Programación I

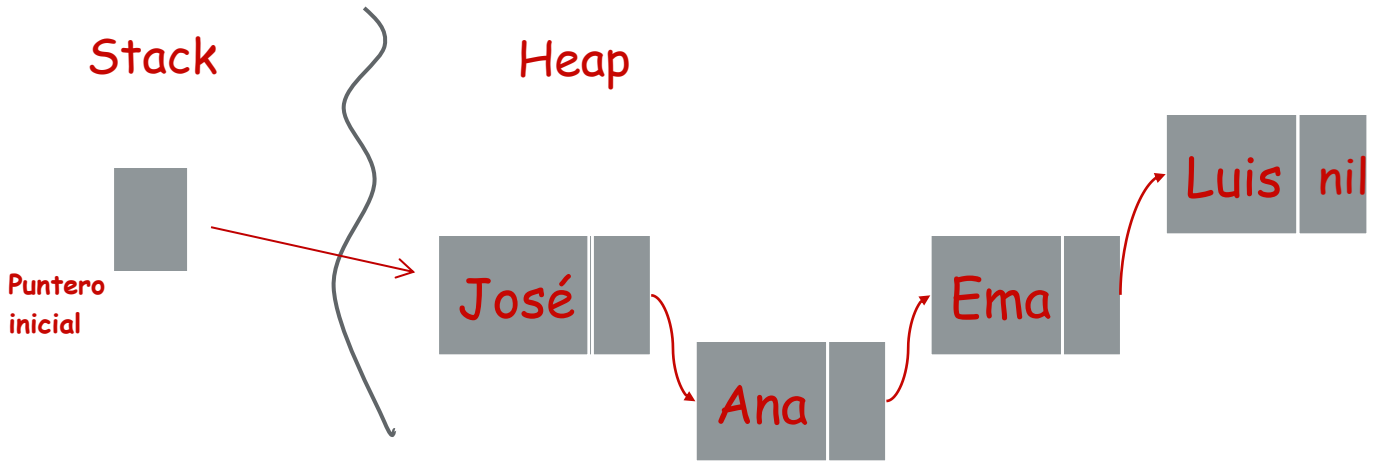
TEMAS de la CLASE

1 TIPO DE DATO LISTA ENLAZADA

- Concepto y Características
- Declaración del tipo en Pascal
- Operaciones frecuentes
- Ejercitación
- Análisis Comparativo Vector vs Lista

TIPO DE DATO LISTA - CONCEPTO

- Colección de elementos homogéneos, con una relación lineal que los vincula, es decir que cada elemento tiene un único predecesor (excepto el primero), y un único sucesor (excepto el último).
- Los elementos que la componen no ocupan posiciones secuenciales o contiguas de memoria. Es decir pueden aparecer dispersos en la memoria dinámica, pero mantienen un orden lógico interno.
- Se accede a sus elementos secuencialmente.



