Estructura de datos VECTOR

TEMAS de la CLASE

1. Operaciones frecuentes en el tipo Vector
   - Búsqueda
   - Borrar un elemento determinado
   - Insertar un elemento en un vector con orden

2. Ejercitación

Tipo vector: Operación de Búsqueda

El proceso de ubicar información particular en una colección de datos es conocido como método de búsqueda.

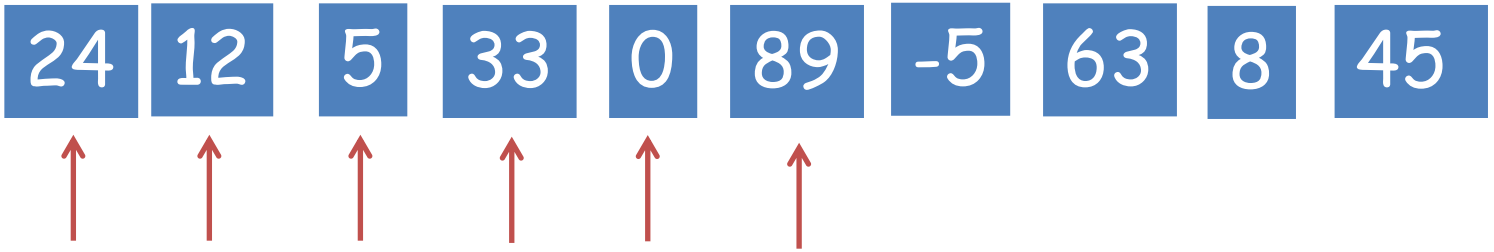
Se deben considerar los siguientes casos:

- Los datos en el vector están almacenados sin ningún orden.

- Los datos en el vector están almacenados ordenados por algún criterio

Tipo Vector: Búsqueda Lineal o secuencial (elementos sin orden)

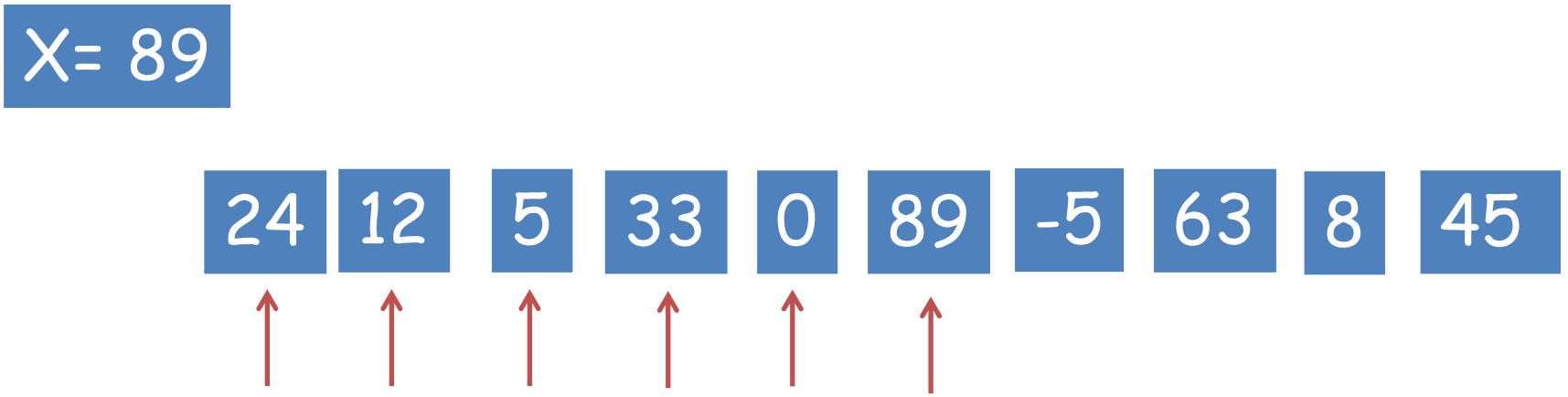
X= 89



- La búsqueda comienza desde el principio y se avanza por la estructura de manera secuencial, uno a uno.

- La solución debería recorrer el vector y detenerse en caso de encontrar el elemento X.

