PROGRAMACIÓN I

AÑO 2025

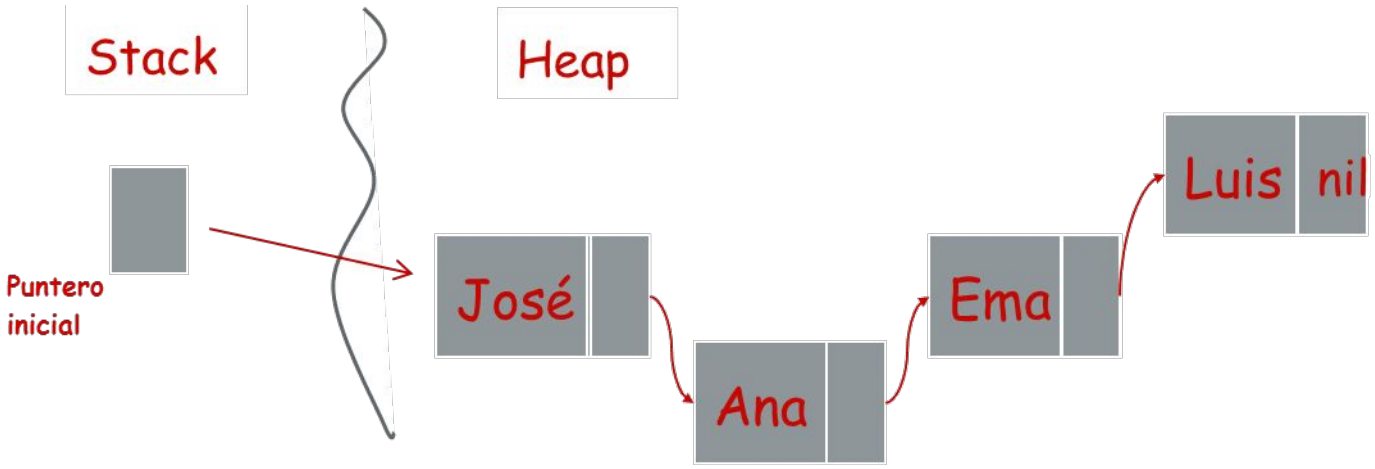
TEMAS de la CLASE

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.



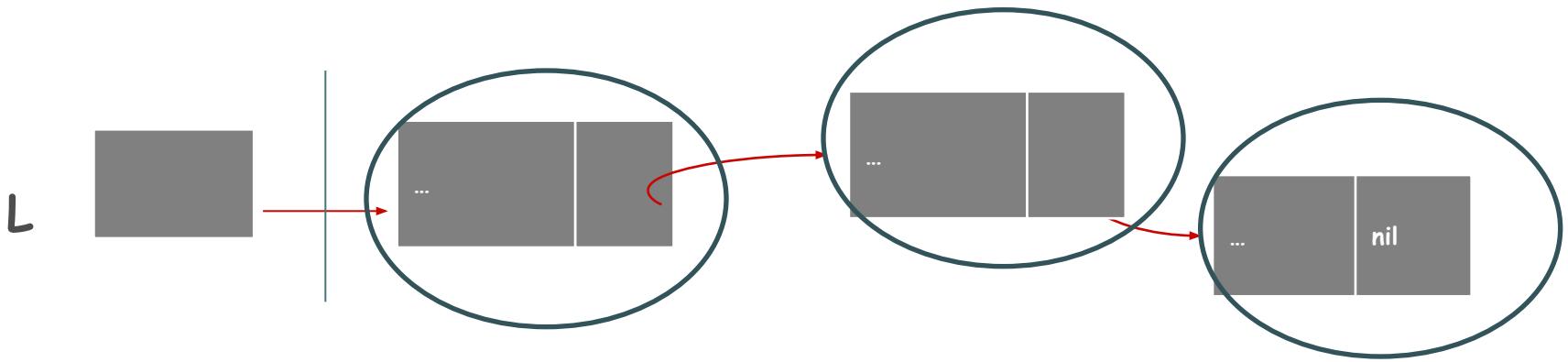
TIPO DE DATO LISTA - CARACTERÍSTICAS

- ➔ 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 – DECLARACIÓN EN PASCAL

