PROGRAMACIÓN I

AÑO 2025

Estructura de datos VECTOR

TEMAS de la CLASE

1. Operaciones frecuentes en el tipo Vector
   - Borrar un elemento de una posición determinada
   - Búsqueda
   - Borrar un elemento determinado
   - Búsquedas en vectores ordenados
   - Insertar un elemento en un vector con orden

2. Ejercitación

Tipo Vector: Borrar un elemento

La operación de Borrar un elemento en un vector consiste en eliminar un elemento determinado o bien eliminar un elemento que ocupa una posición determinada.
En el caso de borrar un elemento en una posición determinada, se debe verificar que la posición sea válida. En el caso de eliminar un elemento determinado, hay que verificar que exista dicho elemento.

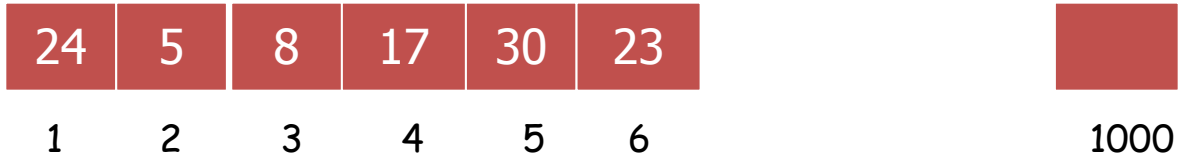
CONSIDERACIONES

- **dimL** : cantidad de elementos en el vector (dimensión lógica).
- Luego de la operación la **cantidad** de elementos es **dimL-1**.

Tipo Vector: Borrar un elemento

Cuando se requiere borrar un elemento en un vector se presentan dos posibilidades distintas:

1 Borrar un elemento de una posición determinada



Posición 2

2 Borrar un elemento determinado del vector

Lo veremos más adelante

Tipo Vector: Borrar un elemento de una posición determinada

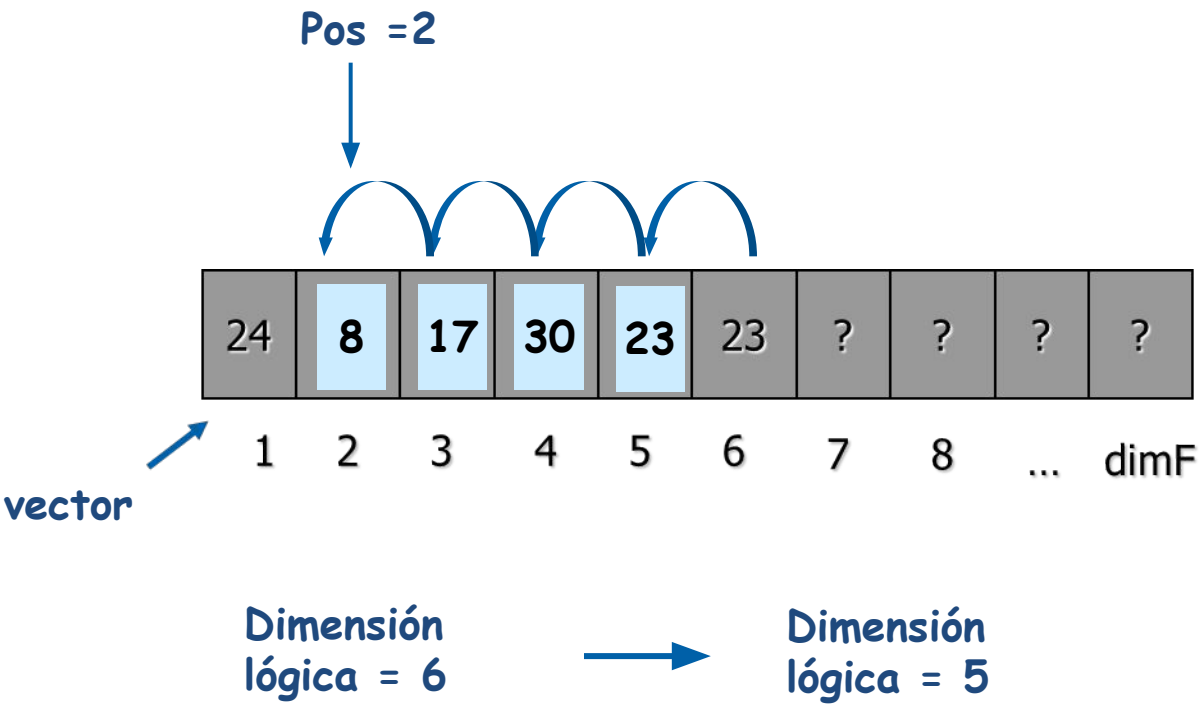
➡ Esta operación también tiene que verificar que la posición sea válida.

