Algoritmos y Estructuras de Datos

Listas, Pilas y Colas

1

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

2

Ejemplos

- Se trata de un modelo abstracto genérico de amplio uso en la vida cotidiana
- Una lista de productos de un supermercado
- Una lista de alumnos de una universidad
- Una lista de facturas a pagar
- Una lista de renglones de una factura
- Una lista de páginas de un documento
- Una lista de egresados de UCU
- Etc.

3

Listas – funcionalidades básicas

- Agregar un elemento "E" a una lista; la lista permite agregar un elemento en cualquier lugar:
  - Al final, el nuevo elemento será ahora el último de la lista
  - Al principio, el nuevo elemento será ahora el primero de la lista
  - En cualquier posición o de acuerdo a algún criterio (ejemplo: en una posición "x", en orden alfabético …)
- Eliminar un elemento "E" de una lista:
  - Sea que conozcamos al elemento "E", o alguna característica del mismo (por ejemplo el número de la cédula de identidad)
  - Sea que queramos eliminar el elemento de cierta posición "x" (el primero, el último, el cuarto,…)
- Acceder a cierto elemento "E" para realizar alguna acción sobre el mismo
- Otras básicas:
  - saber si está vacía
  - saber cuántos elementos tiene
  - vaciarla

4

Listas – métodos a desarrollar

Una vez implementadas las funcionalidades básicas, a partir de ellas se pueden construir algoritmos más complejos, por ejemplo:

- Eliminar eventuales elementos duplicados de una lista
- Concatenar una lista con otra
- Por ejemplo, si las listas implementan conjuntos matemáticos, las operaciones sobre conjuntos (la lista de alumnos inscriptos a Sistemas Digitales "o" a Fundamentos de Gestión, pero no a ambas)
- Etcétera

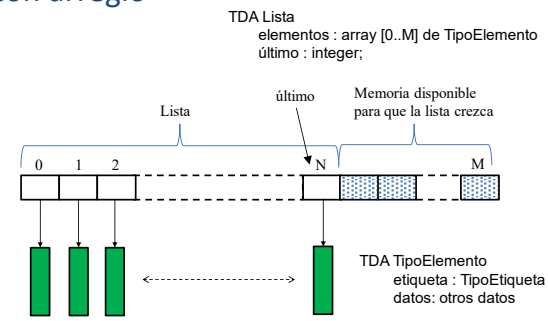
5

TDA: Tipo de datos abstracto

- 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.

6

Implementación del TDA Lista con arreglo



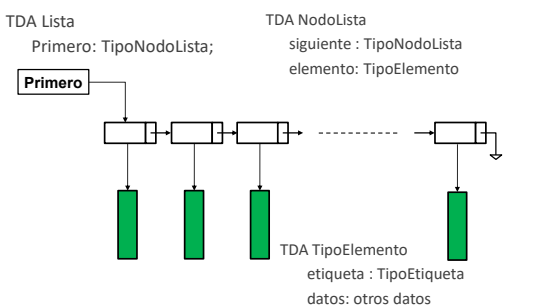
7

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.

8

Implementación ("mínima") del TDA Lista con referencias



9

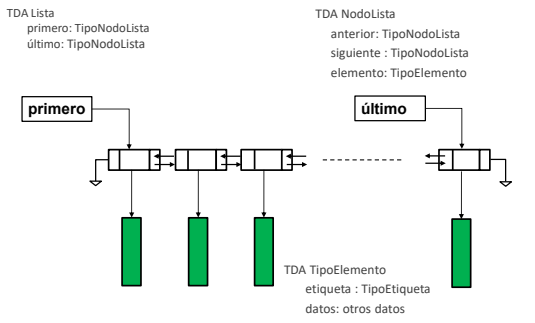
Implementación con referencias

- La lista está formada por nodos o celdas; cada nodo contiene un elemento de la lista y una referencia al siguiente nodo.
- El último nodo contiene una referencia nula.
- Evita el empleo de memoria contigua.
- Evita los desplazamientos de elementos al insertar o eliminar.
- Precio adicional : espacio requerido por las referencias.
- 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 puede estar desperdiciando 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.

