UNIDAD II TDA Lista.

2.1.1 Descripción del TDA Lista.

En general, una lista es una secuencia de elementos de la forma: a_1, ... ,a_n, donde n>=0, y cada elemento a_i es de tipo genérico. El tamaño de la lista es n, pero si n=0, se dice que la lista es vacía. Para todas las listas, menos esta última, el primer elemento es a_1 y el último elemento es a_n. Se dice que ai+1 es sucesor de ai (i<n) y que ai-1 es predecesor de ai (i>1). Por lo tanto, los elementos pueden estar ordenados en función de su posición (i en el caso de a_i).

A diferencia de los conjuntos puede haber elementos repetidos en la lista, y a diferencia de las matrices y registros, el número de elementos de la lista no es fijo, es variable por lo tanto, no está limitado desde el principio.

En una máquina de información cada elemento suele denominarse nodo (celda o caja) que puede contener uno o más punteros.
Las listas admiten una serie de operaciones, que podemos agrupar en:

operaciones de construcción (crear),
operaciones de posicionamiento (fin, primero, siguiente, anterior),
operaciones de consulta (vacía, longitud, recupera) y finalmente
operaciones de modificación (modifica, suprime, inserta),

que se detallarán e implementarán más adelante.

Los TDA-lista se pueden realizar mediante memoria estática(vectores o arrays), o mediante asignación de memoria dinámica con punteros y mediante el uso de un modelo simulado de memoria dinámica con el TDA SMemoria.

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

Donde a_1 es el primer elemento y a_n es el último elemento, $\forall_{a_i} \in L$

2.1.2 Especificación del TDA Lista.
Partiendo de lo establecido en la Unidad 1. Especificacion Informal tenemos lo siguiente:
Elementos que Conforman la estructura

TDA Lista (**VALORES** Todos los Numeros Enteros , **OPERACIONES** crea, fin, primero, siguiente, anterior, vacia, recupera, longitud, inserta, inserta_primero, suprime y modifica)

DESCRIPCIÓN DE LAS OPERACIONES

crea (L: Lista)
Utilidad: Sirve para inicializar la Lista
Entrada: Lista L
Salida: Ninguna.
Precondición: Ninguna.
Poscondición: Lista L inicializada vacia

Fin (L: Lista) devuelve (Direccion)
Utilidad: Retorna la dirección donde se encuentra el último elemento de la lista L
Entrada: Lista L
Salida: Dirección
Precondición: Lista no vacía
Poscondición: Ninguna

primero(L: Lista) devuelve (Direccion)
Utilidad: Retorna la dirección donde se encuentra el primer elemento de la lista L
Entrada: Lista L
Salida: Dirección
Precondición: Lista no vacía
Poscondición: Ninguna

siguiente(L:Lista; P:Direccion) devuelve (Direccion)
Utilidad: Devuelve la direccion que ocupa el elemento sucesor del elemento que ocupa la direccion P en la lista L
Entrada: Lista L
Salida: Dirección
Precondición: Lista no vacía y P no es la dirección del ultimo elemento
Poscondición: Ninguna

anterior(L:Lista; P:Direccion) devuelve (Direccion)
Utilidad: Devuelve la direccion que ocupa el elemento predecesor del elemento que ocupa la direccion P en la lista L.
Entrada: Lista L
Salida: Dirección
Precondición: Lista no vacía y P no es la dirección del último elemento
Poscondición: Ninguna

vacia(L:Lista) devuelve (booleano)
Utilidad: Devuelve cierto si L es la lista vacía, y falso en caso contrario
Entrada: Lista L
Salida: Valor Booleano
Precondición: Ninguna.
Poscondición: Ninguna.

recupera(L: Lista; P:Direccion) devuelve (E: Elemento)
Utilidad: Devuelve en E el elemento que ocupa la dirección P en la lista L.
Entrada: Lista L
Salida: Elemento
Precondición: La lista L es no vacía. La dirección P es la dirección de un elemento de la lista L.
Poscondición: Ninguna.

longitud(L:Lista) devuelve (entero)
Utilidad: Devuelve la longitud de la lista L/ Cantidad de elementos
Entrada: Lista L
Salida: Numero
Precondición: Ninguna.
Poscondición: Ninguna.