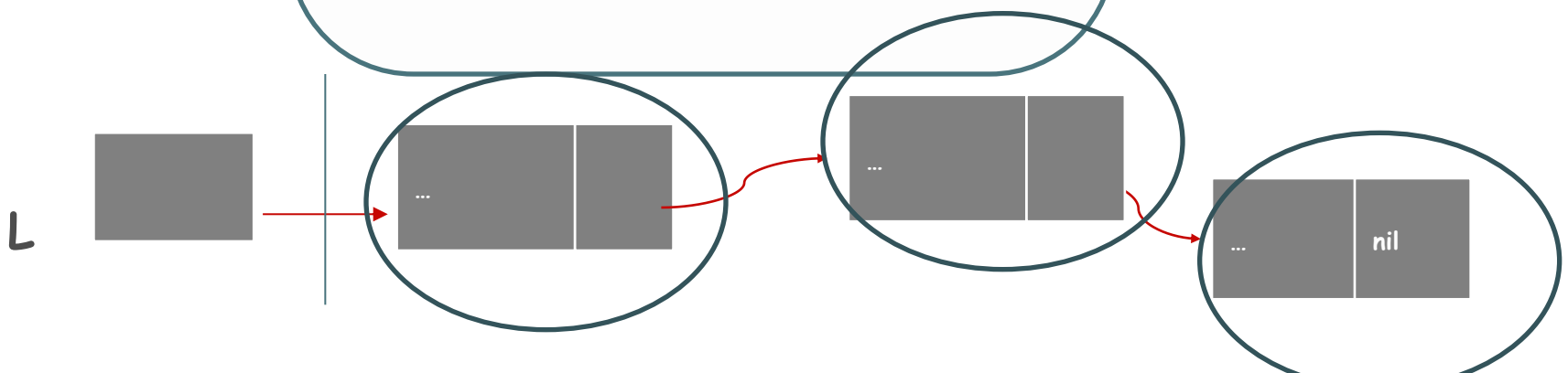
TIPO DE DATO LISTA - CARACTERISTICAS

- Están compuesta por nodos
- Los nodos se conectan por medio de enlaces o punteros
- Cuando se necesitan agregar nodos a la estructura, se solicita espacio adicional
- Cuando existen nodos que ya no se necesitan, pueden ser borrados, liberando memoria
- Siempre se debe conocer la dirección del primer nodo de la lista (puntero inicial) para acceder a la información de la misma
- El último nodo de la lista se caracteriza por tener su enlace en Nil

LISTAS – DECLARACION EN PASCAL

```
Type
  info = …;
  Lista = ^ nodo;
  nodo = record
           Datos : info;
           Sig: Lista;
         End;

Var
  L : Lista;
```



LISTAS – OPERACIONES

- Recorrer una lista
- Buscar un elemento en la lista
- Crear una lista vacía
- Agregar un elemento al principio de una lista
- Agregar un elemento al final de una lista
- Eliminar un elemento de la lista
- Insertar un nuevo elemento en una lista ordenada

RECORRIDO DE UNA LISTA

Se quiere imprimir los datos guardados en una lista enlazada. Para ello es necesario recorrer la lista completa, desde el primer nodo al último.

Supongamos la declaración:

```
Type
  cadena50 = string[50];
  persona= record
              nom: cadena50;
              edad : integer
            end;
  lista= ^nodo;
  nodo = record
           datos : persona;
           sig   : lista;
         end ;
 var L: lista;
 begin
   CargarLista (L);
   recorrido (L);
 end.
```

Observar parámetro...

Recorre hasta el final

```
Procedure recorrido ( pri : lista);

Begin
 while (pri <> NIL) do begin
   write (pri^.datos.nom,
          pri^.datos.edad) ;
   pri:= pri^.sig
 end;
end;
```

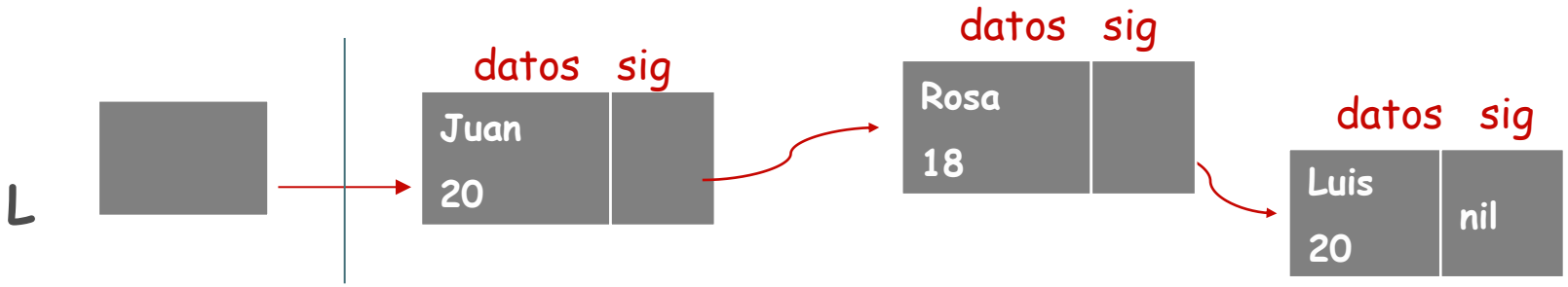
```
Type
  cadena50 = string[50];
  persona= record
              nom: cadena50;
              edad : integer
            end;
  lista= ^nodo;
  nodo = record
           datos : persona;
           sig   : lista;
         end ;
var L: lista;
begin
  CargarLista (L);
  recorrido (L);
end.
```

```
Procedure recorrido ( pri : lista);

Begin                          Recorre hasta el final
 while (pri <> NIL) do begin
   writeln (pri^.datos.nom, ' ',
            pri^.datos.edad) ;
   pri:= pri^.sig
 end;
end;
```