Tipo Vector: Búsqueda Lineal o secuencial (elementos sin orden)




```
Buscar (Recibe el vector donde buscar, el elemento a buscar,
        la dimensión lógica  y devuelve la posición donde se encontró)

Ubicarse al principio del vector
Mientras (no llegue al final del vector) y (no encuentre el elemento)
     avanzar una posición en el vector

Al salir del mientras se debe evaluar por cual de las condiciones
finalizó
```

Operación de Búsqueda Lineal o secuencial (elementos sin orden)

Consideremos la siguiente declaración genérica:

```
Const
    DimF = ...      {máxima longitud del vector}
Type
    TipoElem = ... {tipo de datos del vector }
    Indice  =  0.. DimF;
    vector =  Array  [ 1..DimF] of TipoElem;
```

entos sin orden)

```
Buscar (Recibe el vector donde buscar, el elemento a buscar,
        la dimensión lógica  y devuelve la posición donde se encontró)
Ubicarse al principio del vector
Mientras (no llegue al final del vector) y (no encuentre el elemento)
     avanzar una posición en el vector
Al salir del mientras se debe evaluar por cual de las condiciones finalizó
```

```
Const
    DimF = ...
Type
    Indice = 0..DimF;
    vector= Array  [1..DimF] of integer;
```

¿Qué tipo de módulo Conviene utilizar?

```
Function BuscarPosElem (x:integer; v:vector; dimL: Indice): Indice;
var pos:Indice; exito: boolean;
Begin
 pos:=1;
 exito:= false;
 while (pos <= dimL) and (not exito) do
      if (x = v[pos]) then exito:= true
                      else pos:=pos+1;
 if (exito = false) then pos:=0;
 BuscarPosElem := pos;
end;
```

¿Qué valor toma pos cuando el elemento no está?

Características de la Búsqueda Lineal o Secuencial

- Se aplica cuando los elementos no tienen orden.

- Requiere excesivo consumo de tiempo en la localización del elemento.

- Número medio de comparaciones (dimL + 1) /2

- Es ineficiente a medida que el tamaño del arreglo crece.

Tipo Vector: Borrar un elemento

Recordemos que la operación de Borrar un elemento en un vector admite dos posibilidades:

1 Borrar un elemento de una posición determinada

Ya lo vimos

2 Borrar un elemento determinado del vector

Tipo Vector: Borrar un elemento determinado

Esta operación requiere primero buscar el elemento y luego borrarlo

```
Buscar la posición del elemento a borrar
Si el elemento está entonces
                    Borrar el elemento
```

```
Const
    DimF = ...
Type
 Indice  =  0.. DimF;
 vector =  Array  [ 1..DimF] of integer;
```

```
Procedure BorrarElem (var v: vector;  var dimL: indice;
                      elem : integer;  var exito: boolean);
var pos: indice;

begin
  exito:= false;
  pos:= BuscarPosElem (elem, v, dimL);
  if pos <> 0 then begin
                    BorrarPosModif (v, dimL, pos);
                    exito:= true;
                   end;
end;
```

Ya lo vimos!!

Revisar BorrarElemPos y adaptarlo!!

Tipo Vector: Borrar un elemento determinado

```
Procedure BorrarElem (var v: vector;  var dimL: indice;
                      elem : integer;  var exito: boolean);
```

```
Function BuscarPosElem (x:integer;v:vector;dimL: Indice) : Indice;
var pos:Indice; exito: boolean;
Begin
 pos:=1; exito:= false;
 while (pos <= dimL) and (not exito) do
      if (x = v[pos]) then exito:= true
                      else pos:=pos+1;
 if (exito = false) then pos:=0;
 BuscarPosElem := pos;
end;
```

Ya lo vimos!!

```
Procedure BorrarPosModif (var v:vector; var dimL:integer; pos:Indice);
 begin

 end;
```

```
var pos: indice;
Begin
  exito:= false;
  pos:= BuscarPosElem (elem, v, dimL);
  if pos <> 0 then begin
                    BorrarPosModif (v, dimL, pos);
                    exito:= true;
                  end;
end;
```

Tipo Vector: Borrar un elemento determinado

```
Procedure BorrarPos (var v:vector; var dimL:integer; pos:Indice;
                                  var exito:boolean);
 var i: integer;
 begin
   exito := false;
   if (pos >=1 and pos <= dimL)
   then begin
          exito := true
          for j:= pos + 1  to   dimL  do
             v [ i - 1 ]  :=  v [ i ] ;
          dimL := dimL - 1 ;
        end;
 end;
```