1. Validar la posición a borrar

2. Desplazar elementos (a partir de la posición siguiente)

3. Disminuir la dimensión lógica

Supongamos que se quiere borrar el elemento de la posición 2 de un vector de valores enteros…



Tipo Vector: Borrar un elemento de una posición determinada

Consideremos la siguiente declaración:

```
Const
    dimF = ...      {máxima longitud del vector}
Type
    vector =  Array  [ 1..DimF] of integer;
    vecotr_dimL = record
                      v: vector;
                      dimL: integer;
                  end;
```

Tipo Vector: Borrar un elemento de una posición determinada

1. Validar la posición a borrar

2. Desplazar elementos (a partir de la posición siguiente)

3. Disminuir la dimensión lógica

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

Validar la posición a borrar

Desplazar elementos

Disminuir la dimensión lógica

Parámetros de la operación:

- ✔v: vector a trabajar
- ✔dimL: cantidad de elementos
- ✔pos: posición a borrar
- ✔exito: resultado operación

Ejercitación

1. Se lee una sucesión de nombres y notas de alumnos que finaliza con nombre “Fulano”. Informar los nombres y notas de los alumnos que superan el promedio del grupo.

Nota: Como máximo se pueden ingresar 100 alumnos.

- **Leer, Guardar y Sumar todas las notas**
- **Calcular promedio**
- **Recorrer y Comparar cada nota con el promedio**

Leer, Guardar y Sumar todas las notas

Inicializar suma de notas

Leer datos de alumno

Mientras (haya lugar) y (no ingrese “Fulano”)

  guardar en el vector

  actualizar suma de notas

  leer datos de alumno

Recorrer y Comparar cada nota con el promedio

Repetir cantidad alumnos guardada

  acceder a los datos del alumno

  si nota > promedio entonces

    informar nombre

3. Se lee una sucesión de nombres y notas de alumnos que finaliza con nombre “Fulano”. Informar los nombres y notas de los alumnos que superan el promedio del grupo.

Nota: Como máximo se pueden ingresar 100 alumnos.

- Leer, Guardar y Sumar todas las notas
- Calcular promedio
- Recorrer y Comparar cada nota con el promedio

```
Program ejemplo2;
const dimF= 100; {dimensión física}
type  str20 = string [20];
   alumno = record
              nombre: str20;
              nota: real;
            end;
   vector = array [1..dimF] of alumno;
   vector_dimL = record
                   v: vector;
                   dimL: integer;
                 end;
{implementación LeerGuardarSumar}
{implementación RecorreryComparar}

var
   vD: vector_dimL;
   suma, promedio : Real;

Begin
  LeerGuardarSumar(vD, suma);
  if (vD.dimL <>0)
  then begin
          promedio := suma/vD.dimL;
          RecorreryComparar(vD, promedio)
       end;
End.
```

Leer, Guardar y Sumar todas las notas

Inicializar suma de notas

Leer datos de alumno

Mientras (haya lugar) y (no ingrese “Fulano”)

- guardar en el vector
- actualizar suma de notas
- leer datos de alumno

```
const dimF= 100; {dimensión física}

type  str20 = string [20];
   alumno = record
              nombre: str20;
              nota: real;
            end;
   vector = array [1..dimF] of alumno;
   vector_dimL = record
                   v: vector;
                   dimL: integer;
                 end;
```

```
procedure LeerGuardarSumar
      (var vD:vector_dimL; var sum: Real);

Procedure leerAlumno (var alu:alumno);
begin
 read (alu.nombre);
 if (alu.nombre <> ´Fulano´)
     then read (alu.nota);
end;

var
   a : alumno;

begin
 sum := 0;
 vD.dimL := 0; {dimensión  lógica}
 leerAlumno (a);
 while (vD.dimL < dimF) and (a.nombre <>
´Fulano´)
   do begin
     vD.dimL := vD.dimL + 1;
     vD.v [vD.dimL]:= a;
     sum := sum + vD.v [vD.dimL].nota;
     leerAlumno (a);
   end
end;
```