pri | nil

Juan 20
Rosa 18
Luis 20



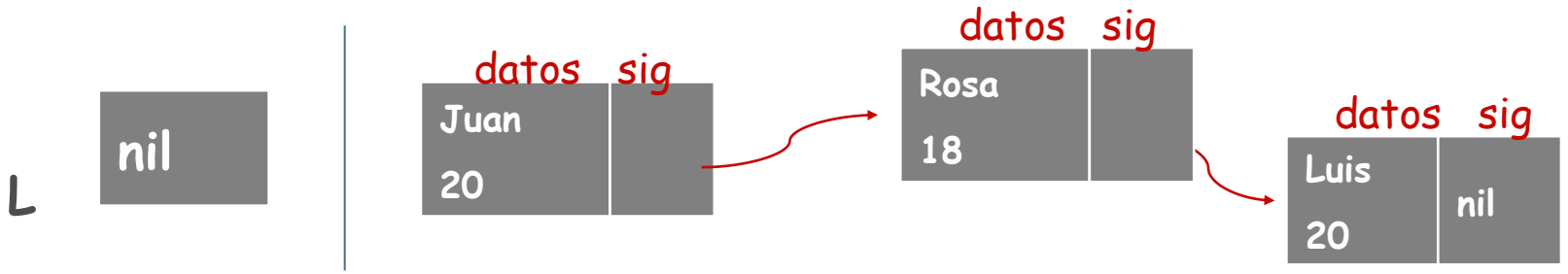
```
Type
  cadena50 = string[50];
  persona= record
              nom: cadena50;
              edad : integer
            end;
  lista= ^nodo;
  nodo = record
           datos : persona;
           sig   : lista;
         end ;
var L: lista;
begin
  CargarLista (L);
  recorrido (L);
end.
```

```
Procedure recorrido ( var pri : lista);

Begin
 while (pri <> NIL) do begin
   writeln (pri^.datos.nom, ' ',
            pri^.datos.edad) ;
   pri:= pri^.sig
 end;
end;
```

MAL!!!

Juan 20
Rosa 18
Luis 20



BUSQUEDA DE UN ELEMENTO EN UNA LISTA

Se quiere saber si existe un elemento determinado en una lista. Se debe recorrer la lista desde el primer nodo hasta encontrar el elemento o bien hasta llegar al final de la lista.

Supongamos la declaración:

```
Type
 cadena50 = string[50];
  persona= record
              nom:cadena50;
             edad:integer
             end;
 lista = ^nodo;
 nodo = record
         datos : persona;
          sig  : lista;
        end;
```

```
function buscar (pri: lista; x:cadena50):boolean;
Var
   encontre : boolean;

begin
 encontre := false;
 while (pri <> NIL) and (not encontre) do
   if x = pri^.datos.nom  then
                            encontre:= true
                          else
                            pri:= pri^.sig;

 buscar := encontre
End;
```

Recorre hasta el final

Recorre hasta encontrarlo

CREAR LISTA VACIA

La operación de Crear Lista Vacía es simplemente asignarle Nil a su puntero inicial.

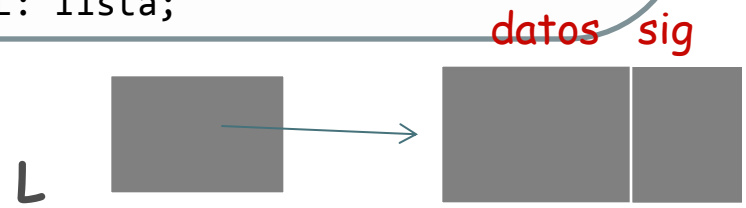
Por ejemplo:

```
Type
 cadena50 = string[50];
  persona= record
              nom:cadena50;
              edad : integer
             end;
 lista = ^nodo;
 nodo = record
         datos : persona;
           sig  : lista;
        end;
var
  L: lista;
```

```
Begin

  ......
  L:=nil;
  .....

End.
```



Observar que NO se usa NEW!!

