Inscripción Grupos
Régimen de cursada y promoción
Ejercicio de Repaso

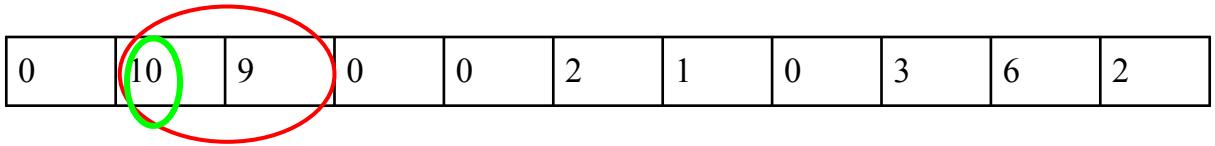
Se tiene un arreglo de enteros positivos (de longitud Max) que forman secuencias, separadas por uno o más ceros. Se desea buscar el mayor número de cada secuencia e insertarlo en otro Arreglo MAYORES(también de longitud Max) en el orden en que se van detectando. El arreglo puede ó no empezar y terminar con ceros. Realice el Diagrama de Estructura y codifique la solución. No se puede utilizar estructuras auxiliares. Considere que el arreglo está cargado hasta Max y el arreglo MAYORES está cargado con un valor discernibles(-1) hasta Max

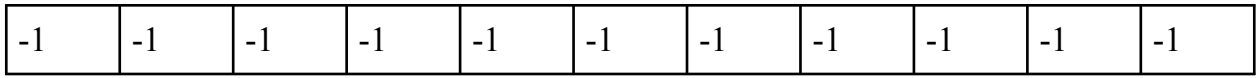
Ejemplo, suponiendo esta carga del arreglo:

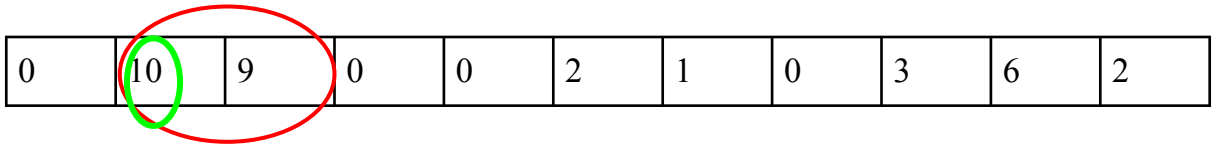
0	10	9	0	0	2	1	0	3	6	2

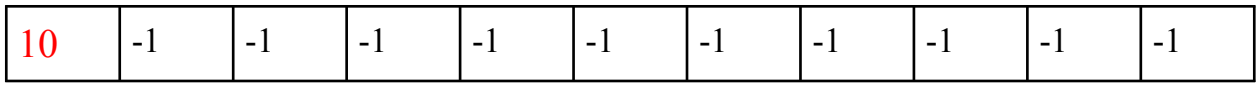
la salida, sera MAYORES:

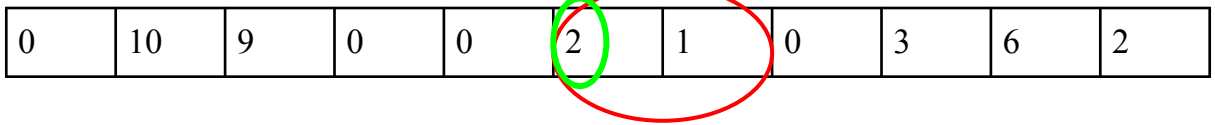
10	2	6	-1	-1	-1	-1	-1	-1	-1	-1

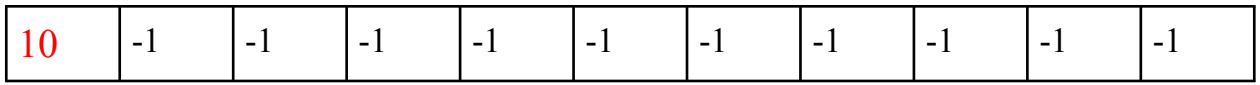

0
10
9
0
0
2
1
0
3
6
2

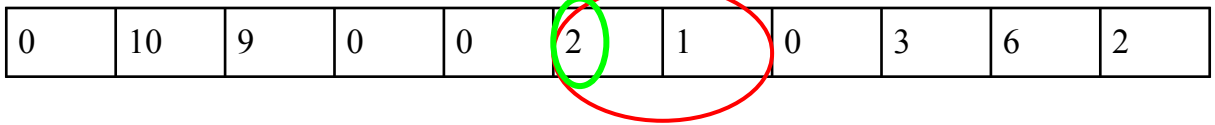

-1
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1

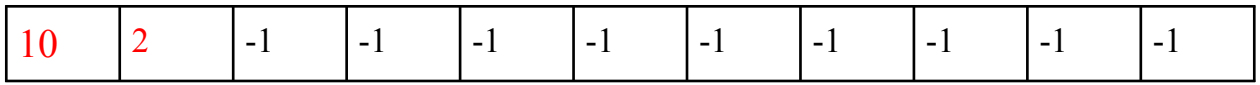

0
10
9
0
0
2
1
0
3
6
2

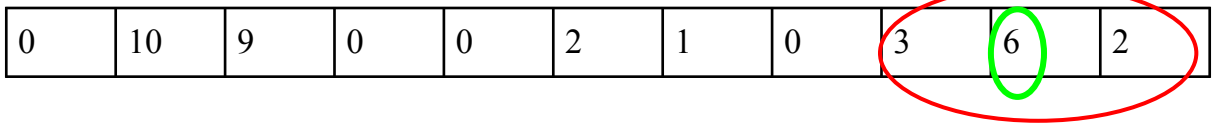

10
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1

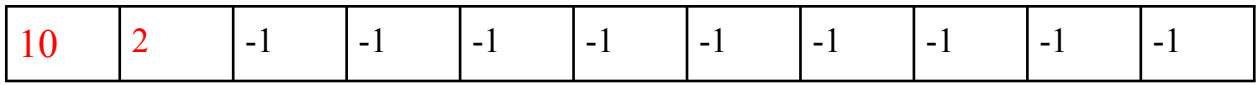

0
10
9
0
0
2
1
0
3
6
2

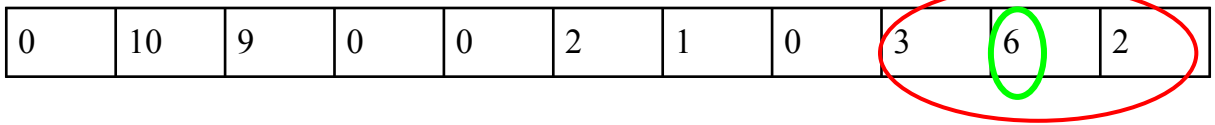

10
-1
-1
-1
-1
-1
-1
-1
-1
-1
-1

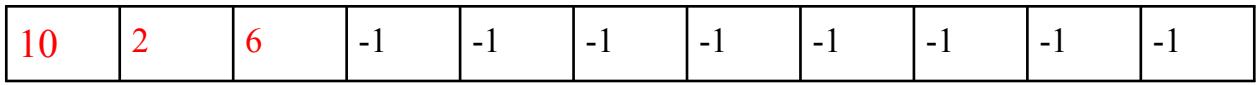

0
10
9
0
0
2
1
0
3
6
2

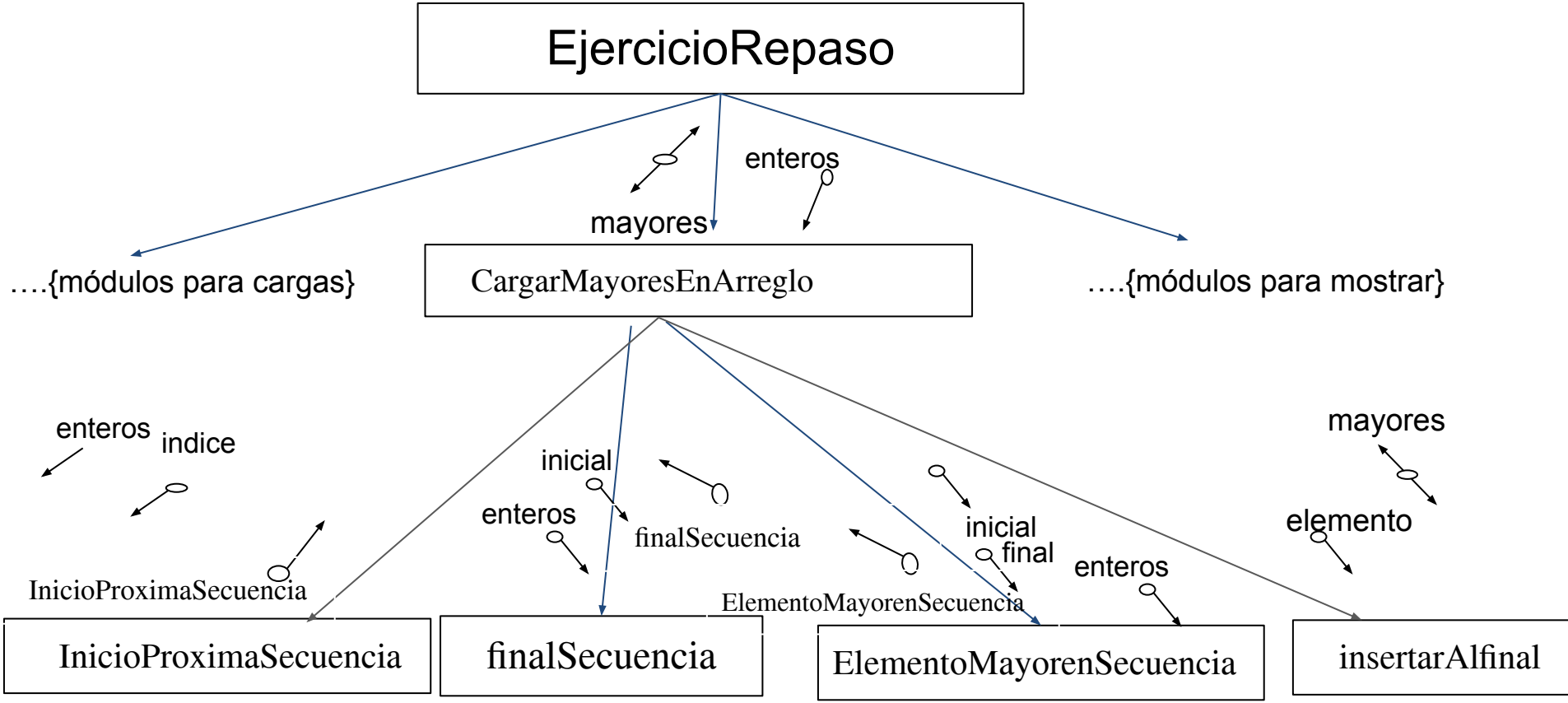

10
2
-1
-1
-1
-1
-1
-1
-1
-1
-1


0
10
9
0
0
2
1
0
3
6
2


10
2
-1
-1
-1
-1
-1
-1
-1
-1
-1


0
10
9
0
0
2
1
0
3
6
2


10
2
6
-1
-1
-1
-1
-1
-1
-1
-1


EjercicioRepaso
mayores
enteros
….{módulos para cargas}
CargarMayoresEnArreglo
….{módulos para mostrar}
enteros
indice
inicial
enteros
finalSecuencia
inicial
final
enteros
mayores
elemento
InicioProximaSecuencia
ElementoMayorenSecuencia
InicioProximaSecuencia
finalSecuencia
ElementoMayorenSecuencia
insertarAlfinal

```
program EjercicioRepaso;

