TDA LISTA

Ing. Mario Milton López Winnipeg

TDA LISTA

- Descripción del TDA Lista.
- Especificación del TDA Lista.
- Aplicaciones con Lista.
- Implementaciones del TDA Lista.
  - Vectores.
  - Nodos enlazados Simulación
  - Nodos enlazados Punteros

2.1 Descripción del TDA Lista

Una lista es una colección de elementos ordenada de acuerdo a las direcciones de éstos (*secuencia*, relación *predecesor-sucesor*)

***primer* elemento**

$$L=<a_1, a_2,...,a_n>$$

***último* elemento**

a_i Î L, $i=1,...,n$ (n es la *longitud* de la lista)

$n=0$ Þ lista *vacía*

Caracterización importante: los elementos pueden insertarse o eliminarse en cualquier direccion de una lista

2.1 Descripción del TDA Lista

Operaciones de construcción

- **CREA**

Operaciones de direccionamiento

- **FIN**
- **PRIMERO**
- **SIGUIENTE**
- **ANTERIOR**

Operaciones de consulta

- **VACIA**
- **RECUPERA**
- **LONGITUD**

Operaciones de modificación

- **INSERTA**
- **SUPRIME**
- **MODIFICA**

2.2 Especificación del TDA Lista

Lista = TDA con operaciones crea, fin, primero, siguiente, anterior, vacia, recupera, longitud, inserta, suprime y modifica.

DESCRIPCION

Los valores del TDA Lista son listas de elementos del tipo Elemento. Las direcciones de los elementos de la lista y la direccion fin de la lista son del tipo Direccion. Las listas son *mutables*: inserta, suprime y modifica añaden, eliminan y modifican elementos en la lista respectivamente.

2.2 Especificación del TDA Lista

OPERACIONES

crea() devuelve (L:Lista)

efecto: Devuelve la lista vacía L.

fin(L:Lista) devuelve (Direccion)

efecto: Devuelve la Direccion fin de la lista L.

primero(L:Lista) devuelve (Direccion)

requerimientos: La lista L es no vacía.

efecto: Devuelve la Direccion del primer elemento de la lista L.

2.2 Especificación del TDA Lista

siguiente(L:Lista; P:Direccion) devuelve (Direccion)

requerimientos: La lista L es no vacía. La direccion P es la direccion de un elemento de lista L.

efecto: Devuelve la direccion que ocupa el elemento sucesor del elemento que ocupa la direccion P en la lista L. Si P es la direccion que ocupa el último elemento de lista L, devuelve la direccion fin de la lista.

anterior(L:Lista; P:Direccion) devuelve (Direccion)

requerimientos: La lista L es no vacía. La direccion P es la direccion de un elemento de lista L distinto del primero, o bien la direccion fin de la lista L.

efecto: Devuelve la direccion que ocupa el elemento predecesor del elemento que ocupa la direccion P en la lista L. Si P es la direccion fin de lista L, devuelve la direccion del último elemento de la lista.

2.2 Especificación del TDA Lista

vacia(L:Lista) devuelve (booleano)

efecto: Devuelve cierto si L es la lista vacía, y falso en caso contrario.

recupera(L:Lista; P:Direccion) devuelve (E:Elemento)

requerimientos: La lista L es no vacía. La direccion P es la direccion de un elemento de la lista L.

efecto: Devuelve en E el elemento que ocupa la direccion P en la lista L.

longitud(L:Lista) devuelve (entero)

efecto: Devuelve la longitud de la lista L.

2.2 Especificación del TDA Lista

inserta(L:Lista; P:direccion; E:Elemento)

requerimientos: La direccion P es la direccion de un elemento de lista L, o bien la direccion fin de la lista.

modifica: L.

efecto: Inserta el elemento E en la lista L como predecesor del elemento que ocupa la direccion P en la lista. Si P es la direccion fin de la lista L entonces el elemento E pasa a ser el penúltimo elemento de la lista tras la operación de inserción. El valor de P, así como el de cualquier otro caso o instancia del tipo de datos direccion existente antes de la operación de inserción, quedan indefinidos tras ejecutarse la operación.

suprime(L:Lista; P:direccion)

requerimientos: La lista L es no vacía. La direccion P es la direccion de un elemento de lista L.

modifica: L.

efecto: Elimina de la lista L el elemento que ocupa la direccion P. El valor de P, así como el de cualquier otro caso o instancia del tipo de datos direccion existente antes de la operación de eliminación, quedan indefinidos tras ejecutarse la operación.

2.2 Especificación del TDA Lista

modifica(L:lista; P:direccion; E:tipoelem)

requerimientos: La lista L es no vacía. La direccion P es la direccion de un elemento de lista L.

modifica: L.

efecto: Modifica el elemento que ocupa la direccion P de la lista L, cambiándolo por el nuevo elemento E.