Ya lo vimos!!

```
Procedure BorrarPosModif (var v:vector; var dimL:integer; pos:Indice);
 var i: integer;
 begin
    for i:= pos + 1  to   dimL  do
       v [ i - 1 ]  :=  v [ i] ;
    dimL := dimL - 1 ;
 end;
```

BorrarPos adaptado

Tipo Vector: Borrar un elemento determinado

```
Procedure BorrarElem (var v: vector;  var dimL: indice;
                      elem : integer;  var exito: boolean);
 Function BuscarPosElem (x:integer;v:vector;dimL: Indice) : Indice;
 var pos:Indice; exito: boolean;
 Begin
  pos:=1; exito:= false;
  while (pos <= dimL) and (not exito) do
       if (x = v[pos]) then exito:= true
                       else pos:=pos+1;
  if (exito = false) then pos:=0;
  BuscarPosElem := pos;
 end;
 Procedure BorrarPosModif (var v:vector; var dimL:integer; pos:Indice);
  var i: integer;
  begin
     for i:= pos + 1  to   dimL  do
        v [ i - 1 ]  :=  v [ i] ;
     dimL := dimL - 1 ;
  end;
var pos: indice;
Begin
   exito:= False;
   pos:= BuscarPosElem (elem, v, dimL);
   if pos <> 0 then begin
                    BorrarPosModif (v, dimL, pos);
                    exito:= true;
                end;
end;
```

Tipo Vector: Búsqueda en Arreglos Ordenados

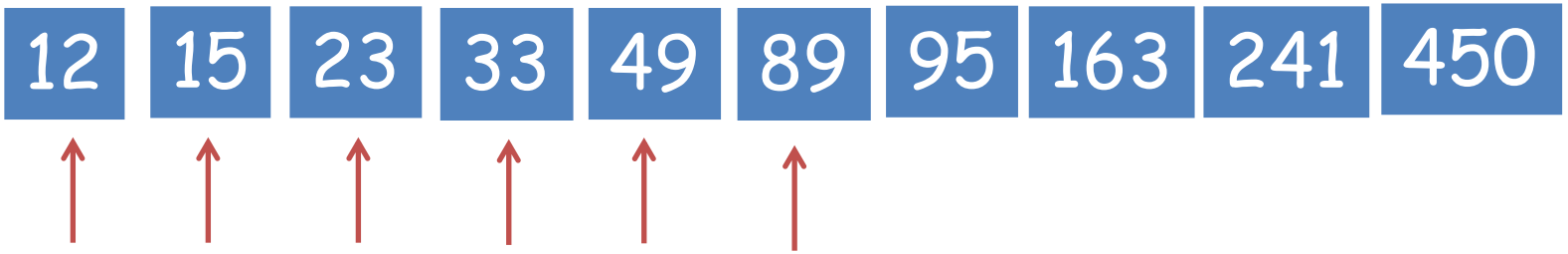
Métodos de Búsqueda

Método 1
Secuencial optimizado:
Se recorre el vector hasta encontrar el número buscado o hasta encontrar uno mayor que él.

Método 2
Binaria o Dicotómica:
Acceder a los elementos del vector de una manera mas “eficiente”…

Tipo vector: Búsqueda en Arreglos Ordenados
Método 1: Secuencial Optimizado

X= 89

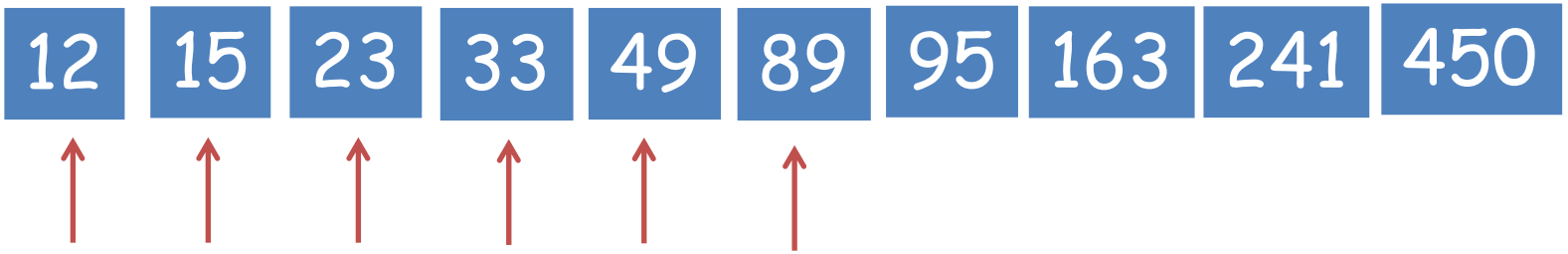


- Se aplica cuando los elementos tienen orden.

