UNIVERSIDAD NACIONAL DE RÍO CUARTO
FACULTAD DE CS. EXACTAS, FCO-QCAS Y NATURALES - DEPARTAMENTO DE COMPUTACIÓN
ASIGNATURA: **INTRODUCCIÓN A LA ALGORÍTMICA Y PROGRAMACIÓN**
Año: 2022

Práctica Nº 9

Tema: Estructuras de datos dinámicas. Listas simplemente encadenadas (LSE). Listas doblemente encadenadas (LDE).
Duración: 6 clases
Esta práctica tiene como objetivos:

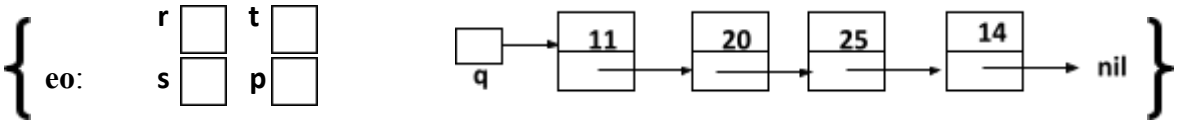
- Comprender la noción de puntero y su utilización para manipular estructuras de datos dinámicas.
- Resolver problemas que requieran la utilización de LSE y LDE.
- Analizar ventajas y desventajas de utilizar LSE y LDE, con o sin elemento ficticio.
- Diseñar algoritmos utilizando LSE y LDE y realizar pruebas de escritorio sencillas mediante estados (inicial, intermedios y final).
- Implementar en C algoritmos que contengan LSE y LDE y realizar pruebas sencillas.

CLASE 1 de 6

Ej. 1) Dada una lista simplemente encadenada (LSE) definida mediante TNodo, y las variables de tipo puntero **r**, **t**, **s** y **q**.

a) Describe gráficamente los estados intermedios y el estado final a partir del estado inicial y del segmento de algoritmo dados a continuación:

```
Algoritmo Ej1
Lexico
  TNodo= <info Є  entero, next Є  puntero a TNodo>
  q, r, t, s, p Є puntero a TNodo
Inicio
  ... {creación de la lista simplemente encadenada}
```




```
  r ← q
  r ← (^r).next
  t ← (^r).next
  Obtener(s)
  (^s).info ← 3
  (^s).next ← (^r).next
  (^r).next ← s
   p ← t
  (^t).info ← 29
   salida ((^t).info)   29

  ... {informar todos los componentes de la LSE}
```

Fin

b) Escribe en notación algorítmica la parte del algoritmo (creación de la LSE), necesario para alcanzar el estado e_o.

c) Escribe en notación algorítmica el final del algoritmo que nos permita informar todos los elementos de la LSE.

Ej.2) Describa gráficamente el estado inicial, los estados intermedios y el estado final del algoritmo del Ej 7 de la Teoría 13 (diapo 68).

Ej. 3)

a) Dado el siguiente léxico, construir una LSE para alojar una palabra de 20 letras, como máximo, y luego informar todas las vocales alojadas en cada campo info. La palabra es almacenada letra por letra en la LSE.
b) Modularizar el algoritmo anterior, desarrollando dos acciones, una acción para construir la LSE y otra para informar todas las vocales.

Léxico

TelemCar =< info ε Caracter, next ε puntero a TelemCar >
sec, aux ε puntero a TelemCar

Ej. 4) Desarrolla acciones que permitan manipular una **LSE de números enteros** de la siguiente manera:

a) Acción InsertarC, que permita insertar un elemento al comienzo de la lista.
b) Acción InsertarF, que permita insertar un elemento al final de la lista
c) Acción Inicializar, que permita crear la lista vacia y Acción VaciarL que elimine todos los elementos de la lista si no está vacía.
d) Acción InsertarPos, que permita insertar una elemento en una posición dada (si esta existe).
e) Acción EliminarC, que permita eliminar el primer elemento de la lista.
f) Acción EliminarPos, que permita eliminar un elemento de una posición dada, si esta existe.
g) Acción Mostrar, que permita informar los elementos de la lista.
h) Desarrollar un algoritmo que mediante un menú de opciones, permita acceder a cada una de las acciones antes enumeradas. Agregar una opción para terminar.