2.2 Especificación del TDA Lista

```
INTERFACE Lista {
   Publico Direccion fin();
   publico Direccion primero();
   publico Direccion siguiente( direccion)
   publico Direccion anterior( direccion)
   publico boolean vacia();
   publico TipoElemento recupera( direccion)
   publico entero longitud();
   publico void inserta( direccion, elemento);
   publico void suprime( direccion)
   publico void modifica( direccion, elemento)
} // fin interface Lista
```

2.3 Ejemplos de Uso

Buscar un elemento en una lista

Escribir los elementos de una lista

Ordenar una lista

2.3 Ejemplos de Uso

```
publico Direccion buscar( Lista L, TipoElemento elemento)
    inicio
        si (L.vacia())
        entonces retornar nulo;
        caso contrario
         inicio
            p= L.primero()
            mientras p<> Nulo
            inicio
               e=L.recupera(p)
               si e= elemento entoces retornar p
               p = l.siguiente(p)
            fin
          fin
        returnar nulo;
    fin
```

2.3 Ejemplos de Uso

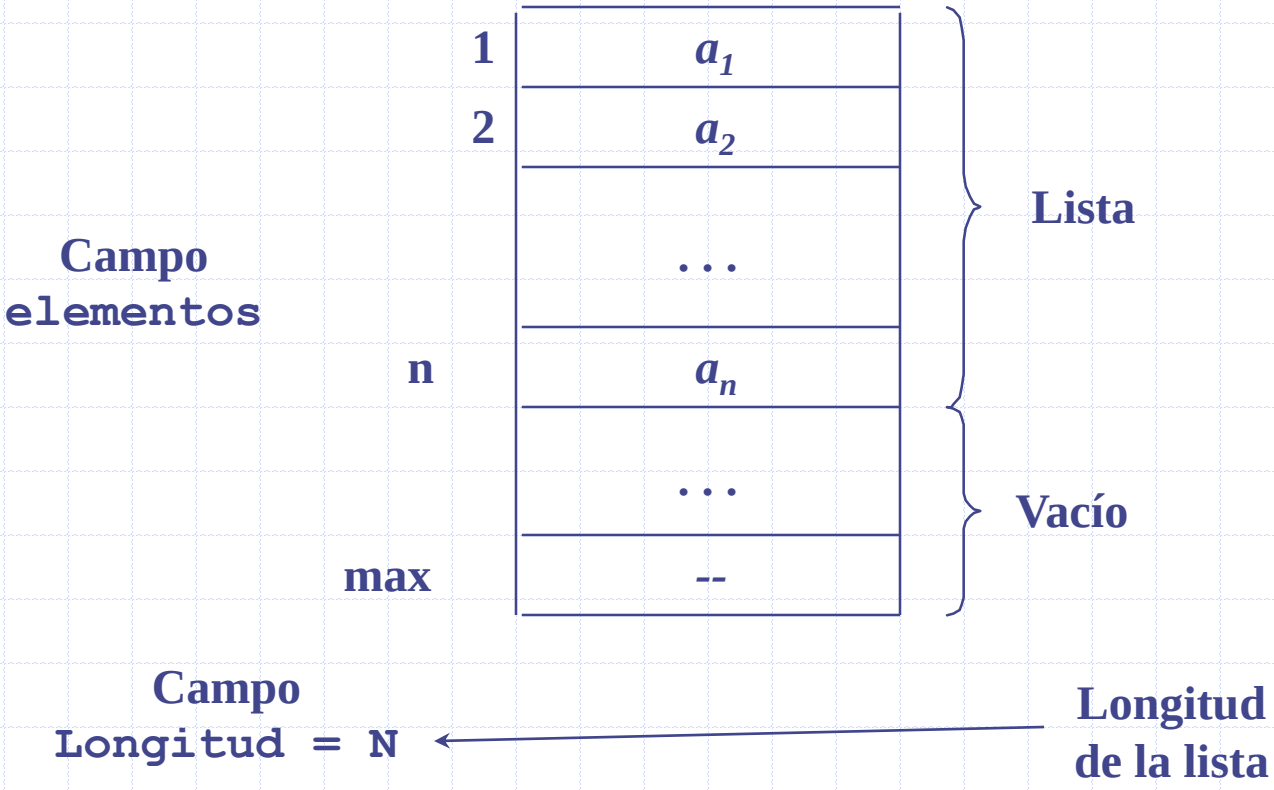
```
publico void imprimir(Lista L)
  inicio
     Si (L.vacia()=Verdadero)
          entoces retornar // termina proceso
          caso contrario
              inicio
               p=L.primero()
               mientras p<>nulo
                   inicio
                     e= L.recupera(p)
                     mostrar ( e ,'   ')
                     p = L.siguiente(p)) {
                   fin
               fin
    fin
```

