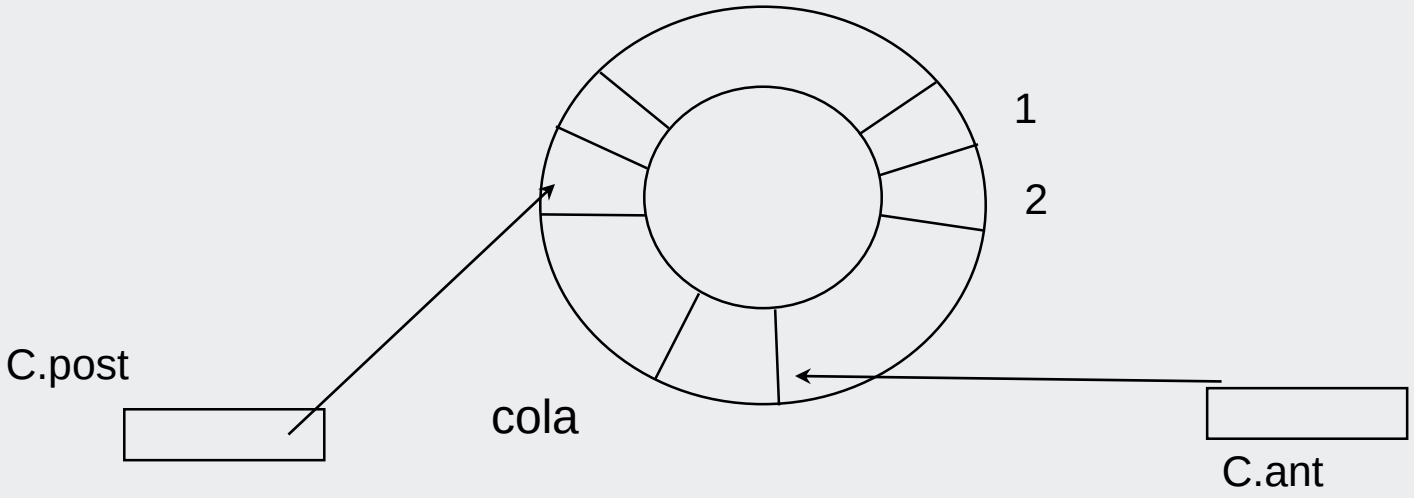
- Tipo especial de lista en el que los elementos se insertan en un extremo (**el posterior**) y se extraen por el otro (**el anterior o frente**).
- Listas ***FIFO*** (First In First Out).
- Operaciones:
  - Anula
  - Frente
  - PoneEnCola(unElemento)
  - QuitaDeCola
  - Vacía

Realización de colas con referencias

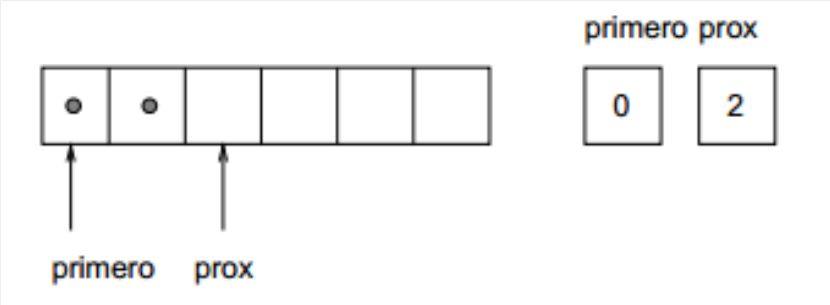
- Igual que en las pilas, cualquier operación de listas es válida para las colas.
- Se puede mantener una referencia al principio y otra al final de la cola.
- Se puede utilizar como encabezamiento una celda adicional en la cual se ignora el campo elemento.

