Taller de Programación

AGENDA

Concepto de Merge

Merge entre dos listas

Merge entre más de dos listas

Merge acumulador

MERGE - Características

La **operación de merge** consiste en generar una nueva estructura de datos (arreglos, listas) ordenada a partir de la mezcla de dos o más estructuras de datos previamente ordenadas.

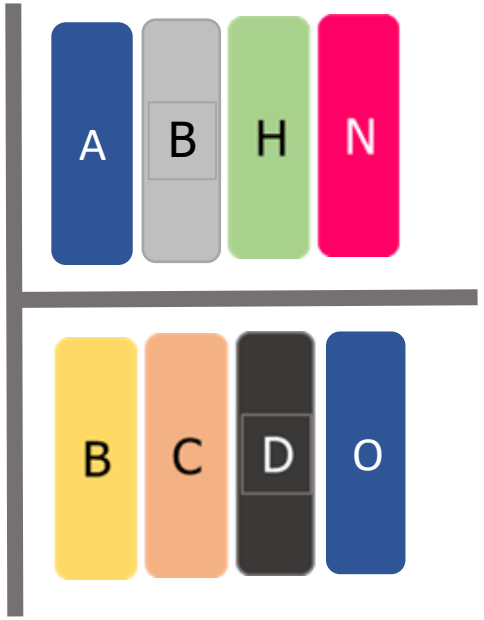
Las estructuras que se combinan guardan el mismo orden lógico interno (por ejemplo datos ordenados alfabéticamente)

Trabajaremos con un ejemplo de la vida real para luego asimilarlo e implementarlo

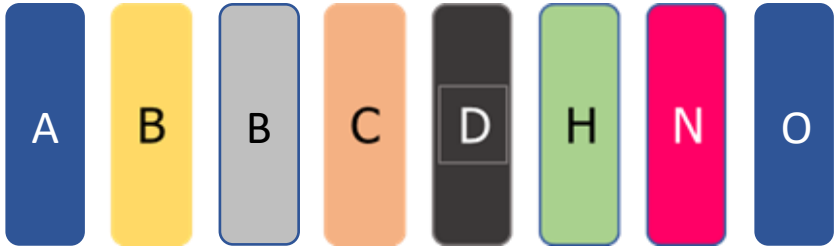
MERGE - Ejemplo

Juan y Paula viven juntos.

Juan y Paula tienen una biblioteca de dos estantes y en cada estante sus libros se encuentran ordenados alfabéticamente.

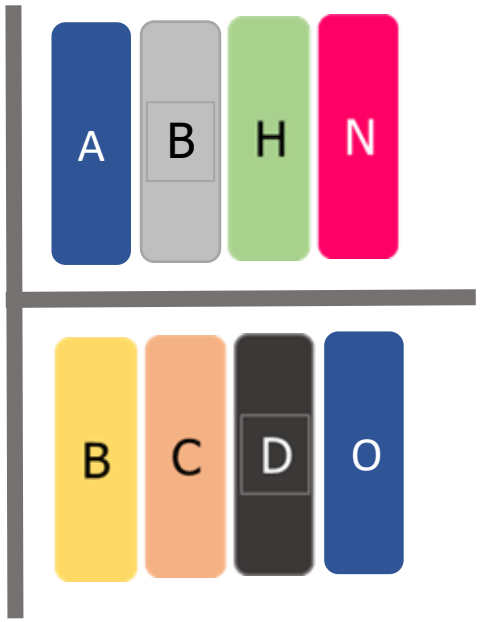


En algún momento deciden armar un único estante ordenado alfabéticamente.



MERGE - Ejemplo

Juan se encargará de acomodar los libros en el nuevo estante.



Paula le irá pasando los libros a Juan de **manera ordenada**.