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

Utilidad: Inserta el elemento E en la lista L como predecesor del elemento que ocupa la dirección P en la lista. Si P es la dirección 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 dirección existente antes de la operación de inserción, quedan indefinidos tras ejecutarse la operación

Entrada: Lista L

Salida: Ninguna

Precondición: La dirección P es la dirección de un elemento de lista L, o bien la dirección fin de la lista.

Poscondición: Lista L incrementada en un elemento mas E

Inserta_primero(L: Lista; E: Elemento)

Utilidad: Inserta el elemento E en la lista L como predecesor del elemento que ocupa el primer lugar en la lista..

Entrada: Lista L

Salida: Ninguna

Precondición: Ninguna

Poscondición: Lista L incrementada en un elemento más E

Inserta_ultimo (L: Lista; E: Elemento)

Utilidad: Inserta el elemento E en la lista L como último elemento de la misma, el elemento E ocupara el último lugar en la lista..

Entrada: Lista L

Salida: Ninguna

Precondición: Ninguna

Poscondición: Lista L incrementada en un elemento más E

suprime (L: Lista; P: direccion)

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

Entrada: Lista L

Salida: Ninguna

Precondición: La lista L es no vacía. La dirección P es la dirección de un elemento de lista L.

Poscondición: Lista L decrementa en un elemento más, el elemento E de la dirección P ya no pertenece a la lista L.

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

Utilidad: Modifica el elemento que ocupa la dirección P de la lista L, cambiándolo por nuevo elemento E

Entrada: Lista L

Salida: Ninguna

Precondición: La lista L es no vacía. La dirección P es la dirección de un elemento de lista L.

Poscondición: Lista L modificada.

2.1.3 Aplicaciones con Lista.

En esta sección se puede apreciar que ya estamos en condiciones para plantear algoritmos usando el TDA Lista, abstrayéndonos de la forma como esta implementado.

Ej1: Implementar un algoritmo que Busque un elemento en la lista L y retorne la Direccion donde se encuentra, caso contrario NULO

```
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 entonces retornar p
          p = l.siguiente(p)
         fin
       fin
     retornar nulo;
  fin
```

Ej2: Implementar un algoritmo que muestre el contenido de una Lista.

```
MostrarLista(Lista L)
  inicio
    Si (L.vacia()=Verdadero)
       entonces 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
```

2.1.4 Implementaciones del TDA Lista.

La implementación de los TDA puede variar sin embargo el Comportamiento del TDA lista no cambiara, por lo que se plantea en este apartado dos formas de implementación sin que esto signifique que son las únicas formas.

2.1.4.1 Implementación con Vector.

Para la implementación del TDA Lista se utilizara un Vector y un Atributo denominado Longitud , donde El vector será el que contendrá los elementos y Longitud guardara la cantidad de elementos contenidos en la lista, se hace notar que los índices que nos permiten manipular el Vector se constituirán en Direcciones para efectos de la implementación.

Definiendo la Clase

```
  Constante max = 100
                Nulo = 0
```

Tipo de Datos
Dirección de tipo Entero
Clase Lista
Atributos
elementos[Max] vector de tipo TipoElemento
longitud de tipo Entero
Métodos
Crear()
Dirección fin()
Dirección primero()
Dirección siguiente(dirección)
Dirección anterior(dirección)
booleano vacia()
TipoElemento recupera(dirección)
entero longitud ()
inserta(dirección, elemento)
inserta_primero(elemento)
inserta_ultimo(elemento)
suprime(dirección)
modifica(dirección, elemento)
Fin definición Clase

Implementación de la Clase Lista

Lista.Crear()
inicio
longitud = 0;
fin
https://youtu.be/laqhCv_i6yg

Dirección Lista.fin()
inicio
Si No vacia () entonces retornar longitud
caso contrario // llamar a error
fin
https://youtu.be/3lespxTivQw

Dirección Lista.primero()
inicio
si no vacia() entonces
retornar 1 // Retorna 1 por que en esa dirección esta el primero
caso contrario
// llamar a excepción ListaVacia
Fin
https://youtu.be/kZQqJaByJ2E

Direccion Lista.siguiente(p direccion)
inicio
si vacia() entonces // llamar a exception ListaVacia
caso contrario
inicio
si p = longitud entonces // llamar exception DireccionErr
caso contrario
retornar (p +1)
fin
fin

https://youtu.be/jiHBtivTXXs

Direccion Lista.anterior(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
    caso contrario
    inicio
      si p = 1 entonces // llamar exception DireccionPrimeraErr
              caso contrario
                        retornar (p -1)
  fin
 fin
```

https://youtu.be/hC3eaSendF8

booleano Lista.vacia()

