LISTAS – MERGE

Explicación Práctica

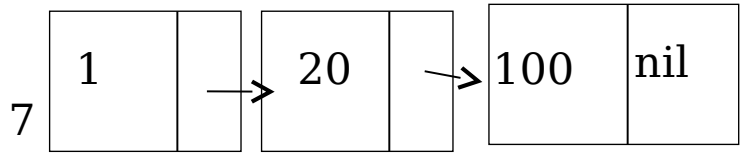
Programación I - 2024

Facultad de Informática y Facultad de

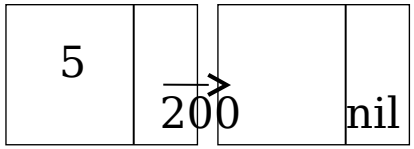
Listas - Merge

Suponga que se tienen dos listas creadas de manera ordenada. Se pide implementar un algoritmo que combine ambas listas y genere una tercer lista ordenada.

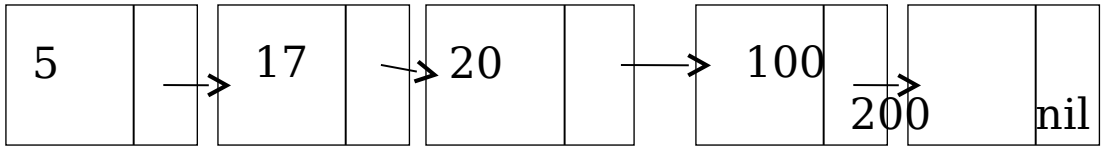
Ambas listas están ordenadas por el mismo criterio. La lista nueva también es generada ordenada por el mismo criterio.



Pri 1

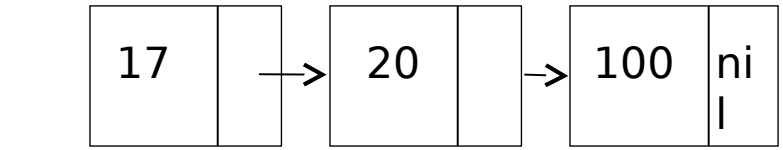


Pri2

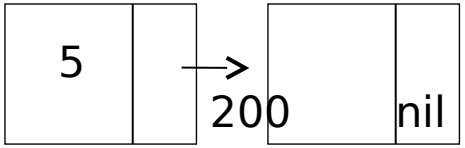


Pri 3

Listas - Merge



Pri1



Pri2

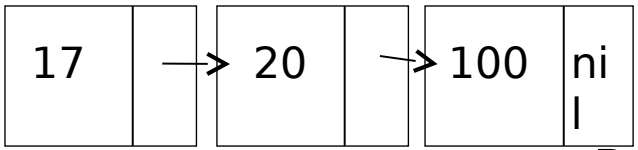
Obtengo el mínimo → 5 → Agrego el mínimo a la lista pri3 → Avanzo en la lista que tuvo el mínimo (pri2)

Obtengo el mínimo → 17 → Agrego el mínimo a la lista pri3 → Avanzo en la lista que tuvo el mínimo (pri1)

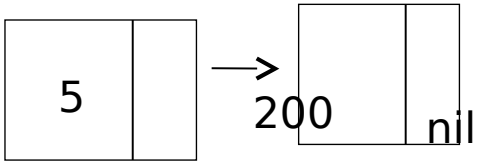
Obtengo el mínimo → 20 → Agrego el mínimo a la lista pri3 → Avanzo en la lista que tuvo el mínimo (pri1)

Obtengo el mínimo → 100 → Agrego el mínimo a la lista pri3 → Avanzo en la lista que tuvo el mínimo (pri1=nil)

LISTAS – MERGE



Pri 1



Pri 2

Obtengo el mínimo → La lista que se terminó no se puede comparar ya que se estaría comparando con nil, entonces devuelve lo que está apuntando pri2.

Obtengo el mínimo → Como ambas listas se terminaron el algoritmo termina

LISTAS – MERGE

```
CONST
   VALOR_CORTE = 9999;
Type
   lista = ^nodo;
   nodo=record
           datos: integer;
           sig: lista;
   end;
...
Var
     L1,L2,L3: lista;
Begin
   CrearListaOrdenada (L1);
   CrearListaordenada (L2);
   L3:= nil;
   Merge (l1, l2, l3);
End.
```

LISTAS – MERGE

```
Procedure merge (l1,l2: lista; var l3:lista);
Var
   min : integer; ultL3: lista;
Begin
  minimo(l1, l2, min);
  while (min <> VALOR_CORTE) do
   begin
      agregarAtras (l3, ultL3,min);
      minimo (l1, l2, min);
   end;
End.
```

Qué representa VALOR_CORTE?