const
   Min=1;
   Max=8;
   dis=-1;

type
   ArregloEnt = Array[Min..Max] of Integer;
```

```
……{módulos a HACER }……

var
   ENTEROS, mayores:ArregloEnt;

begin
   writeln('Ingresar secuencias separadas por ceros');
   writeln('Se considera arreglo completo ');
   Cargar(ENTEROS); {NO  HACER }
   CargarDiscernibles(Mayores); {NO  HACER }
   CargarMayoresEnArreglo(ENTEROS,Mayores); {  HACER }
   writeln('luego de la carga el arreglo quedo: ');
   mostrar(Mayores); {NO  HACER }
end.
```

```
Procedure Cargar(var arreglo:ArregloEnt);
var i:Integer;
begin
   For i:= Min to Max do
      begin
       writeln('Ingrese valor en la posición: ', i );
       readln(arreglo[i]);
       end;
end;

Procedure CargarDiscernibles(var arreglo:ArregloEnt);
var i:Integer;
begin
   For i:= Min to Max do arreglo[i]:= dis;
end;
```

```
procedure CargarMayoresEnArreglo(ENTEROS: ArregloEnt; var Mayores:ArregloEnt);
var
  inicio, final: integer;

begin
  inicio:= InicioProximaSecuencia(ENTEROS, Min);
  while  (inicio<=Max) do
   begin
        final:= FinalSecuencia (enteros, inicio);
        insertarAlfinal ( Mayores, ElementoMayorenSecuencia(ENTEROS, inicio,final));