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

Var
  L : Lista;
```



LISTAS – OPERACIONES

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

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.

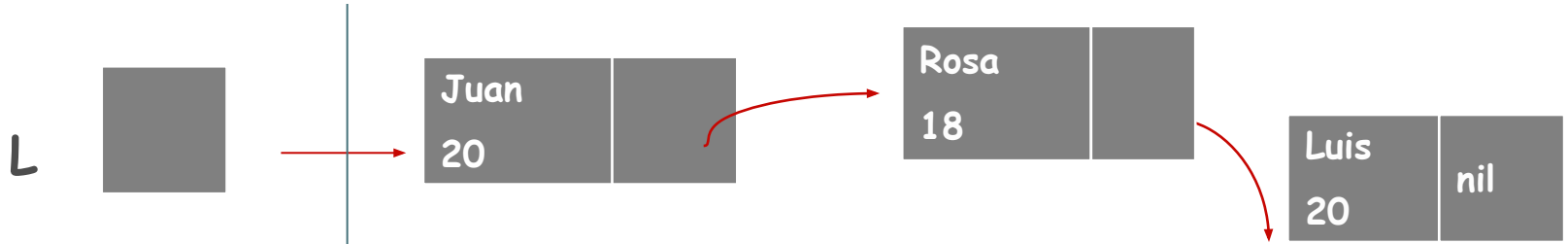
```
Type
 cadena50=string[50];
 persona= record
            nom:cadena50;
            edad:integer;
      end;
 lista = ^nodo;
 nodo = record
          datos: persona;
            sig: lista;
        end;
Var
 L : Lista;
 p : persona;
```

```
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.
```

Al leer Luis 20, Rosa 18 y Juan 20, la lista quedará armada de la siguiente forma:



RECORRIDO DE UNA LISTA

Se dispone de una lista y se quieren imprimir los datos guardados en ella. 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...

```
Procedure recorrido ( pri : lista);

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

Recorre hasta el final

BÚSQUEDA DE UN ELEMENTO EN UNA LISTA

Se dispone de una lista y necesitamos saber si existe un elemento determinado en ella. 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

Ejercicio 1

Se desean procesar los productos de una venta del supermercado. Cada producto está caracterizado por código de producto, tipo, código de marca (valor entero entre 200 y 300) y precio. El ingreso de los productos finaliza cuando se lee el código -1. Implementar un programa que ingrese la información, genere una lista con los códigos y precio de los productos de tipo “Limpieza” e informe la cantidad de productos de cada marca.

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;
 lista_pri_ult = record
              pri: lista;
              ult: lista;
             end;
Var
 lpu : lista_pri_ult;
 p : persona;
```

```
Procedure AgregarAlFinal2
(var lpu: lista_pri_ult; per:persona);

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

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

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

```
procedure AgregarAlFinal2 (var lpu: lista_pri_ult; per: persona);
var  nue : lista;

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

Si la lista no tiene elementos

Si la lista 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.
```

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

Ejercicio 2

Un supermercado dispone de una estructura con la información de sus productos. Cada producto está caracterizado por código de producto, tipo, código de marca (valor entero entre 200 y 300) y precio. La estructura está ordenada de manera ascendente por código de marca. Implementar un programa con:

a) Un módulo que reciba la estructura con la información de los productos y dos valores reales, y retorne, en una estructura adecuada, todos los productos con precio entre los dos valores recibidos. La estructura debe generarse ordenada por código de marca.

b) Un módulo que reciba la estructura con la información de los productos y un código de marca y retorne dicha estructura sin los productos de dicho código..

