Práctica 7

Estructura de Datos Lista II

Algoritmos y Programación 1
Ciencia de Datos en Organizaciones

2025

Temas de la Práctica 7

- Contenidos
  - **Sintaxis (set de instrucciones nuevas)**
  - **Operaciones con listas**
    - Actualizar contenido
    - Agregar al principio
    - Buscar en la lista
    - Insertar Ordenado
    - Eliminar un nodo
    - Eliminar ocurrencias

Listas en Pascal Recordatorio

```
Program uno;

uses GenericLinkedList;

type
    Lista = specialize LinkedList<TIPO>;

Var
    L: Lista;
```

Necesitamos incluir el tipo Lista y sus operaciones

Cualquiera de los tipos vistos hasta ahora

Declara una variable del tipo de la lista

TIPOS DE DATO LISTAS

Operaciones

- Crear una lista vacía
- Agregar nodos al final de la lista
- Recorrer la lista

- **Nuevas**
  - Actualizar contenido de la lista
  - Agregar nodos al principio de la lista
  - Buscar en la lista con y sin orden
  - Insertar Ordenado en la lista
  - Eliminar un nodo en la lista con y sin orden
  - Eliminar ocurrencias en la lista con orden y sin orden

Sintaxis: Instrucciones en Pascal

sintaxis	semántica
Lista = specialize LinkedList <TIPO>;	Declaración del tipo Lista (va en la sección type)
L: Lista	Declaración de variable del tipo Lista (en la sección var)
L:= Lista.create()	Creación de lista vacía asignada a la variable L
L.reset()	Se posiciona al principio de la lista L, se debe hacer siempre antes de recorrer una lista.
L.eol()	Devuelve True si no hay más nodos en la lista L o False en caso contrario.
L.current ()	Devuelve el nodo actual de la lista L.
L.next()	Avanza al siguiente nodo de la lista L o a / si no tiene más nodos.
L.add(elemento)	Agrega un nodo con el contenido de elemento al final de la lista L.

Sintaxis: Instrucciones en Pascal

sintaxis	semántica
L.setCurrent(elemento)	Actualiza el valor actual de la lista L con el contenido de elemento.
L.addFirst(elemento);	Agrega un nodo con el contenido de elemento al principio de la lista L.
L.insertCurrent(elemento);	Inserta un nodo con el contenido de elemento en la posición actual de la lista L. Se utiliza para mantener el orden.
L.removeCurrent()	Elimina el nodo en la posición actual de la lista L.
/	Representa la lista vacía o fin de lista.

Ejercicio 1 P7

Utilizando el programa del ejercicio 3 Práctica 6, realizar los siguientes cambios:

- a.Modificar el módulo armarLista para que los elementos se guarden en la lista en el orden inverso en que fueron ingresados (agregar adelante).
- b.Modificar el módulo armarLista para que los elementos se guarden en la lista en orden ascendente (insertar ordenado).

```
program ListasPractica7;
type
      ListaEnteros = specialize LinkedList <integer>;

Procedure armarLista (var le:ListaEnteros);
 var
   num: integer;
 begin
        le:= ListaEnteros.create(); // crea la lista vacía
        read(num);
        while (num <> 0) do begin
                  le.add(num);
                  read(num);
        end;
 end;
Var {declaración de variables del programa principal}
    le : ListaEnteros;
    x: integer;
Begin     {cuerpo del programa principal}
   armarLista(le);
   //imprimir lista
   // modificar lista
 end.
```