        inicio:= InicioProximaSecuencia(ENTEROS, final+1);
  end;
end;
```

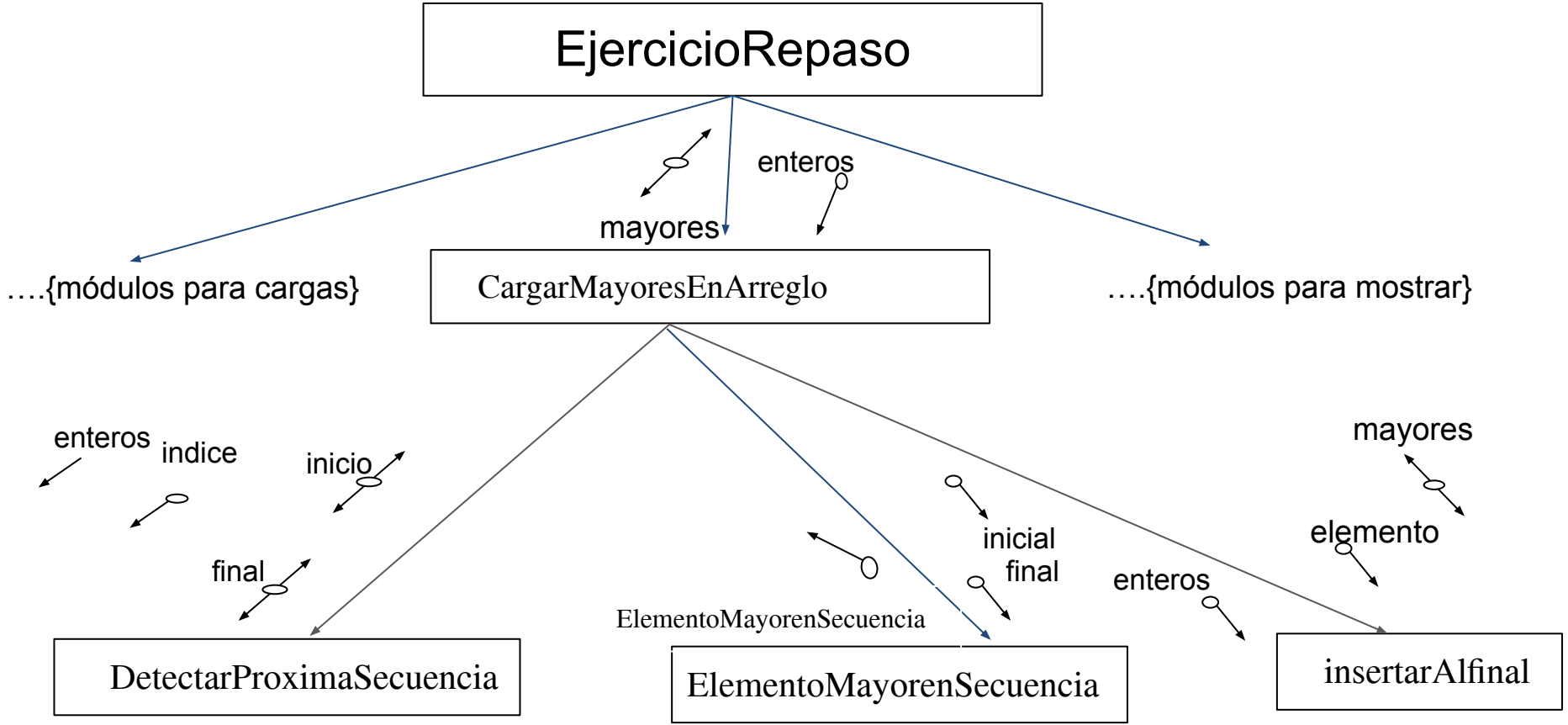
```
function InicioProximaSecuencia(arreglo:ArregloEnt; i:integer):integer;
begin
   while (i<=Max) and (arreglo[i]=0) do i+=1;
   InicioProximaSecuencia:= i;  // si i > max se indico que no hay mas secuencia
end;


function finalSecuencia(arreglo:ArregloEnt; i:integer):integer;
begin
   while (i<=Max) and (arreglo[i]<>0) do i:=i+1;
   finalSecuencia:=i-1;
end;
```

```
function ElementoMayorenSecuencia(arreglo:ArregloEnt ; inicio,final:Integer):integer;
  var i,Mayor:Integer;
  begin
     Mayor:= arreglo[inicio];
     for i:=inicio+1 to final do
        if(arreglo[i]>= Mayor) then  Mayor:=arreglo[i];  // si hay dos iguales se queda con   el último
     ElementoMayorenSecuencia:= Mayor
  end;
```

```
procedure insertarAlfinal (var Mayores:ArregloEnt; Elemento: integer);
var
   i: integer;
begin
   i:= Min;
   while Mayores[i]<>dis do i:= i +1; // en este ejercicio no se llega
// nunca al final porque ambos arreglos igual dimensión y Mayores va a tener menos elementos
   Mayores[i]:= Elemento;
end;
```