```
 inicio
     retornar (longitud = 0)
 fin
```

https://youtu.be/F4aSSX3E7Z0

TipoElemento Lista.recupera(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
           caso contrario
             inicio
               si no (p>=1 y p <=longitud )
                 entonces
                  // llamar a exception DireccionErr
                 caso contrario
                   retornar elementos[ p ]
             fin
 fin
```

https://youtu.be/P_OIiDWhOws

entero lista.longitud()

```
Inicio
   retornar longitud
fin
```

https://youtu.be/FlSd2Vz6xxA

lista.inserta(p Direccion, elemento TipoElemento)

```
Inicio
   si longitud = max
          entonces // llamar a exception listallena
   si no (p >=1 y p<=longitud)
         entonces // llamar a exception DireccionErr
   para cada i = (longitud +1)  descontando hasta (p + 1)
    inicio
       elementos[i]=elementos[i-1]
    fin
   elementos [p] = elemento
   longitud = longitud + 1
 Fin
```

https://youtu.be/t9wq3bUKJDU

lista.inserta_primero(elemento TipoElemento)

```
Inicio
   si longitud = max
          entonces // llamar a exception listallena
  para cada i = (longitud +1) descontando hasta 2
                  inicio
                     elementos[i]=elementos[i-1]
                   fin
   elementos [1] = elemento
   longitud = longitud + 1
 Fin
```

https://youtu.be/uaje_VTv_pI

lista.inserta_ultimo(elemento TipoElemento)

```
Inicio
   si longitud = max
         entonces // llamar a exception listallena
        caso contrario
        longitud=longitud +1
        elementos [longitud ] = elemento
Fin
```

https://youtu.be/Im--6c08zhM

lista.suprime(p Direccion)

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

https://youtu.be/E5vUYO0KcsM

lista.modifica(p Direccion, elemento tipoelemento)

```
Inicio
   si longitud = 0
        entonces // llamar a exception listaVacia
   si no (p >=1 y p<=longitud)
        entonces // llamar a exception direccionErr
  elementos[p]=elemento
Fin
```

https://youtu.be/PHOSOjRB2WI

2.1.4.2 Implementación con Simulación de Memoria (usando la clase CSMemoria)

Esta forma de implementación es netamente académica en virtud a que lo que busca es una mejor comprensión sobre los punteros, para ello se entiende que se usara como Memoria nuestra clase CSmemoria implementada en la unidad uno.

Definiendo la clase Lista
Tipo de dato
Nodo
elemento TipoElemento
sig Puntero a Nodo(Valor entero para esta implementación)
fin
Direccion Puntero a Nodo (valor entero para esta implementación)

Clase Lista
Atributos
PtrElementos Direccion
longitud de tipo Entero
Métodos
Crear()
Dirección fin()
Dirección primero()
Dirección siguiente(dirección)
Dirección anterior(dirección)
booleano vacia()
TipoElemento recupera(dirección)
entero longitud ()
inserta(dirección, elemento)
inserta_primero(elemento)
inserta_ultimo(elemento)
suprime(dirección)
modifica(dirección, elemento)
Fin definición Clase

Implementación clase lista utilizando Simulador de Memoria CSmemoria.

Lista.Crear()

```
inicio
  longitud = 0    ptrElementos=nulo
fin
```

https://youtu.be/dc6ZIigxAic

Direccion Lista.fin()

```
inicio
  si vacia() entonces // llamar a exception listavacia
        caso contrario
        inicio
          x = PtrElementos
          mientras x<> nulo
              PtrFin= x
              x = m.obtener_dato(x,'->sig')
            fin mientras
          retornar PtrFin
        fin
fin
```

https://youtu.be/AKvP67BziG0

lista.inserta(p Direccion, E TipoElemento)

```
Inicio // x tendria direcion de memoria si existe espacio
 x=M.New_espacio('elemento,sig')
 si x <> nulo entonces
  inicio
    m.poner_dato(x,'->elemento',E), m.poner_dato(x,'->sig',Nulo)
    si  vacia() entonces  PtrElementos= x  , longitud = 1
            caso contrario  inicio
            longitude=longitude +1
            si P=primero() entonces m.poner_dato(x,'->sig',p)
                          PtrElementos= x
                    caso contrario
                       ant = anterior(p) ,
                       m.poner_dato(ant, '->sig',x)
                       m.poner_dato(x, '->sig',p)
                     fin
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

https://youtu.be/EKFoJfovTEg

lista.inserta_primero(E TipoElemento)