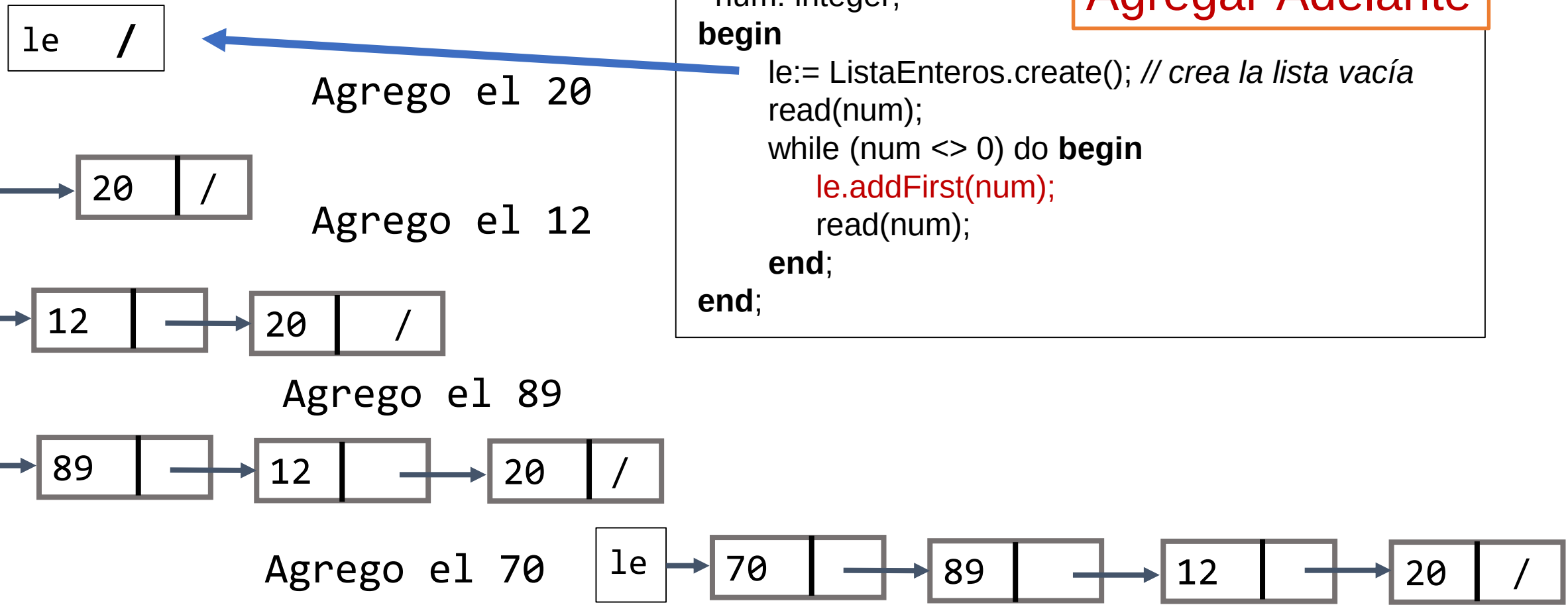
Agregar al Final

Ejercicio 1

- **Agregar adelante**

```
program Listas;
Uses GenericLinkedList;
type
     ListaEnteros = specialize LinkedList <integer>;
Procedure armarLista (var le:ListaEnteros);
var
  num: integer;
begin
     le:= ListaEnteros.create(); // crea la lista vacía
     read(num);
     while (num <> 0) do begin
          le.addFirst(num);
          read(num);
     end;
end;
```

Agregar Adelante



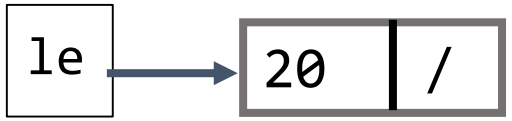
Ejercicio 1

- **Insertar Ordenado**

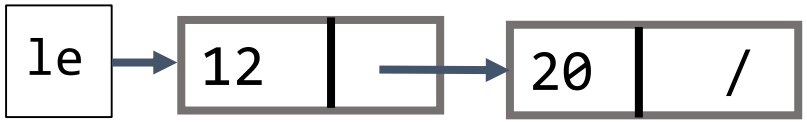
```
procedure  armarListaOrdenada(var le : ListaEnteros);
var
    num: integer;
begin
     le:= ListaEnteros.create(); // crea la lista vacía
     read(num);
       while (num <> 0) do begin
             insertar(le, num);
             read(num);
     end;
end;
```

Insertar Ordenado

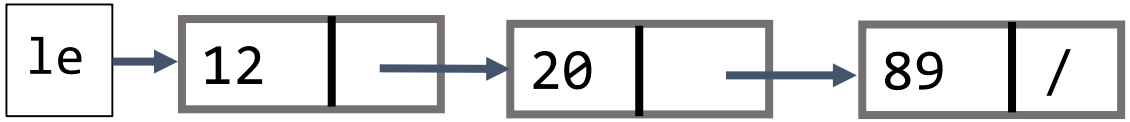
Agrego el 20



Agrego el 12



Agrego el 89



Agrego el 70

Ejercicio 1 b

- b.Modifica el módulo armarLista para que los elementos se guarden en la lista en orden ascendente (insertar ordenado).

```
program Listas;
type
     ListaEnteros = specialize LinkedList <integer>;

Procedure insertar (var le:ListaEnteros, valor: integer);

procedure armarListaOrdenada(var le : ListaEnteros);
var
  num: integer;
begin
     le:= ListaEnteros.create(); // crea la lista vacía
     read(num);
     while (num <> 0) do begin
           insertar(le, num);
           read(num);
     end;
end;
 Var {declaración de variables del programa principal}
   le : ListaEnteros;

Begin     {cuerpo del programa principal}
 armarListaOrdenada(le);
end.
```

Ejercicio 1 P7

```
program Listas;
Uses GenericLinkedList;
type
      ListaEnteros = specialize LinkedList <integer>;
Procedure insertar (var le:ListaEnteros; valor:integer);
var
  seguir: boolean;
begin
      le.reset();
      seguir:= true;
      while(not le.eol()) and seguir do begin
            if (le.current() > valor) then
                        seguir:= false
            else
                        le.next();
      end;
      le.insertCurrent(valor);
end;
```