variante en la Solución

En vez de tener dos funciones para detectar la secuencia, se podría pensar como un único procedure que detecta el inicio y fin de una secuencia


EjercicioRepaso
mayores
enteros
….{módulos para cargas}
CargarMayoresEnArreglo
….{módulos para mostrar}
enteros
indice
inicio
final
DetectarProximaSecuencia
ElementoMayorenSecuencia
inicial
final
ElementoMayorenSecuencia
mayores
elemento
enteros
insertarAlfinal

Variante

```
Procedure DetectarProximaSecuencia(arreglo:ArregloEnt; indice:integer; var inicio,final:Integer);
begin
  while (indice <= MAX)   and (arreglo[indice]=  0) do indice:= indice + 1;
  if indice <= MAX
    then
      begin
              inicio:= indice;
              while (indice <= MAX)  and (arreglo[indice]<> 0) do indice:= indice + 1;
              final:= indice-1;
       end
    else
      final:= indice;
end;
```

```
procedure CargarMayoresEnArreglo(ENTEROS: ArregloEnt; var Mayores:ArregloEnt);
var
   inicio, final: integer;
begin
inicio:= 0;
final:= 0;
DetectarProximaSecuencia(arreglo, min, inicio, final);
while  (final<=Max) do
   begin
        insertarAlfinal ( Mayores, ElementoMayorenSecuencia(ENTEROS, inicio,final));
       DetectarProximaSecuencia(arreglo, final+1, inicio, final);
  end;
end;
```

¿Qué cambios harían a la solución si MAYORES se debe ir creando ordenado?