CLASE 2 de 6

Ej. 5) Dado el siguiente léxico:

Tpers= <nom ε cadena, ape ε cadena, dni ε Z, edad ε (0..99) >
Telem=<info ε Tpers, next ε puntero a Telem>
q ε puntero a Telem

Analizar el funcionamiento de las siguientes acciones y funciones y corregir los errores que se encuentren.

a)

```
Acción Crear( resultado list ε puntero a Telem)
Inicio
   list ←  nil
Facción
```

b)

```
Acción InsertarCab (dato list ε puntero a Telem, reg ε Tpers)
Lexico local
 aux  ε puntero a Telem
Inicio
  Obtener(aux)
  ( ^aux).info ←   reg
  ( ^aux).next ←  list
   list ←   aux
Facción
```

c)

```
Acción SuprimirCab (dato list ε puntero a Telem)
Lexico local
 aux  ε puntero a Telem
Inicio
 si no (vacia(list)) entonces
    aux ←   list
    list ←   (^list).next
    Liberar (aux)
  fsi
Facción
```

d)

```
Función Vacia (dato list ε puntero a Telem) --> Lógico
Inicio
  ←   (list = nil)
Ffunción
```

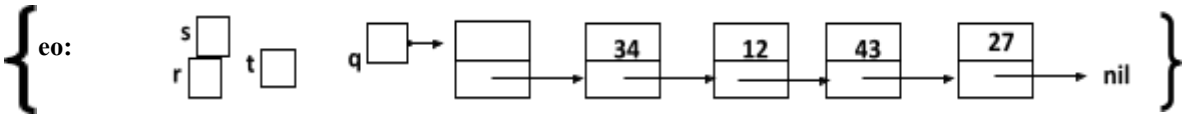
e)

```
Acción Listar (dato list ε puntero a Telem)
Lexico local
 aux  ε puntero a Telem
Inicio
  aux ←  list
  mientras aux <> nil hacer
      Salida: (^aux).info.nom
      Salida: (^aux).info.ape
      Salida: (^aux).info.edad
      aux ←   ( ^aux).next
  fmientras
Facción
```

Crear un algoritmo que, mediante un menú de opciones permita elegir cada una de las acciones antes definidas, para manipular una LSE de personas.

Ej. 6) Dada una lista simplemente encadenada con elemento ficticio, y las variables de tipo puntero **r**, **t**, **s** y **q**.

a) Describe gráficamente los estados intermedios y el estado final a partir del siguiente estado inicial, que se produce el siguiente segmento de algoritmo.




```
r ← (^q).next
t ←(^r).next
(^q).next ←t
(^r).next ← (^t).next
(^t).next ← r
r ← (^r).next
Obtener(s)
(^s).info ← 9
(^s).next ← (^r).next
(^r).next ← s
```

b) ¿Cómo se crea una lista con elemento ficticio?

CLASE 3 de 6

Ej. 7) Desarrolle acciones, utilizando una **LSE con elemento ficticio**, que permita manipular la lista de la siguiente manera:

a) Acción InsertarC, que permita insertar un elemento al comienzo de la secuencia.
b) Acción InsertarF, que permita insertar un elemento al final de la secuencia
c) Acción Inicializar, que permita crear la lista vacia y Acción VaciarL que elimine todos los elementos de la lista si no está vacía.
d) Acción InsertarPos, que permita insertar una elemento en una posición dada (si esta existe).
e) Acción EliminarPos, que permita eliminar un elemento de una posición dada, si esta existe.
f) Desarrolla una función que permita Buscar un empleado por el nombre. Esta función debe devolver un entero que representa la posición en la secuencia de la primera aparición de una persona con ese nombre. En caso de no existir, la función debe devolver –1.
g) Acción MuestraReg que permita informar los datos de un empleado. Al invocar esta acción debe previamente utilizar la función BUSCAR y en caso que el empleado exista, pasar como parámetro a MuestraReg la posición del empleado cuyas datos van a ser informados.
h) Acción Mostrar, que permita informar los elementos de la secuencia.

i) Desarrollar un algoritmo que mediante un menú de opciones, permita acceder a cada una de las acciones antes enumeradas. Agregar una opción para terminar.
j) Traduzca a C las acciones resultantes.