Recorrer y Comparar cada nota con el promedio

Repetir cantidad alumnos guardada

  acceder a los datos del alumno

  si nota > promedio entonces

            informar nombre

```
const dimF= 100; {dimensión física}

type  str20 = string [20];
   alumno = record
              nombre: str20;
              nota: real;
            end;
   vector = array [1..dimF] of alumno;
   vector_dimL = record
                   v: vector;
                   dimL: integer;
                 end;
```

```
Procedure RecorreryComparar  (vD: vector_dimL; prom: real);
 var i: integer;
begin
  for i := 1 to vD.dimL do
       if (vD.v[i].nota > prom) then
          Writeln (vD.v[i].nombre, ´ ´, vD.v[i].nota );
End.
```

Tipo vector: Operación de Búsqueda

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

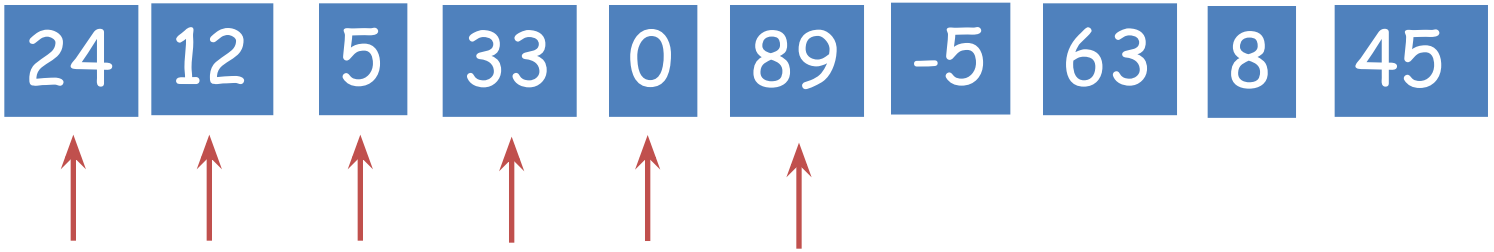
Se deben considerar los siguientes casos:

- Los datos en el vector se encuentran almacenados sin ningún orden.

- Los datos en el vector se encuentran almacenados ordenados por algún criterio

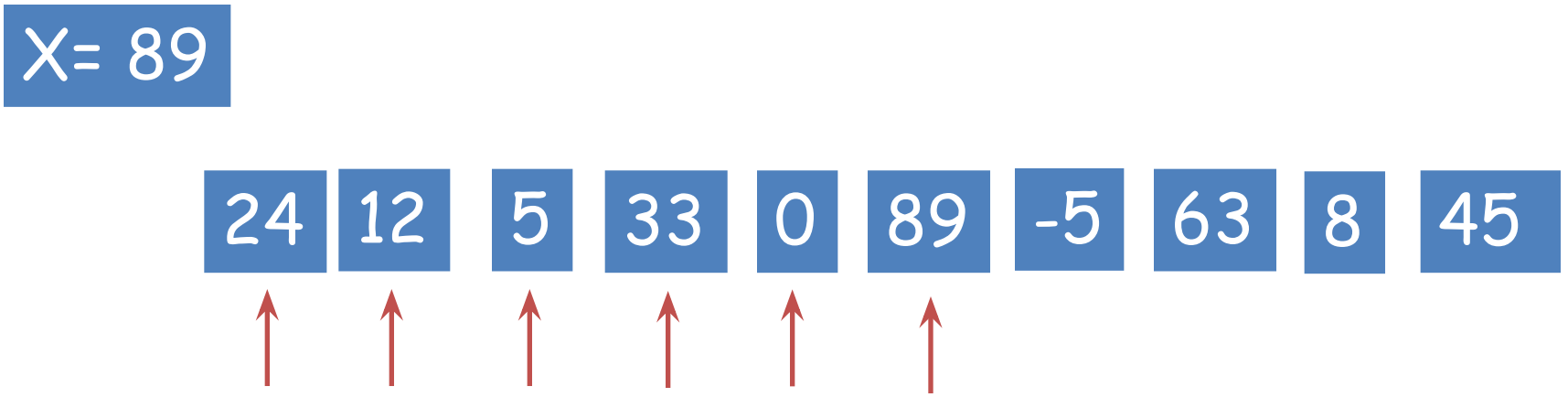
Tipo Vector: Búsqueda Lineal o secuencial (elementos sin orden)

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
```

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

entos sin orden)

```
const dimF= ...; {dimensión física}

type  vector = array [1..dimF] of integer;
      vector_dimL = record
                      v: vector;
                      dimL: integer;
                    end;
```

¿Qué tipo de módulo Conviene utilizar?