Implementación de colas con arreglos “circulares”

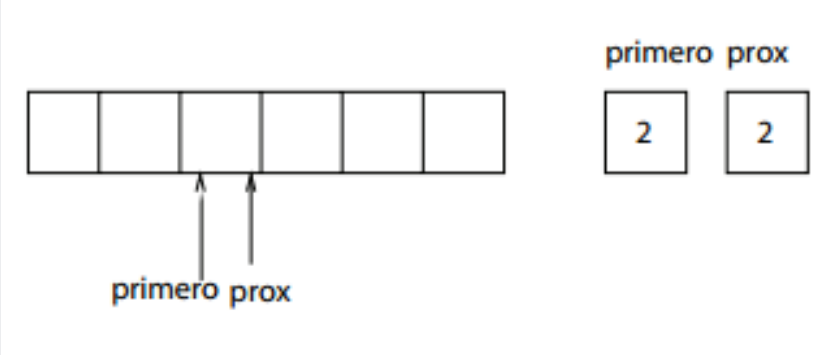
- La representación de listas por medio de arreglos puede también usarse para las colas, pero no es muy eficiente.
- Para mejorar, utilizamos un arreglo como si fuera un círculo, es decir, después del último elemento viene nuevamente el primero.
- La cola se encuentra en alguna parte del círculo, utilizando posiciones consecutivas.



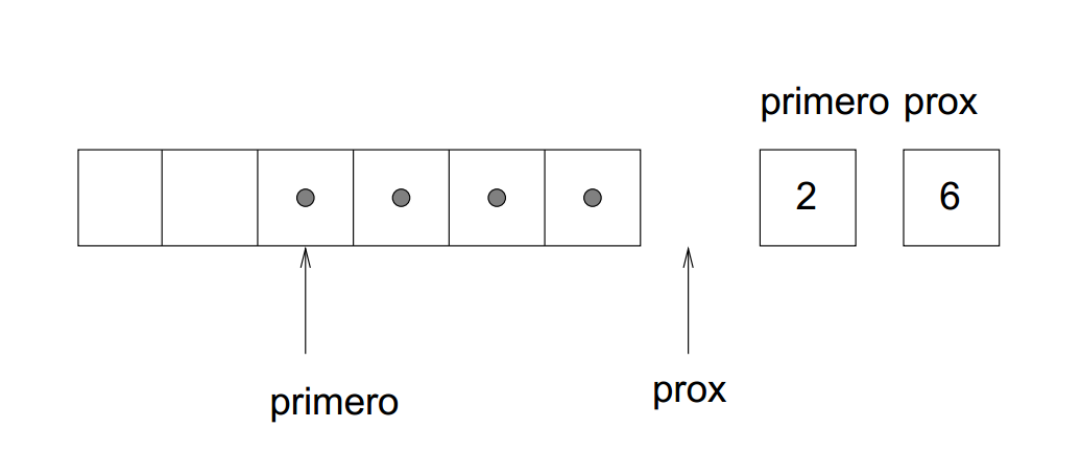
Implementación de colas con arreglos “circulares”



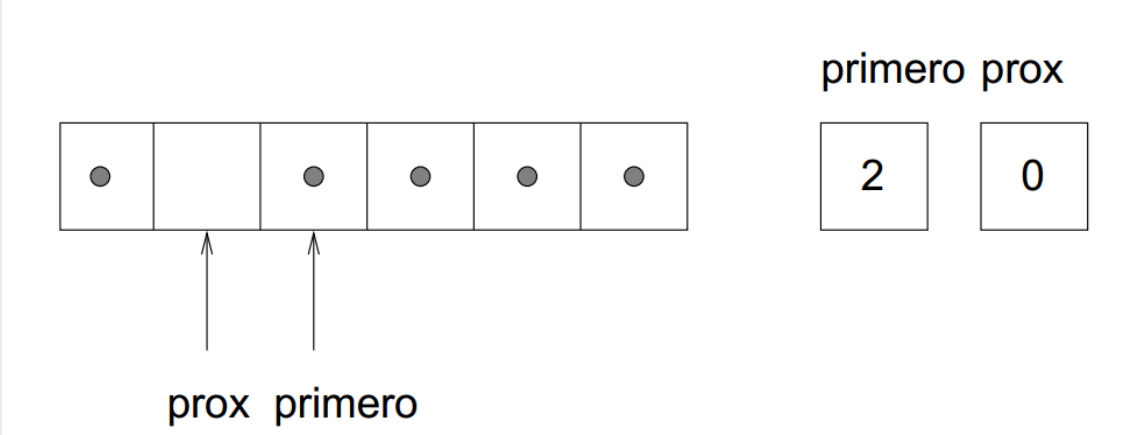
- estaVacía() ?
- insertar (unElemento)
- quitar