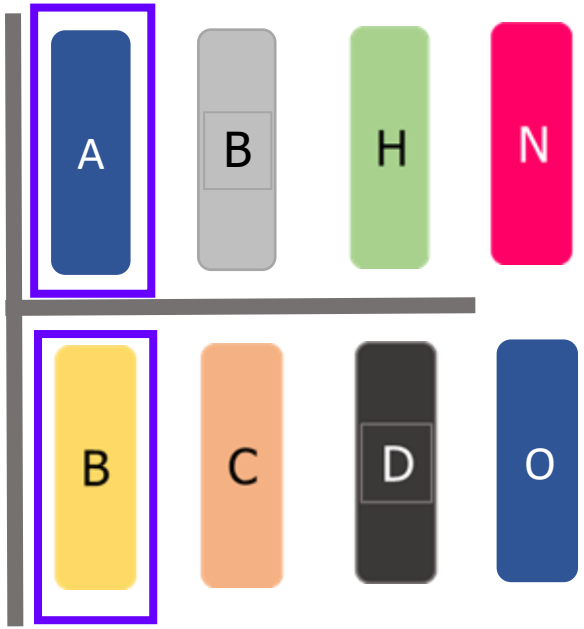
MERGE - Ejemplo

Comienzan a trabajar

Paula pásame un libro

Cómo lo hago de manera ordenada?

Elijo el menor entre el primero de cada estante



Calculo el mínimo

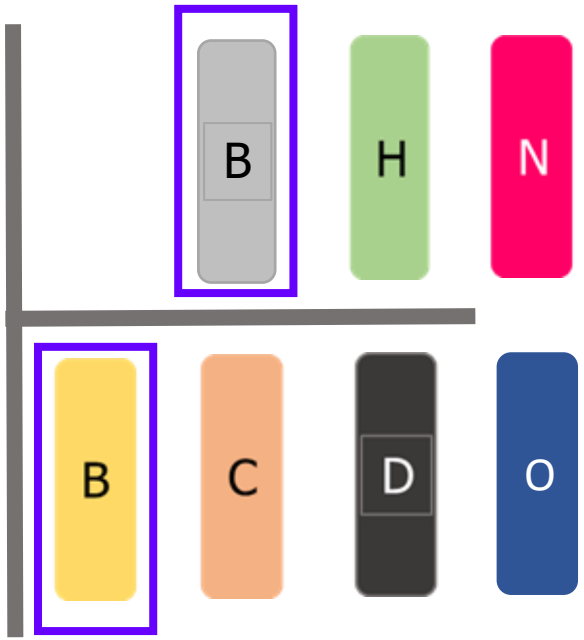
Por qué está bien que haga eso?

MERGE - Ejemplo

Arman el estante único

Paula pásame un libro

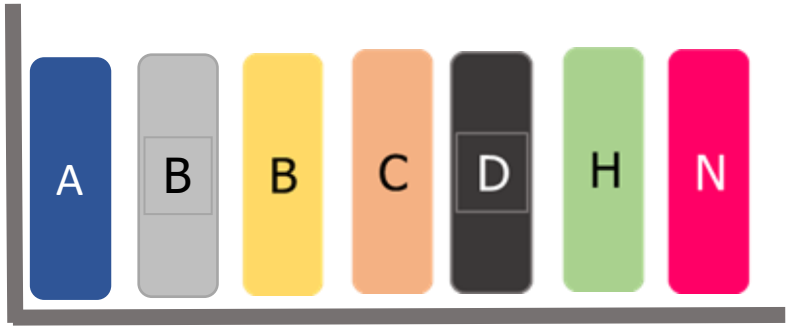
Solo debo ver el primero de cada estante