AGREGAR UN ELEMENTO AL PRINCIPIO DE LA LISTA

Supongamos que se ingresan el nombre y la edad de personas, hasta que se ingresa la edad 0. Los datos de cada persona se deben guardar en una lista.

```
Type
 cadena50=string[50];
 persona= record
           nom:cadena50;
           edad:integer;
          end;
 lista = ^nodo;
 nodo = record
         datos: persona;
          sig: lista;
        end;
Var
 L : Lista;
 p : persona;
```

```
Procedure AgregarAdelante
(var L:lista; per:persona);

Var nue:Lista;
Begin
  New(nue);
  nue^.datos:=per;
  nue^.sig:=L;
  L:=nue;
End;
```

```
Begin {prog. ppal}
 L:=nil;
 leerPersona (p);
 While (p.edad <> 0) do Begin
    AgregarAdelante (L, p);
    leerPersona (p);
 End;
End.
```

AGREGAR UN ELEMENTO AL FINAL DE LA LISTA

Supongamos que se ingresan el nombre y la edad de personas, hasta que se ingresa la edad 0. Los datos de cada persona se deben guardar en una lista, respetando el orden de ingreso.

```
Type
 cadena50 = string[50];
 persona= record
           nom:cadena50;
           edad:integer
           end;
 lista = ^nodo;
 nodo = record
         datos: persona;
         sig : lista;
        end;

Var
 L : Lista;
 p : persona;
```

```
Procedure AgregarAlFinal
(var L:lista; per:persona);

Var
   ……..
Begin
    …..;
End;
```

```
Begin {prog. ppal}
 L:=nil;
 leerPersona (p);
 While (p.edad <> 0) do Begin
   AgregarAlFinal (L, p);
   leerPersona (p);
 End;
End.
```

AgregarAlFinal recibe como parámetros el puntero inicial de la lista y los datos de la persona que se guarda

```
procedure AgregarAlFinal (var pri: lista; per: persona);
var  act, nue : lista;

begin
 new (nue);
 nue^.datos:= per;
 nue^.sig := NIL;
 if pri <> Nil then begin
                  act := pri ;
                  while  (act^.sig <> NIL ) do act := act^.sig ;
                  act^.sig := nue ;
                end
                else
                  pri:= nue;
end;
```

```
Begin {prog. ppal}
 L:=nil;
 leerPersona (p);
 While (p.edad <> 0) do Begin
   AgregarAlFinal (L, p);
   leerPersona (p);
 End;
End.
```

AGREGAR UN ELEMENTO AL FINAL DE LA LISTA (otra solución)

Podríamos plantear la operación de AgregarAlFinal2 como un procedimiento que recibe el puntero inicial, el puntero al último nodo y el número a guardar…

```
Type
 cadena50 = string[50];
 persona= record
           nom: cadena50;
           edad : integer
           end;
 lista = ^nodo;
 nodo = record
         datos: persona;
         sig: lista;
        end;
Var
 L, Ult : Lista;
 p : persona;
```

```
Procedure AgregarAlFinal2
(var L, Ult: lista; per:persona);

Var
 ….
Begin
 ….
 …..
End;
```

```
Begin {prog. ppal}
 L:=nil; ult:=nil;
 leerPersona (p);
 While (p.edad <> 0) do begin
   AgregarAlFinal2 (L, Ult, p);
   leerPersona (p);
 End;
End.
```

AGREGAR UN ELEMENTO AL FINAL DE LA LISTA (otra solución)
(con puntero al último nodo)

```
procedure AgregarAlFinal2 (var pri, ult: lista; per: persona);
var  nue : lista;

begin
 new (nue);
 nue^.datos:= per;
 nue^.sig := NIL;
 if pri <> Nil then
                  ult^.sig := nue
               else
                  pri := nue;
 ult := nue;
end;
```

Si la lista tiene elementos

Si la lista no tiene elementos

BORRAR UN ELEMENTO DE LA LISTA

Supongamos que se dispone de una lista de personas y se quiere eliminar a una persona cuyo nombre se lee de teclado, de ser posible.

```
Type
 cadena50 = string[50];
 persona= record
            nom: cadena50;
            edad: integer;
           end;
 lista = ^nodo;
 nodo = record
         datos: persona;
         sig: lista;
        end;
Var
 L : Lista;
 nombre : cadena50;
 éxito: boolean;
```

```
Procedure BorrarElemento
(var L:lista; nom:cadena50; var éxito:boolean);

Var
   ……..
Begin
   …..;
   ….
End;
```

```
Begin {prog. ppal}
 CargarLista (L);
 read (nombre);
 BorrarElemento(L, nombre, exito);
 if éxito then write (‘Se eliminó’)
          Else write (‘No existe’);
End.
```