INSERTAR UN ELEMENTO EN UNA LISTA

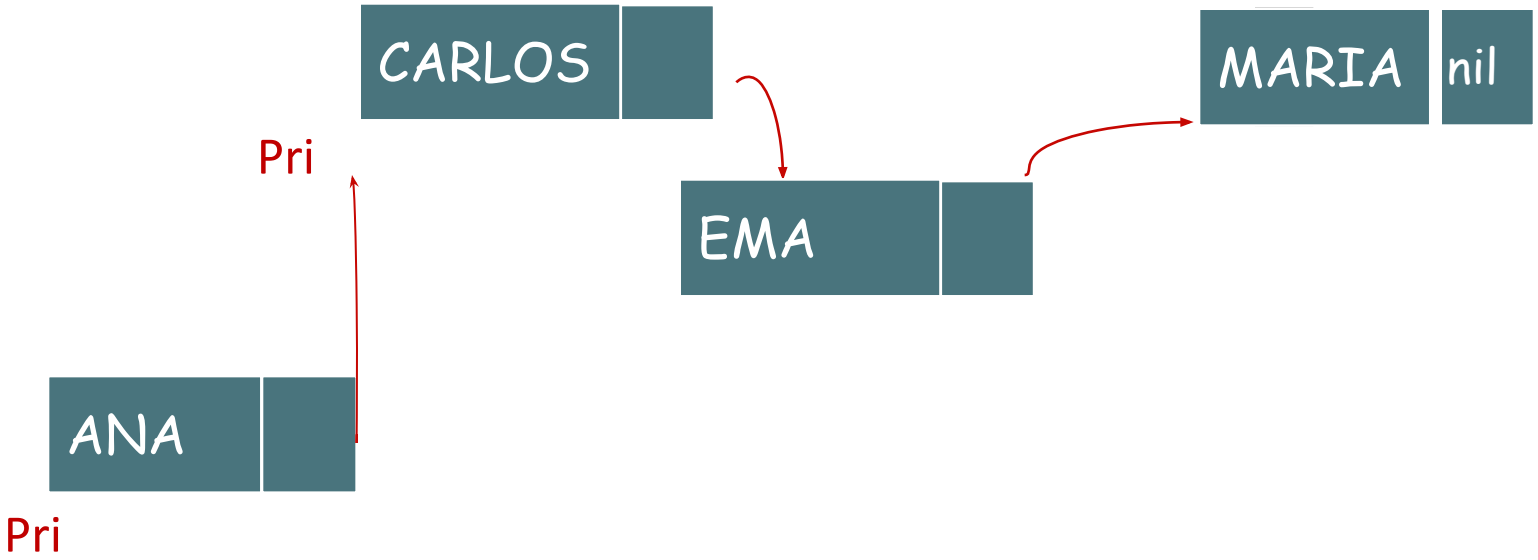
Supongamos que tenemos una lista de personas ordenadas alfabéticamente y queremos insertar los nombres Ana, Zulma y Juan.