Calculo el mínimo

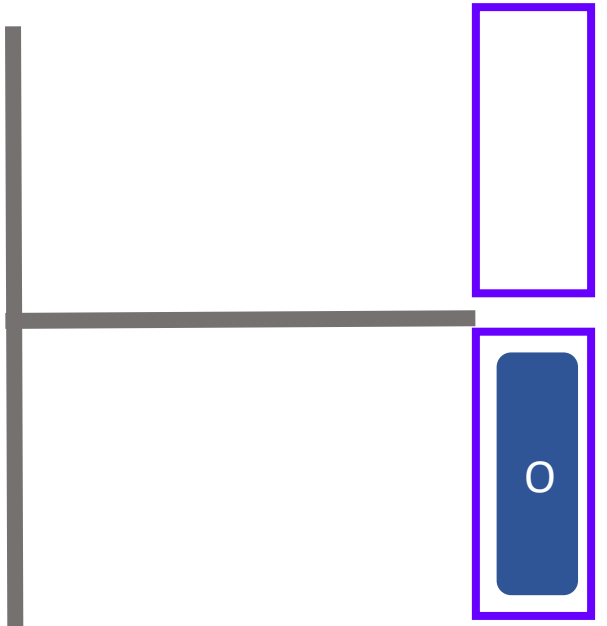
MERGE - Ejemplo

Arman el estate único



Paula pásame un libro

Solo debo ver el primero de cada estante

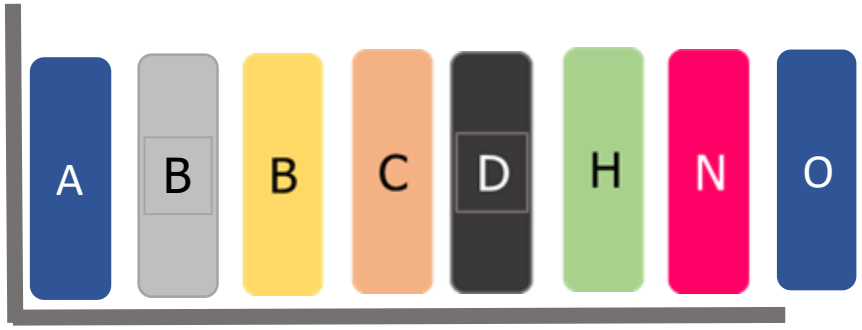


Calculo el mínimo

Qué pasa cuando un estante se termina?

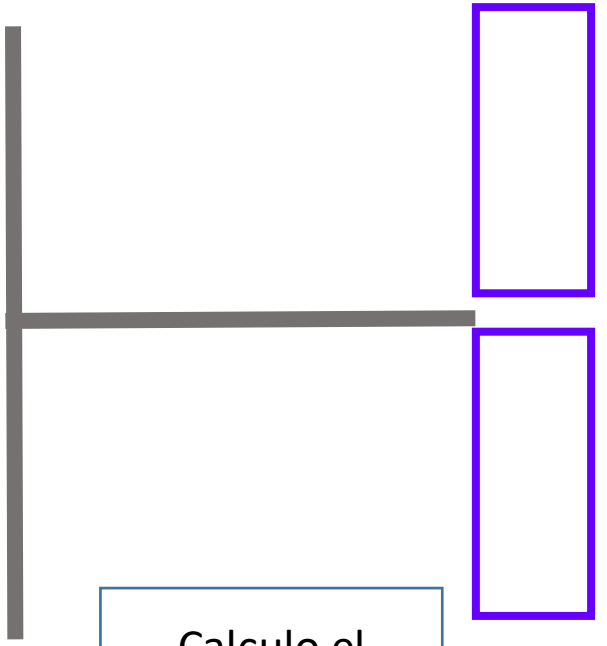
MERGE - Ejemplo

Arman el estante único



Paula pásame un libro

Solo debo ver el primero de cada estante



Calculo el mínimo

Se terminaron los libros.

Qué pasa cuando ambos estantes se terminan?

MERGE

Supongamos que se dispone de dos listas con los nombres de libros ordenadas alfabéticamente y se pide generar una única lista de libros ordenada alfabéticamente.

DEL EJEMPLO ANTERIOR

Libro ➡ Elemento de la Lista

Estante ➡ Lista

Estante Nuevo ➡ la lista a generar

```
Program merge;
Type
 lista =^nodo;
 nodo = record
  dato:string;
  sig:lista;
 end;

Var
 estante1, estante2: lista;
 estanteNuevo: lista;

Begin
 generarEstante (estante1);
 generarEstante (estante2);
 merge (estante1,estante2,estanteNuevo);
End.
```

MERGE

Supongamos que se dispone de dos listas con los nombres de libros ordenadas alfabéticamente y se pide generar una única lista de libros ordenada alfabéticamente.