El tipo empleado se define como sigue:

En notación algorítmica
TEmpleado = <nombre ∈ Cadena, telefono ∈ Cadena, direccion ∈ Cadena, edad ∈ (18..65)>

CLASE 4 de 6

Ej. 8) Se necesita manipular una lista de empleados. Desarrolle los siguientes módulos de un algoritmo utilizando **lista simplemente encadenada (LSE)** con **elemento ficticio**, que permitan realizar lo siguiente:

El tipo empleado se define como sigue:

```
TEmpleado = <nombre ∈ Cadena, telefono ∈ Cadena, dirección ∈ Cadena, edad ∈ (18..65)>

TDobleteEmp = //completar definición de LSE
```

Los módulos deben ser autocontenidos. En cada caso deben definirse los parámetros formales adecuados y los tipos de pasaje de parámetros (dato, resultado o dato-resultado). Por ejemplo, la acción InsertarC podría tener el siguiente perfil:

Acción InsertarC (**dato** emp ∈ TEmpleado, **dato-resultado** listaEmpleados ∈ puntero a TDobleteEmp)

a) Acción Inicializar, que permita crear la lista vacía
b) Acción InsertarC, que permita insertar un elemento al comienzo de la lista.
c) Acción VaciarL que elimine todos los elementos de la lista si no está vacía.
d) Acción InsertarPos, que permita insertar un elemento en una posición dada (si esta existe). Esta acción está dada, puede verse a continuación:

Acción InsertarPos (**dato-resultado** m ε puntero a TDobleteEmp, **dato** pos ε Z, x ε TEmpleado)
Léxico local
aux ε puntero a TDobleteEmp
i ε Z
Inicio
i <-- 1
aux <-- m
mientras ((^aux).next ≠ nil y i ≠ pos) hacer
i <-- i + 1
aux <-- (^aux).next

```
fmientras
si i = pos entonces
   Obtener(s)
   (^s).info <-- x
   (^s).next <-- (^aux).next
   (^aux).next <-- s
sino
   msg <-- "posicion imposible"
  Salida: msg
fsi
faccion
```

e) Acción EliminarPos, que permita eliminar un elemento de una posición dada, si esta existe.

f) Desarrolla una función Buscar que permita BUSCAR un empleado por el nombre. Esta función debe devolver un entero que representa la posición en la lista de la primera aparición de una persona con ese nombre. En caso de no existir, la función debe devolver –1. Esta función está dada, puede verse a continuación:

```
Función Buscar (dato m ε puntero a TDobleteEmp, dato n ε Cadena) → Z
Léxico local
  aux ε puntero a TDobleteEmp
  i ε Z
Inicio
i <-- 0
aux <-- (^m).next
mientras (aux ≠ nil y (^aux).info.nombre ≠ n)) hacer
   i <-- i + 1
  aux <--(^aux).next
fmientras
si aux = nil  entonces
   <-- -1
sino
     <-- i
fsi
ffuncion
```

g) **Acción** MuestraReg (**dato** pos ε Z, m ε puntero a TDobleteEmp) que permita informar los datos de un empleado. Al invocar esta acción debe asegurarse que la posición es posible o existe.

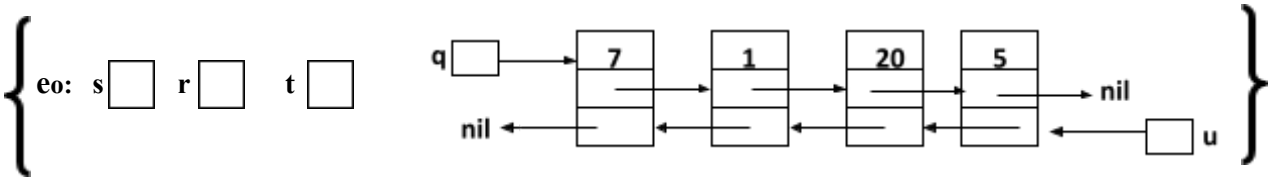
h) **Acción** Mostrar, que permita informar todos los elementos de la lista. Esta acción está dada, puede verse a continuación:

```
Acción Mostrar (dato m ε puntero a TDobleteEmp)
Léxico local
 pos ε Z
Inicio
  pos <-- 1
```

```
  aux ← (^m).next
  mientras aux ≠ nil) hacer
     MuestraReg(pos, m)
     pos <-- pos +1
    aux <-- (^aux).next
 fmientras
faccion
```

i) Desarrollar un algoritmo que mediante un menú de opciones, permita acceder a cada uno de los módulos (incisos a, b, c, d, e y h). Agregar una opción para finalizar.