AgregarAlFinal recibe como parámetros el puntero inicial de la lista y los datos de lal persona que se guarda

BORRAR UN ELEMENTO DE LA LISTA

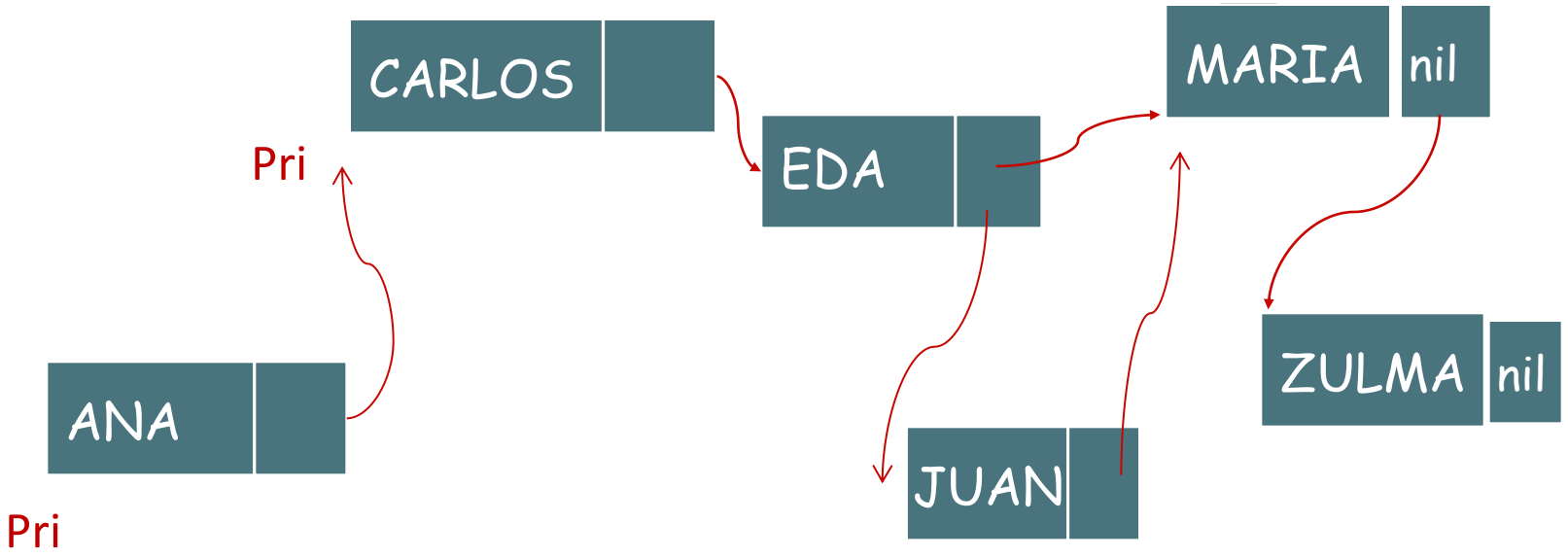
```
Procedure BorrarElemento (var pri:lista; nom:cadena50; var exito: boolean);
var ant, act: lista;
begin
  exito := false;
  act := pri;
  {Recorro mientras no se termine la lista y no encuentre el elemento}
  while  (act <> NIL)  and (act^.datos.nom <> nom) do begin
      ant := act;
      act := act^.sig
  end;
  if (act <> NIL) then begin
                    exito := true;
                      if (act = pri) then  pri := act^.sig;
                                     else  ant^.sig:= act^.sig;
                      dispose (act);
                  end;
 end;
```

El dato a borrar es el primero

El dato a borrar es uno cualquiera

INSERTAR UN ELEMENTO EN UNA LISTA

Supongamos que tenemos una lista de personas ordenadas alfabéticamente y queremos insertar los nombres Ana, Zulma y Juan.