- La búsqueda comienza desde el principio y se avanza por la estructura de manera secuencial y de a uno hasta que encuentro el número buscado o hasta que encuentro uno mayor.

Tipo vector: Búsqueda en Arreglos Ordenados
Método 1: Secuencial Optimizado

X= 89




```
Módulo Buscar (el elemento a buscar, el vector donde buscar,
               la dimensión lógica  y devuelve la posición donde se encontró)

Ubicarse al principio del vector
Mientras (no llegue al final del vector) y
         (el elemento a buscar sea mayor que el elemento observado)
  avanzar una posición en el vector

Al salir del mientras se debe evaluar por cual de las condiciones finalizó.
```

Tipo vector: Búsqueda en Arreglos Ordenados

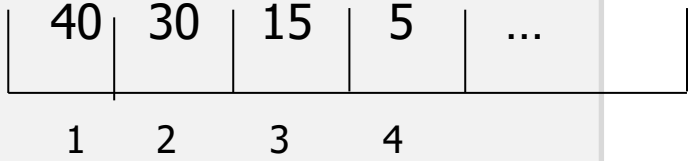
```
Módulo Buscar (el elemento a buscar, el vector donde buscar,
               la dimensión lógica  y devuelve la posición donde se encontró)

Ubicarse al principio del vector
Mientras (no llegue al final del vector) y
         (el elemento a buscar sea mayor que el elemento observado)
  avanzar una posición en el vector
Al salir del mientras se debe evaluar por cual de las condiciones finalizó
```

```
Const
  DimF=...
Type
 Indice   =  0.. DimF;
 vector =  Array  [ 1..DimF] of integer;
```

Vector ordenado de menor a mayor

```
Function BuscoPosElemOrd (x: integer; v:Vector; dimL: Indice): Indice;
  var pos : Indice;
  begin
    pos:=1;
    while (pos <= dimL) and (x > v[pos]) do
              pos:=pos+1;
    if ( pos > dimL ) or (x < v [pos]) then pos:=0;
    BuscoPosElemOrd:= pos;
  end;
```



x->45
dimL->4
Pos->1->2->3->4->5->0

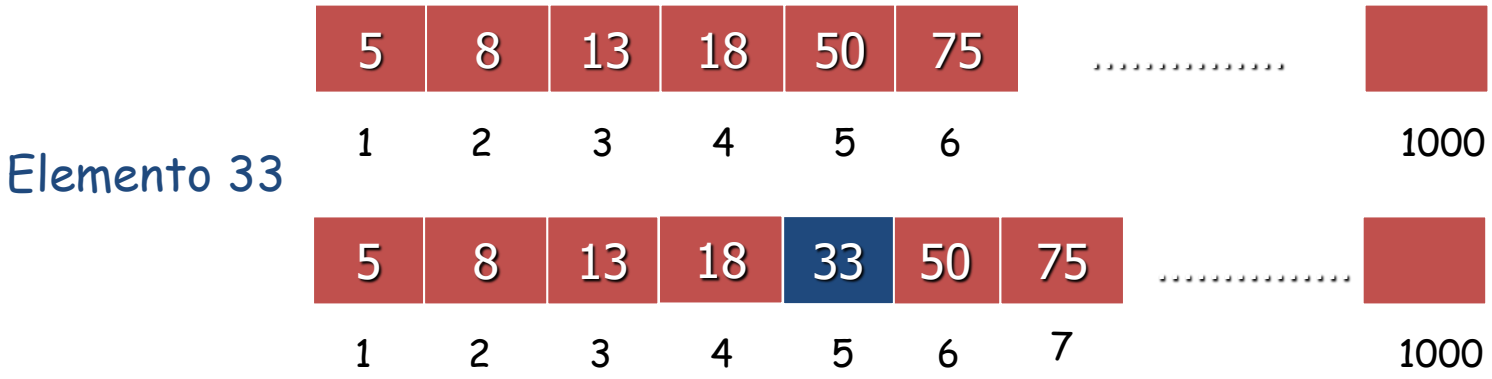
Tipo Vector: Insertar un elemento

Recordemos que la operación de Insertar un elemento en un vector admite dos posibilidades:

1 Insertar un elemento en una posición determinada

Ya lo vimos

2 Insertar un elemento manteniendo un orden determinado



Tipo Vector: Insertar un elemento en un vector ordenado

- Esta operación requiere verificar el espacio disponible, buscar la posición correspondiente manteniendo el orden y luego insertar el elemento en el vector

Verificar espacio en el vector

Determinar posición donde se insertara

Insertar elemento en la posición determinada

```
Const
  DimF = ...
Type
   Indice  =  0.. DimF;
   vector =  Array  [ 1..DimF] of integer;
```

```
Procedure InsertarElemOrd (var v: vector;  var dimL: indice;
                           elem : TipoElem; var exito: boolean);

var pos: indice;
Begin
   exito := false;
   if (dimL < dimF)
   then begin
          pos:= DeterminarPosicion (elem, v, dimL);
          Insertar (v, dimL, pos, elem);
          exito := true;
        end;
end;
```

Revisar BuscoPosElemOrd y adaptarlo!!

Revisar InsertarPos y adaptar

Adaptamos "BuscoPosElemOrd" para escribir "BuscarPosicion"

```
Const
  DimF = ...
Type
  Indice = 0.. DimF;
  vector = Array [1..DimF] of integer;
```

```
Function BuscoPosElemOrd (x: integer; v:Vector; dimL: Indice): Indice;
  var pos : Indice;
  begin
    pos:=1;
    while (pos <= dimL) and (x > v[pos]) do
            pos:=pos+1;
    if ( pos > dimL ) or (x < v [pos]) then pos:=0;
    BuscoPosElemOrd:= pos;
  end;
```

Ya lo vimos!!

```
Function DeterminarPosicion ( x: integer;  v:Vector; dimL: Indice): Indice;
    var pos : Indice;
   begin
     pos:=1;
     while (pos<=dimL) and (x > v[pos]) do
        pos:=pos+1;
     DeterminarPosicion:= pos;
   end;
```

BuscoPosElemOrd adaptado

Adaptamos "InsertarPos" para escribir "Insertar"

```
Const
  DimF = ...
Type
  Indice = 0.. DimF;
   vector = Array [1..DimF] of integer;
```

```
Procedure INSERTARPOS(var v: vector; var dimL: integer; elemento: integer;
                      pos: integer; var exito: boolean );

var i : integer;
Begin
  exito := false;
  if  (dimF > dimL) and ((pos>=1) and (pos<= dimL))
          then begin
             exito := true;
             for  i := dimL downto pos do
                 v [ i + 1 ] := v [ i ] ;
              v [pos] := elemento;
              dimL := dimL + 1;
          end;
end;
```

Ya lo vimos!!

```
Procedure Insertar (var v:vector; var dimL:Indice; pos: Indice; elem:integer);
    var j: indice;
    begin
        for j:= dimL downto pos do
            v [ j +1 ] := v [ j ] ;
        v [ pos ] := elem;
        dimL := dimL + 1;
    End;
```

InsertarPos adaptado

Tipo Vector: Insertar un elemento en un vector ordenado

```
Procedure InsertarElemOrd (var v: vector;  var dimL: indice; elem : TipoElem;
                                   var exito: boolean);
```

```
Function DeterminarPosicion ( x: integer;  v:Vector; dimL: Indice): Indice;
  var pos : Indice;
   begin
     pos:=1;
      while (pos<=dimL) and (x > v[pos]) do
           pos:=pos+1;
     DeterminarPosicion:= pos;
   end;
```

Nuevo!!!

```
Procedure Insertar (var v:vector; var dimL:Indice; pos: Indice; elem:integer);
   var j: indice;
   begin
     for j:= dimL downto pos do
          v [ j +1 ] := v [ j ] ;
     v [ pos ] := elem;
     dimL := dimL + 1;
   End;
```

Nuevo!!!

```
 var pos: indice;

Begin
    exito := false;
    if (dimL < dimF) then begin
                              pos:= DeterminarPosicion (elem, v, dimL);
                              Insertar (v, dimL, pos, elem);
                              exito := true;
                         end;
end;
```

Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

- Se aplica cuando los elementos tienen orden.