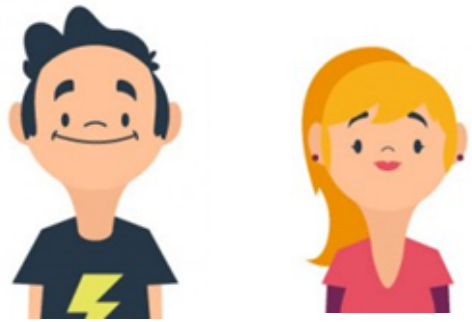
```
Program Estantes;
Type
 lista =^nodo;
 nodo = record
  dato:string;
  sig:lista;
 end;

Var
 estante1, estante2: lista;
 estanteNuevo: lista;

Begin
 generarEstante (estante1);
 generarEstante (estante2);
 merge (estante1,estante2,estanteNuevo);
End.
```

```
Procedure merge (e1,e2: lista; var eNuevo:lista);

Begin
  iniciliazar el estante eNuevo
  mientras (los estantes e1 y e2 tienen libros)do
    begin
     Paula busca el mínimo de los estantes e1 y e2
     Juan guarda el mínimo en el estante eNuevo
    end;
End;
```



MERGE

Supongamos que se dispone de dos listas con los nombres de libros ordenadas alfabéticamente y se pide generar una única lista de libros ordenada alfabéticamente.

```
Procedure merge (e1,e2: lista; var eNuevo:lista);

Begin
  iniciliazar el estante eNuevo
  mientras (los estantes e1 y e2 tienen libros)do
    begin
     Paula busca el mínimo de los estantes e1 y e2
     Juan guarda el mínimo en el estante eNuevo
    end;
End;
```

Qué significa que Juan guarde el libro?

Generar un espacio en el estante nuevo.

Agregar el libro que le da Paula al final del estante nuevo

Qué significa que Paula busque el mínimo?

Como las listas de libros están ordenadas alfabéticamente por título, sólo debe mirar el primer elemento de cada lista, compararlos y determinar el menor entre estos.

MERGE

Supongamos que se dispone de dos listas con los nombres de libros ordenadas alfabéticamente y se pide generar una única lista de libros ordenada alfabéticamente.

Qué significa que Paula busque el mínimo?

PaulaBuscaMinimo

```
libroMinimo :=  valor_muy_alto
if (estante_1 noVacío) and (estante_2 noVacío)then
        if (libroEnEstante_1 <= libroEnEstante_2 ) then
          libroMinimo:= libroEnEstante_1
        else
          libroMinimo:= libroEnEstante_2

else (estante_1 noVacío) and (estante_2 vacio) then
   libroMinimo:= libroEnEstante_1

else (estante_1 Vacío) and (estante_2 novacio) then
  libroMinimo:= libroEnEstante_2
```

Como las listas de libros están ordenadas alfabéticamente por título, sólo debe mirar el primer elemento de cada lista, compararlos y determinar el menor entre estos.

Qué devuelve si ambos estantes están vacíos?

MERGE

Supongamos que se dispone de dos listas con los nombres de libros ordenadas alfabéticamente y se pide generar una única lista de libros ordenada alfabéticamente.

```
Program merge;
Type
 lista =^nodo;
 nodo = record
  dato:string;
  sig:lista;
 end;

Var
 estante1, estante2: lista;
 estanteNuevo: lista;

Begin
 generarEstante (estante1);
 generarEstante (estante2);
 merge (estante1,estante2,estanteNuevo);
End.
```

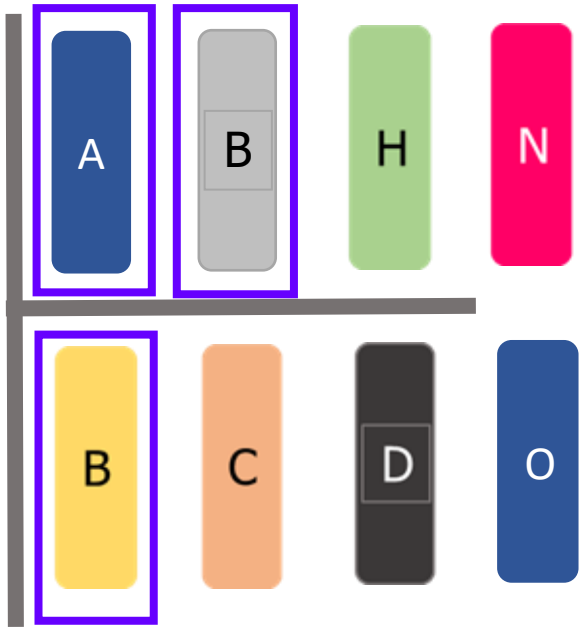
```
Procedure merge (E1,E2:lista; var Enuevo:lista);
Var
 min: string;
Begin
  Enuevo:= nil;
  minimo (E1,E2,min);
  while (min <> 'ZZZ') do
    begin
     agregarAtras (Enuevo,min);
     minimo (E1,E2,min);
    end;
End;
```