Implementación del TDA Lista con referencias doblemente encadenada



Implementación del TDA Lista con referencias

UCU

- TDA Lista
  - Atributos:
    - Primero: TipoNodoLista;
  - Operaciones:
    - Inserta (E de TipoElemento, pos de TipoNodoLista): boolean
    - Elimina (pos de TipoNodoLista): TipoElemento
    - Busca (x de TipoEtiqueta): TipoNodoLista
- TDA NodoLista
  - Atributos:
    - siguiente : TipoNodoLista;
    - elemento: TipoElemento
    - etiqueta: TipoEtiqueta
  - Operaciones:
    - imprimirEtiqueta
- TDA Etiqueta
  - Atributos:
    - Etiqueta
  - Operaciones:
    - Imprimir
- TDA Elemento
  - Atributos:
    - Etiqueta: TipoEtiqueta
    - Datos: TipoDatos

13

Listas: Planteo de un problema 1

UCU

- 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 métodos de un lenguaje de programación.

14

Resolución del Problema: PASO 1

UCU

- 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.

15

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

UCU

- Lista.ImprimirEtiquetas
  //imprime las etiquetas de todos los elementos, del primero al último.

```
nodoActual de TipoNodoLista
COMIENZO
nodoActual ← UnaLista.Primero
Mientras nodoActual <> nulo hacer
    nodoActual.Etiqueta.Imprimir
    nodoActual ← nodoActual.Siguiente
Fin mientras
FIN
```


Seudocódigo

UCU

- 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

UCU

- 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)

19

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

- UnaLista.EliminaDuplicados

```
//elimina los elementos con etiqueta duplicada de la lista
nodoActual, otroNodo de Tipo Elemento

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


20

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

- UnaLista.EliminaDuplicados

```
//elimina los elementos con etiqueta duplicada de la lista
nodoActual, otroNodo de Tipo Elemento

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

¿Cómo sería la condición si el nodo no tuviera la etiqueta del elemento?

21

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

UCU

- UnaLista.EliminaDuplicados
  //elimina los elementos con etiqueta duplicada de la lista
  nodoActual, otroNodo de Tipo Elemento

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

¿Cómo se comporta "elimina"?
- **Implementar**
- **Implementar con lista doblemente encadenada**

LISTAS: Algoritmos de ejemplo con seudocódigo TDA

UCU

- Implementar las operaciones básicas:
  - Inserta (E de TipoElemento, pos de TipoNodoLista): Boolean // Si "pos"
    - es nulo, inserta al final
    - es primero, inserta al principio
    - sino inserta, el elemento como siguiente de "pos"
    - devuelve verdadero si la inserción fue exitosa
  - Elimina (pos de TipoNodoLista): TipoElemento
    - luego de la eliminación, "pos" será el que era su siguiente
    - devuelve el elemento apuntado por "pos"
  - Busca (x de TipoEtiqueta): TipoNodoLista
    - devuelve la posición del elemento con etiqueta "x"

TDA y LISTAS EN LA WEB

UCU

- 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)

ORDENAR UNA LISTA

UCU

- Podemos distinguir 3 métodos de ordenación o clasificación (sort) sencillos e intuitivos, llamados "directos", para ordenar una secuencia de elementos:
  - Intercambio: recorrer la lista "n" veces intercambiando los elementos contiguos que no respeten el orden.
  - Selección: en cada paso obtener el elemento de menor clave e insertarlo en el lugar que le corresponde en el área de salida.
  - Inserción: en cada paso quitar el siguiente elemento del área de entrada y colocarlo en el lugar que le corresponde en el área de salida.

ORDENAR UNA LISTA ENLAZADA

UCU

- El método de intercambio está especialmente desaconsejado para ordenar una lista enlazada.
- Los métodos directos que pueden usarse son:
  - Selección
  - Inserción

ORDENAR UNA LISTA ENLAZADA

UCU

- **Selección directa**
  - Crear una nueva lista vacía, sea "ordenada"
  - Hasta que la lista original quede vacía
    - Obtener el elemento de mayor (menor) clave
    - Quitarlo de la lista
    - Insertarlo en la primera (última) posición de la lista ordenada
  - Devolver la lista ordenada

ORDENAR UNA LISTA ENLAZADA

UCU

- **Inserción directa**
  - Crear una nueva lista vacía, sea "ordenada"
  - Hasta que la lista original quede vacía
    - Obtener el primer elemento
    - Quitarlo de la lista
    - Insertarlo en la posición que le corresponde de la lista ordenada
  - Devolver la lista ordenada