LISTAS – MERGE

¿Por qué paso los 3 parámetros por referencia?

```
Procedure minimo( var l1:lista; var l2: lista; var min:integer);
Var
Begin
 {ambas listas vacías}
  if (l1 y l2 están vacías) then min:= VALOR_CORTE;
  else
{si las dos listas tienen elementos, obtengo el menor}
            if (l1 no está vacía) and (l2 no está vacía) then
                  elijo el número más chicos de los dos
                  avanzo el puntero del número que era más chico
            else
                  if (l1 no está vacía) then {sólo l1 tiene elementos}
                        el min es de l1 y avanzo l1
                  else {sólo l2 tiene elementos}
                        el min es de l2 y avanzo l2
end;
```

LISTAS – MERGE

```
Procedure minimo (var l1:lista; var l2: lista; var min:integer);
Begin
  if (l1 =  nil) and (l2=nil) then min:= VALOR_CORTE;
  else
        if (l1<> nil) and (l2<> nil) then
              if (l1^.datos <= l2^.datos) then begin
                    min:= l1^.datos;
                    l1:= l1^.sig;
              end
              else begin
                    min:= l2^.datos;
                    l2:= l2^.sig;
              end
        else
            //alguna de las listas está vacía
              if (l2 = nil) then begin  //L2 vacía, L1 <> nil
                        min:= l1^.datos;
                        l1:= l1^.sig;
              end
              else begin                    //L1 vacía, L2 <> nil
                        min:= l2^.datos;
                        l2:= l2^.sig
              end;
end;
```

Merge de Listas Simples Enlazadas

Ejemplo:

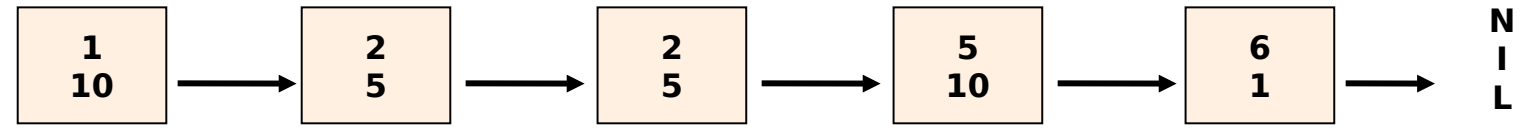
Se disponen de 2 listas que contienen información de las ventas realizadas por c/u de las 2 sucursales de un supermercado. De cada venta se conoce: el **código de producto y** la **cantidad vendida**.

Se dispone además de una lista con los precios unitarios de cada producto.

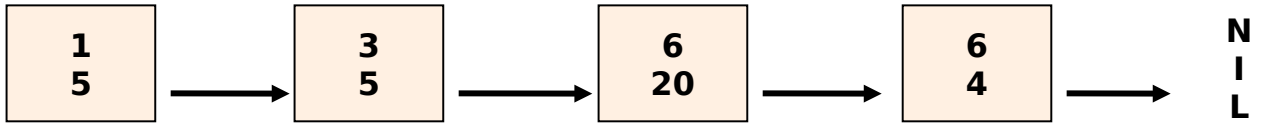
Realizar un modulo que procese los datos y genere una **nueva lista** ordenada por cod. de producto que contenga para cada producto vendido su codigo, la cantidad y el monto total vendido.

Importante: las listas están ordenadas por código de producto y deben recorrerse una única vez.

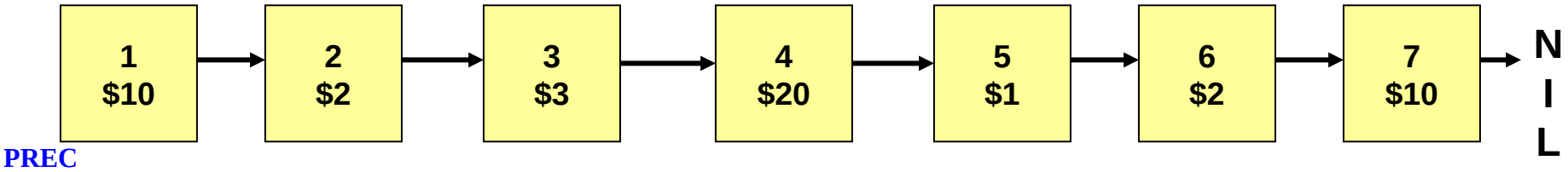
Merge de Listas Enlazadas

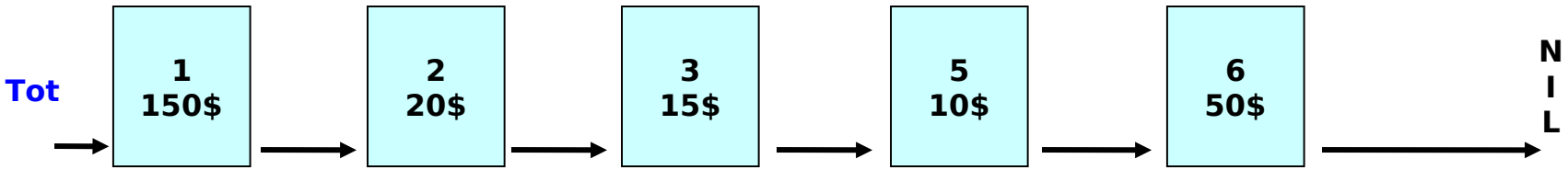


lista1



lista2





Merge de Listas Enlazadas

procedure totalizar (l1, l2 (listas de entradas); l3 (lista de salida));