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

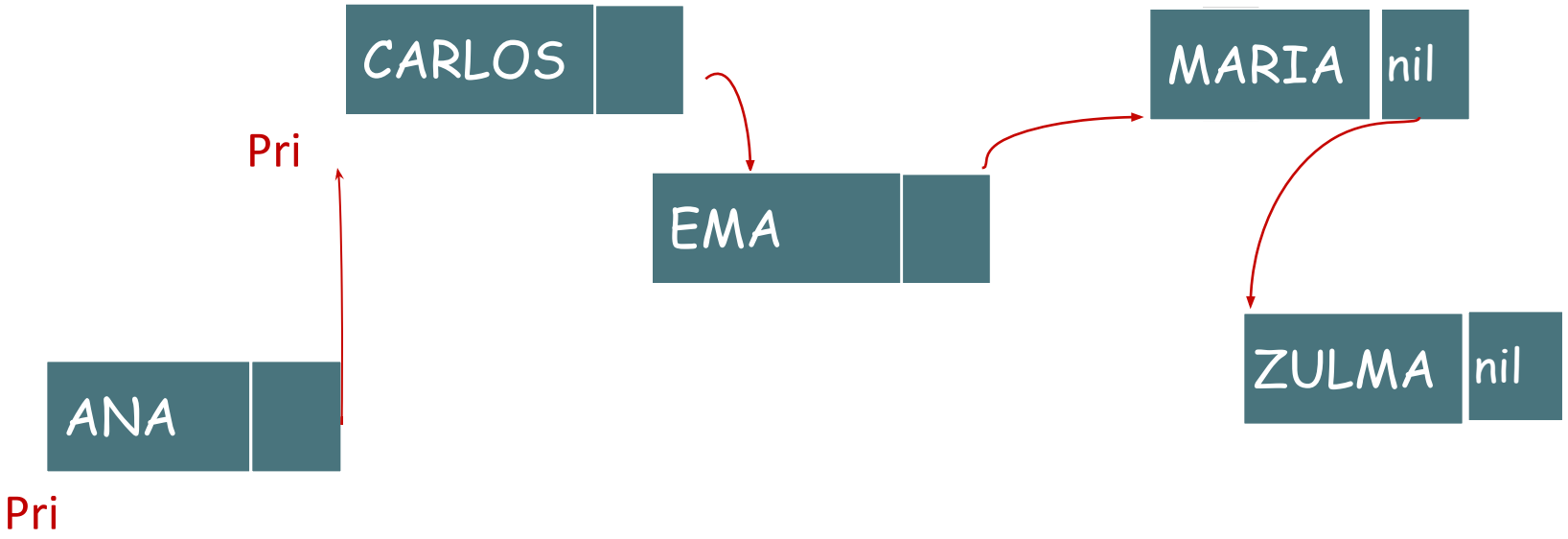


Pasos a seguir:

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

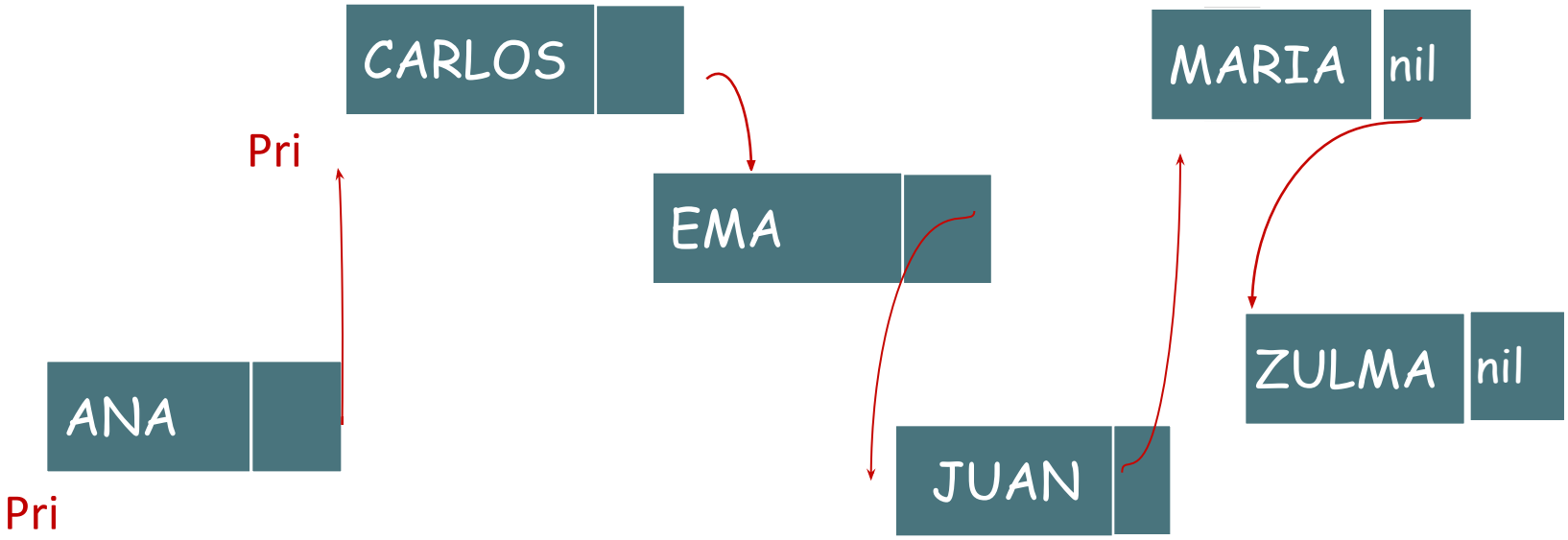


Pasos a seguir:

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 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 a través del 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 3

Un supermercado dispone de una estructura con la información de sus productos. Cada producto está caracterizado por código de producto, tipo, código de marca (valor entero entre 200 y 300) y precio. La estructura está ordenada por tipo. Implementar un programa con:

a) Un módulo que reciba la estructura con la información de los productos y dos valores reales, y retorne, en una estructura adecuada, todos los productos con precio entre los dos valores recibidos. La estructura generada debe estar ordenada por código de marca.

b) Un módulo que reciba la estructura con la información de los productos e informe la cantidad de productos existentes de cada tipo.