```
Function BuscarPosElem (vD: vector_dimL; x:integer): integer;
var pos: integer;
Begin
 pos:=1;
 while (pos <= vD.dimL) and (vD.v[pos] <> x) do
      pos:=pos+1;
 if (pos<= vD.dimL) then BuscarPosElem := pos
                    else BuscarPosElem := 0
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= ...; {dimensión física}

type  vector = array [1..dimF] of integer;
      vector_dimL = record
                      v: vector;
                      dimL: integer;
                    end;
```

```
Procedure BorrarElem (var vD: vector_dimL;
                      elem : integer;  var exito: boolean);
var pos: integer;

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

Ya lo vimos!!

Revisar BorrarElemPos y adaptarlo!!

Tipo Vector: Borrar un elemento determinado

```
Procedure BorrarElem (var vD: vector_dimL;
                      elem : integer;  var exito: boolean);
```

Ya lo vimos!!

```
Function BuscarPosElem (vD: vector_dimL; x:integer): integer;
var pos: integer;
Begin
 pos:=1;
 while (pos <= vD.dimL) and (vD.v[pos] <> x) do
      pos:=pos+1;
 if (pos<= vD.dimL) then BuscarPosElem := pos
                    else BuscarPosElem := 0
end;
```

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

 end;
```

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

Tipo Vector: Borrar un elemento determinado

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

Ya lo vimos!!

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

BorrarPos adaptado

Tipo Vector: Borrar un elemento determinado

```
Procedure BorrarElem (var vD: vector_dimL;
                      elem : integer;  var exito: boolean);

 Function BuscarPosElem (vD: vector_dimL; x:integer): integer;
 var pos: integer;
 Begin
  pos:=1;
  while (pos <= vD.dimL) and (vD.v[pos] <> x) do
       pos:=pos+1;
  if (pos<= vD.dimL) then BuscarPosElem := pos
                     else BuscarPosElem := 0
 end;
Procedure BorrarPosModif (var vD:vector_dimL; pos: integer);
 var i: integer;
 begin
    for i:= pos + 1  to   vD.dimL  do
       vD.v [ i - 1 ]  :=  vD.v [ i] ;
    vD.dimL := vD.dimL - 1 ;
 end;
var pos: indice;
Begin
  exito:= False;
  pos:= BuscarPosElem (vD, elem);
  if pos <> 0 then begin
                    BorrarPosModif (vD, pos);
                    exito:= true;
                end;
end;
```

Tipo Vector: Búsqueda en Arreglos Ordenados

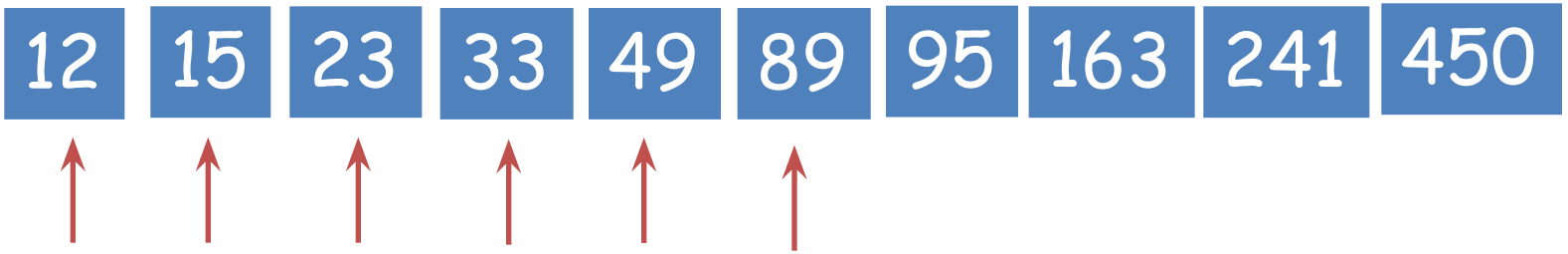
Métodos de Búsqueda

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

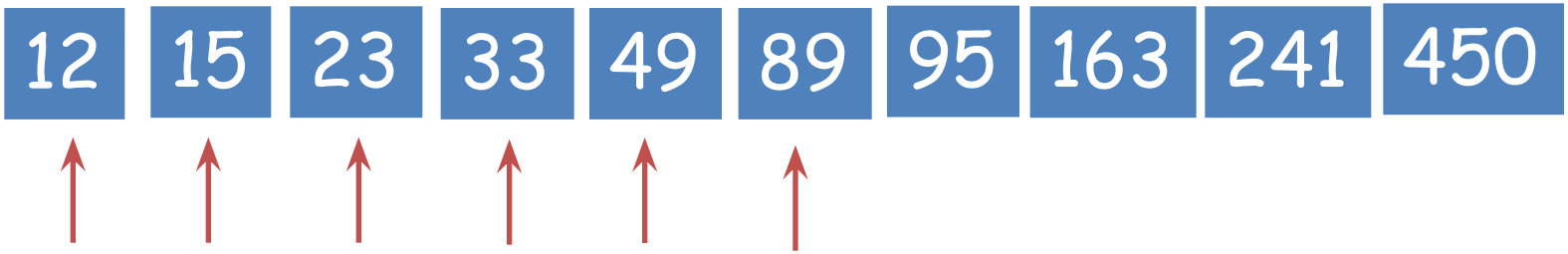


- Se aplica cuando los elementos tienen orden.

- 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= ...; {dimensión física}

type  vector = array [1..dimF] of integer;
      vector_dimL = record
                      v: vector;
                      dimL: integer;
                    end;
```