Ej. 9) Dada una lista doblemente encadenada y las variables de tipo puntero **s**, **r**, **t** y **q**. Describe gráficamente los estados intermedios y el estado final partiendo del siguiente estado inicial y de acuerdo al siguiente segmento en notación algorítmica.




```
s ← q
s ← (^s).next
Obtener(r)
Obtener(t)
(^r).info ← 8
(^r).back ← s
(^r).next ← (^s).next
(^(^s).next).back ← r
(^s).next ← r
s ← (^s).next
(^t).info ← 12
(^t).back ← s
(^t).next ← (^s).next
(^(^s).next).back ← t
(^s).next ← t
```

CLASE 5 de 6

Ej. 10) Dada una **LDE de números enteros**, desarrolle acciones que permitan:

a) Insertar un entero, que se recibe como parámetro, en una posición dada (asumir que la lista No tiene elemento ficticio)
b) Ahora resuelva el inciso a) considerando que tiene un **elemento ficticio**.
c) Informar los elementos de la lista desde el final al principio.

CLASE 6 de 6

Ej. 11) Utilizar una Lista Simplemente Encadenada (LSE) para representar una lista de libros.

Una librería de la ciudad posee una secuencia ya cargada, de cada uno de sus libros. Por cada libro se tiene la siguiente información: titulo del libro, numero de ISBN, año de publicación y precio. Se desea incrementar el precio en un 20% de los libros publicados posteriores al año 1980 y anteriores al año 2015.

Desarrolle un Algoritmo que resuelva el problema.

Ej. 12) Dada una Lista simplemente encadenada (LSE con ficticio) que almacena la información de las precipitaciones pluviales de diversas localidades de una provincia, mediante un tipo registro:

TLluvia = <localidad ∈ Cadena, lluviaAnual ∈ R, lluviaMinima ∈ N, lluviaMáxima ∈ R>

dónde **localidad** es el nombre del pueblo o ciudad, **lluviaAnual** es el promedio anual de precipitaciones de esa localidad, **lluviaMinima**, es la precipitación más baja de todo el año para esa localidad, y **lluviaMáxima** es la precipitación más alta de todo el año para esa localidad.

Desarrolle un **algoritmo** que informe el nombre de las localidades en las cuales la **lluviaMinima** sea menor al promedio de lluvia anual (**lluviaAnual**) de todas las localidades incluidas en la lista.

Notas:

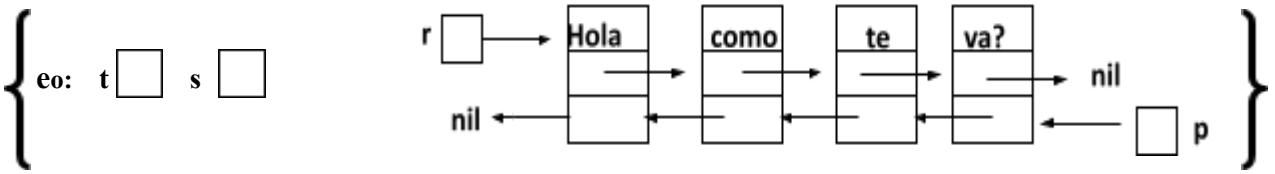
La lista tiene una variable puntero Cab que permite el acceso a la lista.

Cada nodo es de la forma TElem = <info ∈ TLluvia, next ∈ puntero de TElem>

Ej. 13) Dada una lista doblemente encadenada **LDE**, definida mediante TNodo, y las variables de tipo puntero **r, t, s** y **p**.

```
Tnodo = <info Є Cadena, next Є puntero a Tnodo, back Є puntero a Tnodo>

r, t, s, p Є puntero a TNodo
```



Desarrolle un algoritmo que:

- Inserte un nuevo nodo entre el primero (“Hola”) y el segundo (“como”), cuyo contenido en el campo info sea la cadena “Tito”.
- Modifique el contenido del tercer nodo (“te”) para que su nuevo contenido en el campo info sea la cadena “estás?”.

- Elimine el último nodo (“va?”).
- Finalmente, recorra la secuencia e informe el contenido de cada nodo.

Plan mínimo de ejercicios a realizar en clases:

1era Clase: Ej. 1) hacer en clases, luego 3.a) y 4.a), 4.b) y 4.c).-

Actividad a pasar a C el Ej. 1) en Laboratorio

2da Clase: Ej. 5) y Ej. 6)

3ra Clase: Ej.7) hacer en clases puntos a), b), analizar la solución al punto c), analizar el f), hacer con ellos el punto g). Desarrollar con los alumnos del Algoritmo el léxico y el cuerpo principal.

4ta Clase: Ej. 8) hacer en clases con alumnxs los incisos que faltan (a, b, c, e, g, i) en el orden dado y leer los que están hechos.

5ta Clase: hacer del Ej. 9) y 10) los incisos 10.b) y 10c) .

6ta Clase: hacer 11) y 13). Como actividades presentar el 12) en algoritmo y en programa C (son dos actividades distintas)