Se posiciona al principio de las dos listas
Determina el minimo (l1, l2, minimo);

mientras las listas no estén vacías
inicializar variables para el nuevo producto que se procesa

mientras el mínimo no cambia, acumulo las ventas del mismo prod.
totaliza la cantidad para el producto
Determina nuevo minimo (l1, l2, minimo);

Cuando el mínimo cambia *(salí del mientras)*
Busco en la lista de precios el precio unitario del producto actual
Guardo en la lista nueva el producto acumulado (agregar atrás en l3)

Merge de Listas Enlazadas

```
type
    venta = record
          cod_pro: integer;
          cant_vend: integer;
     end;
    lis_ventas = ^nodo_ven;
    nodo_ven = record
          ven: venta;
          sig: lis_ventas;
     end;

    precio_prod = record
          cod_pro: integer;
          pre_uni: real;
      end;
     lis_precios = ^nodo_pre;
     nodo_pre = record
          dato: precio_prod;
          sig:lis_precios;
     end;

    total_prod = record
          cod_pro: integer;
          cant: integer;
          monto: real;
    end;

    lis_totales = ^nodo_tot;
     nodo_tot = record
          dato: total_prod;
           sig: lis_totales;
     end;
```

```
procedure totalizar(l1, l2: lis_ventas; lprec:  lis_precios; var ltot:
    lis_totales);
var   min: venta;
      prod_actual, total : integer;
       preciou: real;
       datototal:  total_prod ;
      ult : lis_totales;
begin
    ltot := nil;
    minimo (l1, l2, min);
     {mientras las listas no estén vacías}
    while ( min.cod_pro <> VALOR_CORTE) do begin
              prod_actual := min.cod_pro;
              total:= 0;
{ Mientras el mínimo no cambia, acumulo las ventas del mismo prod.}
           while (prod_actual = min.cod_pro) do begin
              total := total + min.cant_vend;
                 minimo (l1, l2, min);
           end;
{Cuando el mínimo cambia guardo en la lista nueva el producto
    acumulado}
              BuscarPrecioUnitario(lprec, prod_actual, preciou);
              datototal.cod_pro:= PROD_ACTUAL;
              datototal.cant:= total;
              datototal.monto:= total*preciou;
              agregar_atras(ltot, ult, datototal);
    end;
```

```
Procedure minimo (var l1: lis_ventas;  var l2: lis_ventas; var min: venta);
Begin
  if (l1 =  nil) and (l2=nil) then min.cod_pro:= VALOR_CORTE;
  else
        if (l1<> nil) and (l2<> nil) then
              if (l1^.ven. cod_pro <= l2^. ven. cod_pro ) then begin
                    min:= l1^.ven;
                    l1:= l1^.sig;
              end
              else begin
                    min:= l2^.ven;
                    l2:= l2^.sig;
              end
        else
            //alguna de las listas está vacía
              if (l2 = nil) then begin  //L2 vacía, L1 <> nil
                        min:= l1^.ven;
                        l1:= l1^.sig;
              end
              else begin                    //L1 vacía, L2 <> nil
                        min:= l2^.ven;
                        l2:= l2^.sig
              end;
end;
```

Merge de Listas Enlazadas

```
procedure agregar_atras (var pri, ult: lis_totales; dato:
    total_prod);
var nue: lis_totales;

Begin
  new (nue);
  nue^.sig := nil;
  nue^.dato := dato;
   if (pri = nil) then  {La lista esta vacía?}
           pri := nue;  {NUE es el primero}
  else
         ult^.sig := nue; { El que era ultimo pasa a tener como
    siguiente al nuevo ultimo}
   ult := nue;  {El ultimo es el que acabo de agregar}
end;
```

Resolver

- BuscarPrecioUnitario(prec, prod_actual, preciou);
  - Pensar modo de pasaje de la lista “prec”
  - Suponer que el producto buscado siempre existe.
- Pensar generalización a N listas de venta
  - ¿Estructura de ventas a usar?
  - ¿Cómo cambia el procedure mínimo?
  - ¿Cambios en el procedure Totalizar?