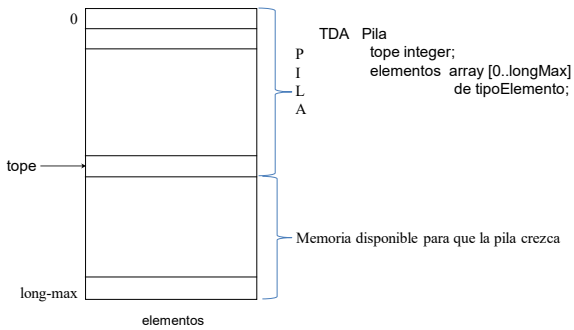
Pilas

UCU

- 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

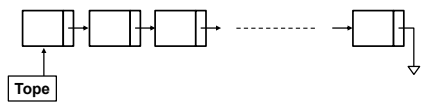
UCU



Realización de pila con referencias

UCU

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



31

Colas

UCU

- 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

32

Realización de colas con referencias

UCU

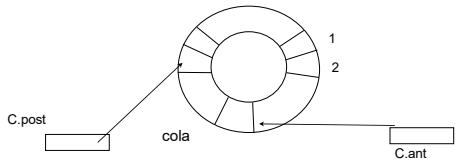
- Igual que en las pilas, cualquier operación de listas es válida para las colas.
- Se puede mantener una referencia al principio y otra al final de la cola.
- Se puede utilizar como encabezamiento una celda adicional en la cual se ignora el campo elemento.

33

Implementación de colas con arreglos “circulares”

UCU

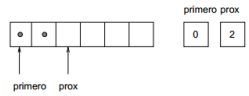
- La representación de listas por medio de arreglos puede también usarse para las colas, pero no es muy eficiente.
- Para mejorar, utilizamos un arreglo como si fuera un círculo, es decir, después del último elemento viene nuevamente el primero.
- La cola se encuentra en alguna parte del círculo, utilizando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



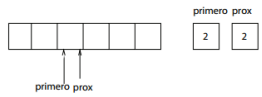
34

Implementación de colas con arreglos “circulares”

UCU



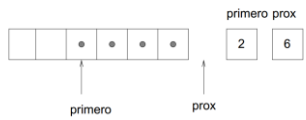
- estaVacía() ?
- insertar (unElemento)
- quitar



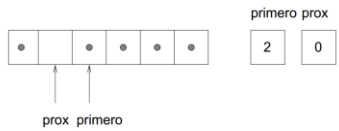
35

Implementación de colas con arreglos “circulares”

UCU



No confundir con “cola llena”…



36

Otros comportamientos interesantes de las listas

UCU

- Invertir
- Ordenar
- Mezclar dos listas

- Estos comportamientos deben implementarse en el código JAVA del TDA

37

LISTAS: Planteo de un problema 3

UCU

- 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.

38

Resolución del Problema: PASO 1

UCU

- 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?

39

Resolución del Problema: PASO 2

UCU

De tipo TLista **Ordenar**
COM
  Lista2 ← nuevaLista
  Mientras no (vacia()) hacer
    elementoActual ← Eliminar (primero)
    Lista2.**InsertarOrdenado** (elementoActual)
  finMientras
  devolver Lista2
FIN

Resolución del Problema: PASO 2

UCU

De tipo TLista **Ordenar**
COM
  Lista2 ← nuevaLista
  Mientras no (vacia()) hacer
    elementoActual ← Eliminar (primero)
    Lista2.**InsertarOrdenado** (elementoActual)
  finMientras
  devolver Lista2
FIN

¿Cómo se comporta "insertarOrdenado"? Implementar

Resolución del Problema: PASO 2(2)

UCU

```
De tipo TLista InsertarOrdenado(de tipoElemento elElemento);
COM
    unNodo ← nuevoNodo (elElemento)
    Si Vacia () entonces
        primero ← unNodo                    // será el primero
        salir
    FinSi
    Si unNodo.clave < primero.clave         // va al principio
        unNodo.Siguiente ← primero
        primero ← unNodo
        salir
     Fin Si
    tempNodo ← primero
    Mientras (tempNodo.Siguiente() <> nulo Y tempNodo.Siguiente.Clave > unNodo.clave) hacer
        tempNodo ← tempNodo.Siguiente
    finMientras
    unNodo ← tempNodo.siguiente
    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