2.4 Implementaciones del TDA Lista

En esta sección se presentan algunas implementaciones para el TDA Lista. En concreto, se utilizarán dos tipos de representaciones:

- **Vectores**
- **Nodos Enlazados simulacion**
- **Nodos enlazados punteros**

2.4.1 Representaciones vectores



2.4.1 Representaciones vectores

Definición básica de la clase Lista con representación contigua:

```
Constante max = 100
Tipo de Datos
Direccion de tipo Entero

Clase Lista
 Atributos
    elementos[Max] vector de tipo TipoElemento
    longitud de tipo Entero
 Metodos
        ……..
```

2.4.1 Representaciones vectores

```
publico Lista.Crear()
inicio
    longitud = 0;
fin

publico Direccion Lista.fin()
inicio
    si vacia() entoces // llamar a exception listavacia
                caso contrario retornar longitud;
fin

publico Direccion Lista.primero()
 inicio
    si no vacia() entoces
        retornar 1 // Retorna 1 por que en esa direccion
                   // esta el primer elemento
        caso contrario
                   // llamar a excepción ListaVacia
 fin
```

2.4.1 Representaciones vectores

```
publico direccion Lista.siguiente( p direccion)
  inicio
   si vacia() entoces // llamar a exception ListaVacia
     caso contrario
     inicio
       si p = longitud entoces // llamar exception DireccionErr
                          caso contraio
                                  retornar (p +1)
    fin
  fin
publico direccion Lista.anterior( p direccion)
  inicio
   si vacia() entoces // llamar a exception ListaVacia
     caso contrario
     inicio
       si p = 1 entoces // llamar exception DireccionPrimeraErr
                  caso contraio
                                  retornar (p -1)
    fin
  fin
```

2.4.1 Representaciones vectores

```
publico booleano Lista.vacia()
inicio
    retornar (longitud = 0)
fin

publico TipoElemento Lista.recupera( p direccion)
  inicio
   si vacia() entoces // llamar a exception ListaVacia
              caso contrario
                 inicio
                    si no (p>=1 y p <=longitud )
                      entoces
                        // llamar a exception DireccionErr
                      caso contrario
                        retornar elementos[ p ]
                 fin
  fin
```

2.4.1 Representaciones vectores

```
Publico entero lista.longitud()
Inicio
    retornar longitud
fin
Publico lista.inserta( p Direccion, elemento TipoElemento)
Inicio
    si longitud = max
            entoces // llamar a exception listallena
    si no (p >=1 y p<=longitud)
            entoces // llamar a exception DireccionErr
    para cada i = (longitud +1)  descontando hasta (p + 1)
     inicio
         elementos[i]=elementos[i-1]
     fin
    elementos [p] = elemento
    lontitud = longitud + 1
 Fin
```

2.4.1 Representaciones vectores

```
Publico lista.suprime( p Direccion)
Inicio
    si longitud = 0
        entoces // llamar a exception listaVacia
    si no (p >=1 y p<=longitud)
        entoces // llamar a exception direccionErr
   para cada i = p  hasta (longitud - 1)
     inicio
         elementos[i]=elementos[i + 1]
     fin
    lontitud = longitud - 1
 Fin
```

2.4.1 Representaciones vectores

```
Publico lista.modifica( p Direccion, elemento
   tipoelemento)
Inicio
    si longitud = 0
        entoces // llamar a exception listaVacia
    si no (p >=1 y p<=longitud)
        entoces // llamar a exception direccionErr
    elementos[p]=elemento
Fin
```

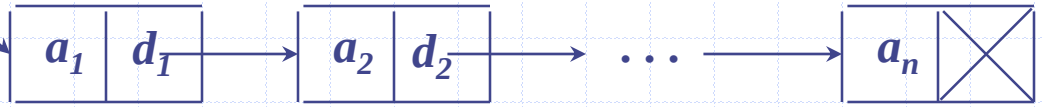
2.4.2 Representaciones enlazadas

- **Los elementos de la lista se almacenan en memoria mediante objetos enlazados entre sí.**
- **Para cada elemento de la lista se crea un objeto (*nodo*) el cual contiene dos tipos de información: el elemento propiamente dicho y un apuntador al nodo que contiene el siguiente elemento de la lista.**
- **Describiremos la representacion con Simple Enlace**

2.4.2.1 Representación con simple enlace

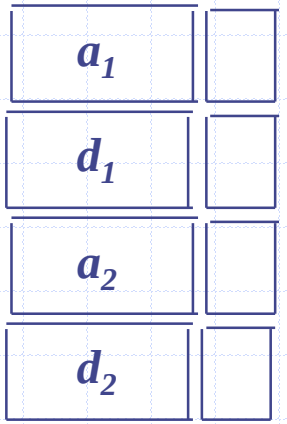
Campo L
Contine direccion primer Nodo

L= DireccionMemoria



Longitud = n

Simulacion de memoria queda asi..