Buscar donde insertar

Ejercicio 2

Dada una lista de lugares turísticos identificados por nombre y país:
c. Eliminar un lugar turístico que se recibe como parámetro
d. Eliminar todas las ocurrencias de un país que se recibe como parámetro

```
program Listas;
Uses GenericLinkedList;
type
      LugaresTuristicos = record
            nombre: String;
            pais: String;
      end;
      ListaLT = specialize LinkedList < LugaresTuristicos >;
Var {declaración de variables del programa principal}
    lt : ListaLT;
    lugarT, pais : String;
    exito: boolean
Begin     {cuerpo del programa principal}
   armarLista(lt); //se dispone
   read(lugarT);
   eliminarLT (lt, lugarT, exito);
   if (exito) then
      writeln('Elemento eliminado satisfactoriamente')
   else
       writeln('No se eliminó ningún elemento');
   read(pais);
   eliminarPais(lt,pais);
end.
```

Ejercicio 2 c

c. Eliminar un lugar turístico que se recibe como parámetro

```
Procedure eliminarLT (var lt: ListaLT; lugar: string; var encontre: boolean);
begin
        lt.reset();
        encontre:= false;
        while (not lt.eol()) and not encontre do begin
            if (lt.current().nombre=lugar) then
                    encontre:= true
            else
                    lt.next();
        end;
        if (encontre) then
             lt.removeCurrent();
end;
```

Ejercicio 2 d

c. Eliminar todas las ocurrencias de un país que se recibe como parámetro

```
Procedure eliminarPais (var lt: ListaLT; pais: string);
begin
      lt.reset();
      while (not lt.eol()) do begin
           if (lt.current().pais = pais) then
                 lt.removeCurrent();
           else
                 lt.next();
      end;
end;
```

Ejercicio 8

Se cuenta con una lista que contiene información de las ventas realizadas por una empresa de venta de pasajes aéreos. Cada venta está compuesta por un nombre de persona, código de vuelo, categoría de pasaje (1..4) y número de asiento. La lista puede contener 0, 1 o más registros por cada código de vuelo, y está ordenada por este campo. El costo de un pasaje depende de su categoría. **Se dispone de** una estructura que por cada categoría (1..4) se almacena su precio.

a) Generar una lista de registros que contenga por cada código de vuelo, el total de pasajes vendidos y el monto total recaudado.

b) Generar una lista de los códigos de vuelos cuya cantidad de pasajes vendidos sea mayor que 46. La lista debe ir generándose ordenada por monto total a medida que se realiza el punto a).

Cada venta está compuesta por un nombre de persona, código de vuelo, categoría de pasaje (1..4) y número de asiento. La lista puede contener 0, 1 o más registros por cada código de vuelo, y está ordenada por este campo. El costo de un pasaje depende de su categoría. Se dispone de una estructura que por cada categoría (1..4) se almacena su precio.

```
Program ej8;
Uses GenericLinkedList;
const
    CATEG = 4;
Type
   pasaje= record
            nombre: string;
            codigo: integer;
            categoria: integer;
            asiento: integer;
   end;
   resumen= record
            codigo: integer;
            cantidad: integer;
            monto: real;
   end;
```

```
listaPasajes = specialize LinkedList <pasaje>;
listaResumen = specialize LinkedList <resumen>;
vectorPrecios = array [1..CATEG] of real;

Var {variables del programa principal}
   lpasajes:listaPasajes;
   listaA,listaB:listaResumen;
   vPrecios: vectorPrecios;
begin
   cargarprecios(vPrecios);  // se dispone
   cargarlista(lPasajes); // se dispone
   procesarLista(lPasajes,vPrecios,listaA,listaB);
   …… // continua
end.
```

```
procedure ProcesarLista(lp:listaPasajes; vp:vectorPrecios; var la,lb: listaResumen);
var
  actual: resumen;
Begin
   lp.reset();
   la:=ListaResumen.create();
   lb:=ListaResumen.create();
   while (not (lp.eol()) ) do begin
      actual.codigo:=lp.current().codigo; //guarda el código actual
      actual.cantidad:=0; //acumula el total de pasajes para el código actual
      actual.monto:=0; // acumula el monto total para el código actual
      while (not (lp.eol()) and (actual.codigo =lp.current().codigo)) do begin
         actual.cantidad := actual.cantidad + 1;
         actual.monto := actual.monto  + vp[lp.current().categoria]; //accedo al precio
         lp.next();
      end;
      la.add(actual); // almaceno en lista a acumulando por codigo de vuelo;
      if (actual.cantidad > 46) then
         insertarOrdenado(lb,actual); // almaceno en lista b por cantidad de pasajes;
   end;
end;
```

Generar una lista de registros que contenga por cada código de vuelo, el total de pasajes vendidos y el monto total recaudado.

Generar una lista de los códigos de vuelos cuya cantidad de pasajes vendidos sea mayor que 46. La lista debe ir generándose ordenada por monto total a medida que se realiza el punto a).

Ejercicio 8

La lista debe ir generándose ordenada por monto total a medida que se realiza el punto a).

```
procedure insertarOrdenado(var l:  listaResumen; res:
resumen);
begin
    l.reset(); // se coloca al inicio de la lista.
    while not(l.eol()) and (l.current().monto < res.monto) do
        l.next();
    l.insertCurrent(res);
end;
```