```
Inicio // x tendria direcion de memoria si existe espacio
 x=M.New_espacio('elemento,sig')
 si x <> nulo entonces
  inicio
    m.poner_dato(x,'->elemento',E)
    m.poner_dato(x,'->sig',PtrElementos)
    longitud=longitud   + 1
    PtrElementos = x
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

https://youtu.be/Qw2Z_roESVc

lista.inserta_ultimo(E TipoElemento)

```
Inicio // x tendria direcion de memoria si existe espacio
 x=M.New_espacio('elemento,sig')
 si x <> nulo entonces
  inicio
    m.poner_dato(x,'->elemento',E)
    m.poner_dato(x,'->sig',Null)
   si longitud <>0 entonces m.poner_dato( fin() ,'->sig',x)
                       caso contrario Ptrelementos=x
    longitud=longitud   + 1
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

https://youtu.be/jE4HAmeXwUE

Direccion Lista.primero()

```
    inicio
      si no vacia() entonces
       retornar PtrElementos  // apunta primer elemento
       caso contrario
                  // llamar a excepción ListaVacia
    fin
```

https://youtu.be/Qw2Z_roESVc

Direccion Lista.siguiente(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
    caso contrario
    inicio
      si p = fin() entonces // llamar exception DireccionErr
                caso contraio
                  retornar (m.obtener_dato(p,'->sig'))
   fin
 fin
```

https://youtu.be/f93XMxzWPBg

Direccion Lista.anterior(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
    caso contrario
    inicio
    si p = primero()entonces //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,'->sig')
           fin
         FIN
    fin
 fin
```

https://youtu.be/HqPemO8aw50

booleano Lista.vacia()

```
 inicio
    retornar (longitud = 0)
 fin
```

https://youtu.be/psd1FU-NmpQ

TipoElemento Lista.recupera(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
          caso contrario retornar m.obener_dato(p,'->Elemento')
 fin
```

https://youtu.be/kVqbsdI5c1U

entero lista.longitud()

```
Inicio
   retornar longitud
fin
```

https://youtu.be/rkpR-jh_M2o

lista.suprime(p Direccion)

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

https://youtu.be/rDP6HeGOBNM

lista.modifica(p Direccion, elemento tipoelemento)

```
Inicio
   si vacia() entonces // llamar a exception listavacia
                m.poner_dato(p,'->elemento',elemento)
Fin
```

https://youtu.be/3KBlRnlqTCg

2.1.4.2 Implementación con punteros.

En esta última forma de implementación planteada lo que se resalta son los cambios que tienen que hacerse al código de la implementación con el simulador de memoria considerando que ahora se esta trabajando con punteros, es asi que se tiene de color rojo los cambios fundamentales en los algoritmos ya vistos quedando por resolver las definiciones formales en c++

Definiendo la clase Lista

Tipo de dato

Nodo

elemento TipoElemento

sig Puntero a Nodo

fin

Direccion Puntero a Nodo

Clase Lista

Atributos

PtrElementos **Direccion**

longitud de tipo Entero

Métodos

Crear()

Dirección fin()

Dirección primero()

Dirección siguiente(dirección)

Dirección anterior(dirección)

booleano vacia()

TipoElemento recupera(dirección)

entero longitud ();

inserta(dirección, elemento)

inserta_primero(elemento)

inserta_ultimo(elemento)

suprime(dirección)

modifica(dirección, elemento)

Fin definición Clase

Implementación clase lista utilizando punteros.