Por qué envío las listas al proc. mínimo?

Por qué invoco de nuevo al procedimiento mínimo?

MERGE

```
Procedure minimo(var e1,e2:lista; var min:string);
Begin
  min := 'ZZZ';
  if (e1 <> nil) and (e2 <> nil)then
       if (e1^.dato <= e2 ^.dato ) then
         begin
           min:= e1^.dato;
           e1:= e1 ^.sig;
         end;
       else begin
         min:= e2 ^.dato;
         e2:= e2 ^.sig;
       end
  else
    if (e1 <> nil) and (e2 = nil) then begin
      min:= e1^.dato;
      e1:= e1 ^.sig;
    end
    else
      if (e1 = nil) and (e2 <> nil) then begin
         min:= e2 ^.dato;
         e2:= e2 ^.sig;
      end;
```



Qué cambio si ahora en la casa hay 5 estantes?

MERGE

```
Program estantes;
cantE=5;
Type
 lista =^nodo;
 nodo = record
  dato:string;
  sig:lista;
 end;
 estantes = array[1..cantE] of lista;

Var
 todos: estantes;
 estanteNuevo: lista;

Begin
 generarEstantes (todos);
 merge (todos,estanteNuevo);
End.
```

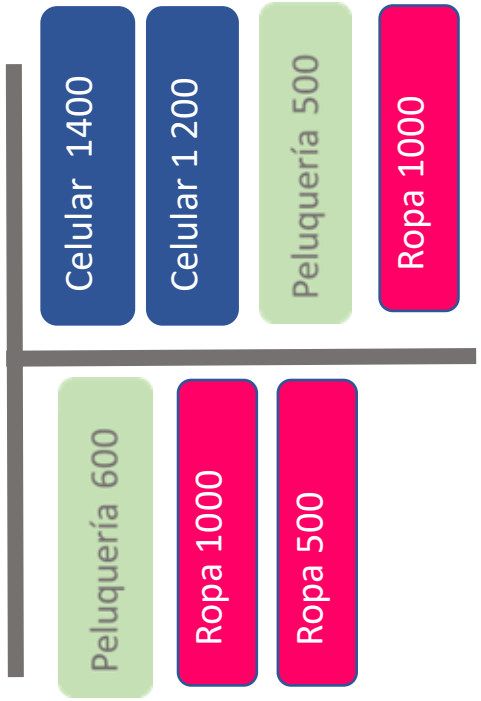
```
Procedure merge (todos:estantes; var Enuevo:lista);

Begin
  Enuevo:= nil;
  minimo (todos,min);
  while (min <> 'ZZZ') do
    begin
     agregarAtras (Enuevo,min);
     minimo (todos,min);
    end;
End;
```

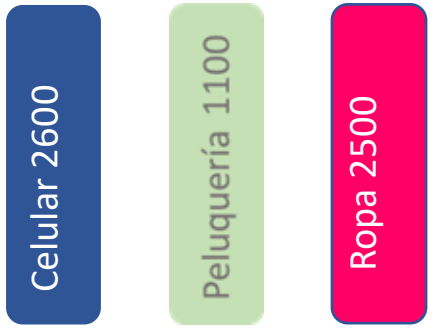
Sólo debo modificar el procedimiento mínimo