- Se compara el valor buscado (x) con el ubicado en el medio del vector (a):

  - Si el elemento ubicado al medio del vector es igual a x, entonces la búsqueda termina.

  - Si no es el valor buscado, debería quedarse con la mitad del vector que conviene, para seguir la búsqueda. Este paso se repite tantas veces hasta que se acaba el vector o encuentro el valor.

Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

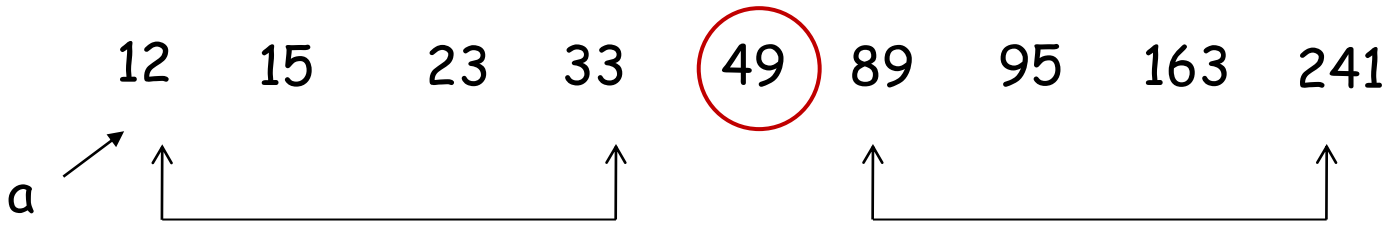
Primera vez

Elemento buscado X= 89

➡ Se calcula la posición del medio del vector original

Si pri=1

Ult=9



a[medio]= a[5] = 49

Dado que 89 > 49, se trabajará con el "subvector" del medio al final

Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

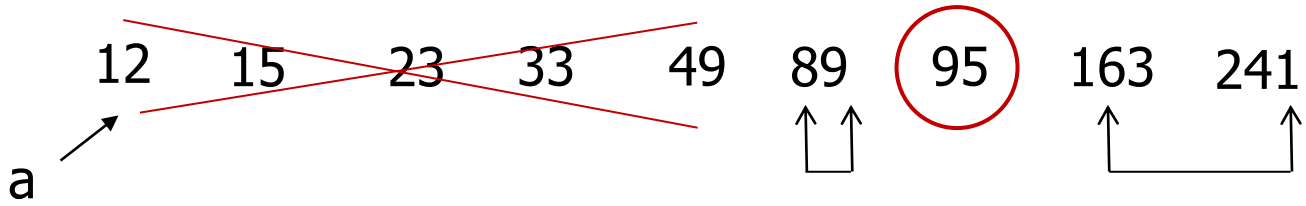
Segunda vez

Elemento buscado X= 89

- Se descarta la primera parte
- Se calcula la posición del medio del "subarreglo" delimitado por:

si Pri=6

Ult=9



a[medio]= a[7] = 95

Dado que 89 < 95, trabajo con el "subvector" del principio al medio

Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

Tercera vez

Elemento buscado X= 89

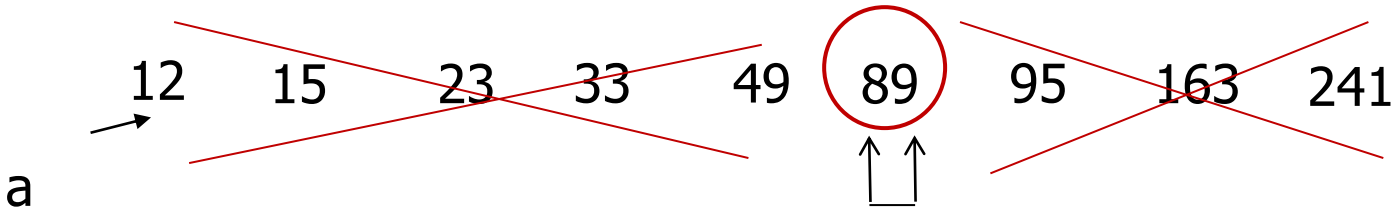
➡ Se descarta la "segunda" parte del "subarreglo" (de 7 a 9)

➡ Se calcula la posición del medio del "subarreglo" delimitado por:

Pri=6

si

Ult=6



a[medio]= a[6] = 89

89 = 89 se encontró el elemento!!!

Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

Observaciones:

- Cada vez que se toma la mitad del arreglo, se va disminuyendo el tamaño del mismo.

- El proceso termina cuando encuentro el elemento, o cuando el vector se hace tan pequeño que no quedan mas elementos, y por lo tanto se puede deducir que el elemento no se encuentra en el vector.

Tipo vector: Búsqueda en Arreglos Ordenados

```
Procedure  BusquedaBin ( var v: Vector; var j: Indice;
                                  dimL: Indice, x : TipoElem) ;
  Var pri, ult, medio :  Indice ;
  Begin
    j :=0 ;
    pri:= 1 ;
    ult:= dimL;
    medio := (pri + ult ) div 2 ;
    While  ( pri < = ult ) and ( x <> v [medio]) do begin
          If ( x < v [ medio ] ) then ult:= medio -1
                                  else pri:= medio+1 ;
          medio := ( pri + ult ) div 2 ;
    end ;
    If pri < = ult then  j := medio
                    else   j := 0 ;
  End ;
```

Calcula la posición del medio del vector

Se queda con la primera mitad

Se queda con la segunda mitad

Recalcula la posición del medio del "subvector"

Características de la Búsqueda Dicotómica

- Se aplica cuando los elementos tienen orden. Caso contrario debería ordenarse el vector previamente.

- Número medio de comparaciones (1+log2(dimL+1))/2.

- Cuando dimL crece el número medio de comparaciones es $\log_2(\text{dimL}+1)/2$.

Eficiencia de la Busqueda secuencial y dicotómica

	busqueda secuencial		Busqueda dicotómica	
	número medio de comparaciones		Número máximo de comparaciones	
N	localizado	no localizado	Localizado	no localizado
7	4	7	3	3
100	50	100	7	7
1.000	500	1.000	10	10
1.000.000	500.000	1.000.000	20	20

ESTRUCTURA DE DATOS VECTOR – Para

Un centro de deportes quiere almacenar la información de sus clientes y de los 4 tipos de actividades que ofrece: *1) Musculación, 2) Spinning, 3) Cross Fit, 4) Libre*. Para ello, se debe leer el precio mensual de cada actividad y almacenarlo en un vector. De cada uno se conoce: código de cliente, DNI, apellido y nombre, fecha de ingreso, edad y el número de actividad elegida (1..4).

Código Cliente	Apellido y Nombre	Fecha de Ingreso	Edad	Actividad
2000	López Juan	12/02/2000	20	1
2100	Díaz Ana	14/06/1999	25	3
3500	Perez Luis	22/05/2016	30	2
4055	Lares Pedro	05/12/2010	55	4
4500	Zanon Ema	06/09/2000	20	2
6400	Lera Sofia	05/12/2010	30	3
7000	Ávila Raul	14/06/1999	70	2
8250	García Mara	15/05/2004	50	1
-1				

Escribir un programa que invoque a módulos para resolver cada inciso:

a) Lea la información de los clientes y los almacene en una estructura de datos. La lectura finaliza con el código de cliente -1 y los clientes se leen ordenados por código de cliente. Como máximo el centro de deportes atiende a 1000 clientes.

b) Informe el nombre y apellido de los clientes cuya edad supera el promedio de las edades de los clientes del centro de deportes.

c) Agregar un nuevo cliente, con el código siguiente al último código de cliente ingresado, para el cliente Juan García con DNI 11111, de 20 años de edad y que eligió la actividad Libre, en el día de la fecha.

d) Informe el nombre y apellido del cliente con un código de cliente determinado.

e) Informe el nombre y apellido de los clientes con fecha de ingreso en un año determinado.

f) Sabiendo que el código de cliente 3300 no existe, inserte un nuevo cliente con ese código para el cliente Ana Paus de 45 años y actividad elegida Musculación, en el día de la fecha.

g) Elimine el cliente correspondiente a un código que se lee.

h) Elimine todos los clientes que realizan la actividad Spinning.