Implementación de colas con arreglos “circulares”



No confundir con “cola llena”…



Otros comportamientos interesantes de las listas

Universidad Católica del Uruguay

- Invertir
- Ordenar
- Mezclar dos listas

- Estos comportamientos deben implementarse en el código JAVA del TDA

LISTAS: Planteo de un problema 3

- Se necesita Ordenar una lista.
  - A partir de una lista de entrada, obtener una segunda lista en la cual los elementos de la primera estarán ordenados de menor a mayor (de acuerdo a sus claves)
- Primer paso: elaborar una solución en lenguaje natural sobre el modelo matemático lista. Escribir la solución. Indicar Precondiciones, Postcondiciones y todas las posibles situaciones internas.
- Segundo paso: refinar la solución del primer paso y elaborar un algoritmo en seudocódigo sobre el TDA LISTA.
- Tercer paso: implantar el TDA con estructuras de datos y procedimientos en un lenguaje de programación.

Resolución del Problema: PASO 1

- Sobre la base del modelo matemático LISTA (secuencia de elementos), una solución puede ser la siguiente – en lenguaje natural - :
- Si la lista origen no está vacía
  - Para cada elemento de la lista de origen
    - Extraerlo de la lista origen
    - Insertarlo en la lista de salida, en forma ordenada
  - Devolver la lista de salida
- Precondiciones?
- Postcondiciones?
- Situaciones posibles en el algoritmo?

Resolución del Problema: PASO 2

De tipo TLista **Ordenar**

COM

Si Vacia () entonces salir //

de tipo TLista **Lista2**

tempNodo ← QuitarPrimero () // analizar qué debe cumplir este método

Mientras tempNodo <> nulo

Lista2.**InsertarOrdenado** (tempNodo)

tempNodo ← QuitarPrimero ()

finMientras

devolver Lista2

FIN

Resolución del Problema: PASO 2(2)

De tipo TLista **InsertarOrdenado(**de tipoNodo unNodo);

COM

```
Si Vacia () entonces
        primero ← unNodo                    // será el primero
salir
FinSi
Si unNodo.clave < primero.clave             // va al principio
    unoNodo.Siguiente ← primero
    primero ← unNodo
    salir
 Fin Si
tempNodo ← primero
Mientras tempNodo.Siguiente() <> nulo
  Si tempNodo.Siguiente.Clave > unNodo.clave
        unNodo.Siguiente ← tempNodo.Siguiente
        tempNodo.Siguente ← unNodo
        salir
  finSi
  tempNodo ←  tempNodo.Siguiente
 finMientras
 tempNodo.Siguiente ← unNodo
```

FIN

Ejercicios, seudocódigo y práctico en papel

- Escriba un algoritmo para intercalar dos listas ordenadas.
- Se define el palíndromo de una palabra a la palabra obtenida invirtiendo todas las letras; por ejemplo el palíndromo de “arroz” es “zorra”. Escriba un algoritmo no recursivo que dada una palabra, genere su palíndromo. El algoritmo debe ser escrito sobre la base de un TDA PALABRA, con sus primitivas correspondientes.
- Escriba un procedimiento para intercambiar dos elementos consecutivos cualesquiera de una lista, en la cual denotaremos como elemento al primero de ellos.
- Escriba un algoritmo para invertir una lista simple.
- Desarrolle los algoritmos para implementar las operaciones de Unión e Intersección sobre el TDA LISTA, utilizado para representar un CONJUNTO. Las listas de entrada se encuentran ordenadas