MERGE - ACUMULADOR

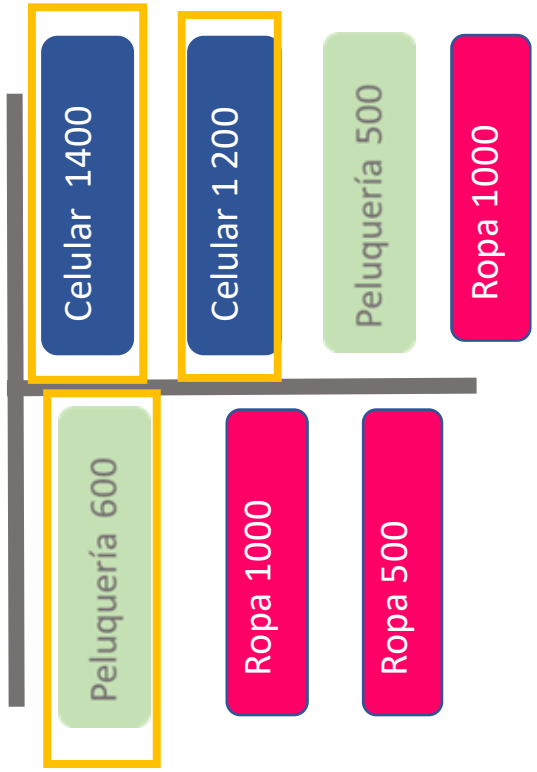
Juan y Paula tienen los gastos de cada uno ordenados por nombre de gasto.

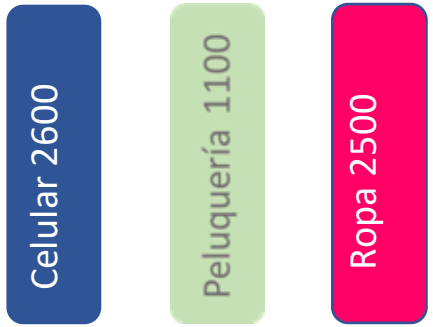


Supongamos que quieren armar un listado ordenado por gasto con el total entre ambos de cada tipo de gasto.



MERGE - ACUMULADOR






```
El mínimo es celular, monto 1400
El mínimo es celular, monto 1200
El mínimo es peluquería, monto 500
```

MERGE

```
Program estantes;
cantE=2;
Type
 gasto= record
  nombre:string;
  monto:real;
 end;
 lista =^nodo;
 nodo = record
  dato:gasto;
  sig:lista;
 end;
 estantes = array[1..cantE] of lista;

Var
 todos: estantes;
 estanteNuevo: lista;

Begin
 generarEstantes (todos);
 merge (todos,estanteNuevo);
End.
```

```
Procedure merge (todos:estantes; var Enuevo:lista);

Begin
  Enuevo:= nil;
  minimo (todos,min);
  while (min <> 'ZZZ') do
    begin
     agregarAtras (Enuevo,min);
     minimo (todos,min);
    end;
End;
```

Sólo debo modificar el procedimiento merge

MERGE

```
Procedure merge (todos:estantes; var Enuevo:lista);
Begin
  Enuevo:= nil;
  minimo (todos,minNombre, monto);
  while (minNombre <> ‘ZZZ’) do
    begin
     actual:= minNombre; montoTotal:=0;
     while ((minNombre <> ‘ZZZ’) and (minNombre = actual) )do
     begin
        montoTotal:= montoTotal + monto;
        minimo (todos,minNombre,monto);
     end;
     agregarAtras(Enuevo,actual,montoTotal);
    end;
End;
```

MERGE

```
Procedure minimo (var todos:estantes; var nomMin:string; var monto:real);

Var
 indiceMin,i:integer;

Begin
  nomMin:= 'ZZZ';
  for i:= 1 to cantE do
     if (todos[i] <> nil) then
       if (todos[i]^.dato.nombre <= nomMin) then begin
                                                 indiceMin:= i;
                                                 nomMin:= todos[i]^.dato.nombre;
                                                end;
  if (nomMin <> 'ZZZ') then
    begin
      //nomMin:= todos[indiceMin]^.dato.nombre;
      monto:= todos[indiceMin] ^.dato.monto;
      todos[indiceMin]:= todos[indiceMin]^sig;
    end;
End;
```