2.4.2.1 Representación con simple enlace

Definición básica de la clase Lista con representación enlazada con simple enlace:

```
Tipo de dato
 Nodo
     elemento TipoElemento
     sig      Puntero a Nodo
 // fin definicion

 Direccion Puntero a espacio de memoria de tipo Nodo

Clase Lista
  Atributos
      PtrElementos  Puntero de tipo Direccion
     longitud de tipo Entero
  Metodos
        ..............
```

2.4.2.1 Representación con simple enlace

```
publico Lista.Crear()
    inicio
        longitud = 0
        ptrElementos=nulo
    fin
publico Direccion Lista.fin()
    inicio
        si vacia() entoces // llamar a exception listavacia
                    caso contrario
                    inicio
                      x = PtrElementos
                      mientras x<> nulo
                        Inicio
                           PtrFin= x
                           x = m.obtener_dato(x,2)
                        fin
                      retornar PtrFin
                    fin
    fin
```

2.4.2.1 Representación con simple enlace

```
publico Direccion Lista.primero()
      inicio
         si no vacia() entoces
          retornar PtrElementos // Por que en esa direccion
                                // esta el primer elemento
          caso contrario
                          // llamar a excepción ListaVacia
      fin
publico direccion Lista.siguiente( p direccion)
  inicio
   si vacia() entoces // llamar a exception ListaVacia
      caso contrario
      inicio
        si p = fin() entoces // llamar exception DireccionErr
                        caso contraio
                               retornar (m.obtener_dato(p,2))
     fin
  fin
```

2.4.2.1 Representación con simple enlace

```
publico direccion Lista.anterior( p direccion)
 inicio
  si vacia() entoces // llamar a exception ListaVacia
    caso contrario
    inicio
    si p = primero()entoces //llamar exception DireccionPrimeraErr
          caso contraio
          INICIO
          x= PtrElementos
          ant= nulo
          mientras x<>Nulo
           inicio
               si x=p entonces retornar ant
               ant = x
               x = m.obtener_dato(x,2)
             fin
          FIN
    fin
 fin
```

2.4.2.1 Representación con simple enlace

```
 publico booleano Lista.vacia()
  inicio
      retornar (longitud = 0) // O (ptrelementos = nulo)
  fin

publico TipoElemento Lista.recupera( p direccion)
  inicio
   si vacia() entoces // llamar a exception ListaVacia
              caso contrario retornar m.obener_dato(p,1)
  fin
Publico entero lista.longitud()
Inicio
    retornar longitud
fin
```

2.4.2.1 Representación con simple enlace

```
Publico lista.inserta( p Direccion, E TipoElemento)
Inicio // x tendria direcion de memoria si existe espacio
 X=M.New_espacio(2)
 si x <> nulo entoces
  inicio
     m.poner_dato(x,1,E) , m.poner_dato(x,2,Nulo)
     si  vacia() entoces  PtrElementos= x  , longitud = 1
                caso contrario  inicio
                longitude=longitude +1
                si P=primero() entoces m.poner_dato(x,2,p)
                                       PtrElementos= x
                                caso contrario
                                       ant = anterior(p) ,
                                       m.poner_dato(ant,2,x)
                                       m.poner_dato(x,2,p)
                                 fin
  fin
    caso contrario // llamar a exception existeespaciomemoria
Fin
```

2.4.2.1 Representación con simple enlace

```
Publico lista.suprime( p Direccion)
Inicio
    si longitud = 0  entoces // llamar a exception listavacia
    si p=ptrelementos entonces // es el primer nodo
                        inicio
                          x=ptrelementos
                          ptrelementos=M.obtener_dato(ptrelementos,2)
                         // Liberar espacio de memoria x
                        fin
                       caso contrario
                        inicio
                            ant = anterior(p)
                            poner_dato(ant,2,siguiente(p))
                            // liberar espacio de memoria p
                       fin
    longitud=longitud -1
fin
```

2.4.2.1 Representación con simple enlace

```
Publico lista.modifica( p Direccion, elemento tipoelemento)
Inicio
    si vacia() entoces // llamar a exception listavacia
    m.poner_dato(p,1,element)
Fin
```

2.4.3 Representación punteros

Implementar la clase lista con punteros

Ejercicios

- Implementar los metodos pertenecientes a la clase Lista

publico direccion lista.localiza (E tipoElemento)

Retorna direccion de elemento E, caso contrario retorna nulo

publico lista.eliminaDato(E tipoelemento)

Elimina la ocurrencia E de la lista

publico lista.anula()

Esta operacion vacia la lista