Lista.Crear()

```
inicio
   longitud = 0     ptrElementos=NULO
fin
```

Direccion Lista.primero()

```
inicio
   si no vacia() entonces
    retornar PtrElementos // Por que en esa direccion
                          // esta el primer elemento
    caso contrario
                 // llamar a excepción ListaVacia
fin
```

Direccion Lista.fin()

```
  inicio
    si vacia() entonces // llamar a exception listavacia
          caso contrario
          inicio
           x = PtrElementos
           mientras x<> nulo
              PtrFin= x
              x = x→sig        // con simulador m.obtener_dato(x,'->sig')
            fin mientras
           retornar PtrFin
          fin
  fin
```

lista.inserta(p Direccion, E TipoElemento)

```
Inicio // x tendria dirección de memoria si existe espacio
 x= new Nodo // con simulador M.New_espacio('elemento,sig')
 si x <> nulo entonces
  inicio
    x→elemento = E  // con simulador m.poner_dato(x,'->elemento',E)
    x→sig = Nulo       // m.poner_dato(x,'->sig',Nulo)
    si  vacia() entonces  PtrElementos= x  , longitud = 1
          caso contrario  inicio
          longitude=longitude +1
          si P=primero() entonces  x→sig = p // con simulador m.poner_dato(x,'->sig',p)
                        PtrElementos= x
                caso contrario
                   ant = anterior(p) ,
                   ant→sig=x   //con simulador m.poner_dato(ant, '->sig',x)
                   x→sig=p      //con simulador m.poner_dato(x, '->sig',p)
                  fin
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

lista.inserta_primero(E TipoElemento)

```
Inicio // x tendria direcion de memoria si existe espacio
 x= New  Nodo
 si x <> nulo entonces
  inicio
    x→elemento=E            //   con simulador m.poner_dato(x,'->elemento',E)
    x→sig = PtrElementos   // con simulador m.poner_dato(x,'->sig',PtrElementos)
    longitud=longitud   + 1
    PtrElementos = x
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

booleano Lista.vacia()

```
 inicio
    retornar (longitud = 0) // O (ptrelementos = nulo)
 fin
```

lista.inserta_ultimo(E TipoElemento)

```
Inicio // x tendria direcion de memoria si existe espacio
 x= New Nodo              // con simulador New_espacio('elemento,sig')
 si x <> nulo entonces
  inicio
    x→elemento = E     // con simulador m.poner_dato(x,'->elemento',E)
    x→sig = Null          // con simulador  m.poner_dato(x,'->sig',Null)
   si longitud <>0 entonces
                          fin()→sig = x  // con simulador m.poner_dato( fin() ,'->sig',x)
                       caso contrario Ptrelementos=x
    longitud=longitud   + 1
  fin
   caso contrario // llamar a exception existeespaciomemoria
Fin
```

Direccion Lista.siguiente(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
    caso contrario
    inicio
     si p = fin() entonces // llamar exception DireccionErr
               caso contraio
                  retornar  p→sig  // con simulador (m.obtener_dato(p,'->sig'))
   fin
  fin
```

Direccion Lista.anterior(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
    caso contrario
    inicio
    si p = primero() entonces //llamar exception DireccionPrimeraErr
         caso contraio
         INICIO
         x= PtrElementos
         ant= nulo
         mientras x<>Nulo
          inicio
             si x=p entonces retornar ant
             ant = x
             x = x→sig    // con simulador m.obtener_dato(x,'->sig')
           fin
         FIN
    fin
  fin
```

TipoElemento Lista.recupera(p direccion)

```
 inicio
  si vacia() entonces // llamar a exception ListaVacia
        caso contrario retornar p→elemento  // con simulador m.obener_dato(p,'->Elemento')
 fin
```

entero lista.longitud()

```
Inicio
   retornar longitud
fin
```

lista.suprime(p Direccion)

```
Inicio
   si longitud = 0  entonces // llamar a exception listavacia
   si p=ptrelementos entonces // es el primer nodo
            inicio
             x=ptrelementos
             ptrelementos=ptrelementos→sig    // con simulador M.obtener_dato(ptrelementos, '->sig')
             delete x     //  con simulador m.delete_espacio(x)
            fin
            caso contrario
             inicio
               ant = anterior(p)
               ant→sig = siguiente(p)  // con simulador poner_dato(ant, '->sig',siguiente(p))
               delete p     //  con simulador m.delete_espacio(p)
             fin
   longitud=longitud -1
fin
```

lista.modifica(p Direccion, elemento tipoelemento)

```
Inicio
   si vacia() entonces // llamar a exception listavacia
                p→elemento = elemento   // con simulador m.poner_dato(p,'->elemento',elemento)
Fin
```