Pasos a seguir:

1. Pedir espacio en memoria para el nuevo nodo
2. Guardar los datos en el nodo
3. Buscar posición donde se debe insertar (secuencialmente a partir del puntero inicial)
4. Reacomodar punteros. Considerar los tres casos:
   a. El nuevo elemento va en el inicio de la lista.
   b. El nuevo elemento va en el medio de dos existentes.
   c. El nuevo elemento va al final de la lista.

INSERTAR UN ELEMENTO EN UNA LISTA

Supongamos que se dispone de una lista de personas ordenada alfabéticamente por el nombre y se desea incorporar la información de una persona a dicha lista. Los datos de la persona se leen de teclado.

```
Type
 cadena50 = string[50];
 persona= record
            nom:cadena50;
            edad:integer;
          end;
 lista = ^nodo;
 nodo = record
          datos:persona;
          sig:lista;
        end;
Var
 L: Lista;
 p: persona;
```

```
Procedure InsertarElemento
(var L:lista; per:persona);

Var
   ……..
Begin
   …..;
   ….
End;
```

```
Begin {prog. ppal}
  CargarLista (L);
  leerPersona (p);
  InsertarElemento (L, p);
End.
```

InsertarElemento recibe como parámetros el puntero inicial de la lista y los datos de la persona que se guarda

INSERTAR UN E

Pasos a seguir:

1. Pedir espacio en memoria para el nuevo nodo
2. Guardar los datos en el nodo
3. Buscar posición donde se debe insertar (secuencialmente a partir del puntero inicial)
4. Reacomodar punteros. Considerar los tres casos:
   1. El nuevo elemento va en el inicio de la lista.
   2. El nuevo elemento va en el medio de dos existentes.
   3. El nuevo elemento va al final de la lista.

```
Procedure InsertarElemento ( var pri: lista; per: persona);
var ant, nue, act: lista;
begin
  new (nue);
  nue^.datos := per;
  act := pri;
  ant := pri;
{Recorro mientras no se termine la lista y no encuentro la posición correcta}
  while (act<>NIL) and (act^.datos.nombre < per.nombre) do begin
      ant := act;
      act := act^.sig ;
  end;
  if (ant = act)  then pri := nue   {el dato va al principio}
                  else  ant^.sig  := nue; {va entre otros dos o al final}
  nue^.sig := act ;
end;
```

Ejercicio 1 – Problema del supermercado

Se desean procesar los productos de una venta del supermercado. De cada producto se conoce código, nombre, marca, precio y fecha de vencimiento. El ingreso de los productos finaliza cuando se lee el código -1. Se pide generar una lista con los códigos e identificación de los productos de marca “SanCor”.

100
Bebida
Coca Cola
10
31-12-2017

121
Nido
Leche en polvo
35
30-01-2020

80
Leche Larga Vida
SanCor
18
30-01-2020

130
Amanda
Yerba
25
31-12-2015

50
Arcor
Tomate al natural
15
25-10-2016

75
Ilolay
Leche Larga vida
25
31-12-2016

200
Dulce de leche
SanCor
15
31-12-2016

ANALISIS COMPARATIVO VECTOR vs LISTA

Analicemos algunas consideraciones que entran en juego en el momento de elegir una estructura de las vistas hasta ahora:

- **Espacio**
- **Tiempo**
  - Recuperar datos
  - Almacenar datos
- **Parámetros**

ANALISIS COMPARATIVO VECTOR vs LISTA

Espacio: se refiere a la cantidad de memoria consumida por una estructura de datos dada.

Si se supone igual cantidad de datos en las dos estructuras se puede afirmar que:

- Vectores : son más económicos.
- Listas : requieren espacio extra para los enlaces.

Si no se conoce la cantidad de datos que contendrá cada estructura, qué análisis se puede hacer en:

- Vectores?
- Listas?

ANALISIS COMPARATIVO VECTOR vs LISTA

Tiempo: se refiere al tiempo que toma almacenar o recuperar datos, entre otros.

Se tendrá que analizar:

- Tiempo empleado para Recuperar Datos en:
  - Vectores
  - Listas
- Tiempo empleado para Almacenar Datos en:
  - Vectores
  - Listas

ANALISIS COMPARATIVO VECTOR vs LISTA

Parámetros: analizar cual es el costo del pasaje de parámetros por valor y por referencia en:

- Arreglos
- Listas