Vector ordenado de menor a mayor

```
Function BuscarPosElemOrd (vD: vector_dimL; x:integer): integer;
var pos: integer;
Begin
 pos:=1;
 while (pos < vD.dimL) and (vD.v[pos] < x) do
       pos:=pos+1;
 if (pos<= vd.dimL) and (vD.v[pos] = x)
 then BuscarPosElem := pos
 else BuscarPosElem := 0
end;
```

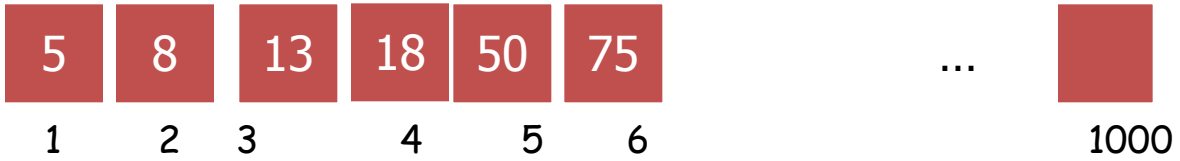
Tipo Vector: Insertar un elemento

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

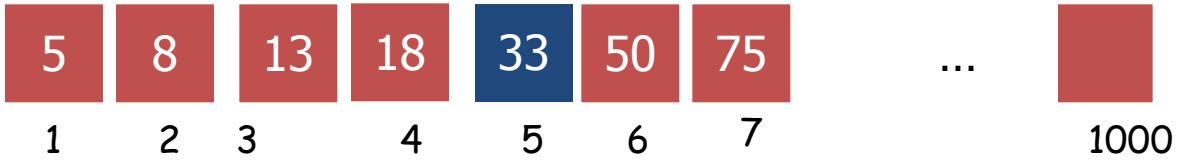
1 Insertar un elemento en una posición determinada

Ya lo vimos

2 Insertar un elemento manteniendo un orden determinado



Elemento 33



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= ...; {dimensión física}

type  vector = array [1..dimF] of integer;
      vector_dimL = record
                      v: vector;
                      dimL: integer;
                    end;
```

```
Procedure InsertarElemOrd (var vD: vector_dimL;
                           elem: integer;  var exito: boolean);
var pos: integer;

begin
  exito:= false:
  if (vD.dimL < dimF)
  then begin
         pos:= BuscarPosicion (vD, elem);
         Insertar (vD, pos, elem);
         exito:= true;
       end;
end;
```

Revisar BuscoPosElemOrd y adaptarlo!!

Revisar InsertarPos y adaptar

Adaptamos "BuscoPosElemOrd" para escribir "BuscarPosicion"

```
const dimF= ...; {dimensión física}
type  vector = array [1..dimF] of integer;
      vector_dimL = record
                      v: vector;
                      dimL: integer;
                    end;
```

```
Function BuscarPosElemOrd (vD: vector_dimL; x:integer): integer;
var pos: integer;
Begin
 pos:=1;
 while (pos < vD.dimL) and (vD.v[pos] < x) do
      pos:=pos+1;
 if (pos <= vD.dimL) and (vD.v[pos] = x) then BuscarPosElem := pos
                                         else BuscarPosElem := 0
end;
```

Ya lo vimos!!

```
Function BuscarPosicion ( vD: vector_dimL; x: integer): integer;
     var pos: integer;
    begin
      pos:=1;
      while (pos<= vD.dimL) and (vD.v[pos] < x) do
         pos:= pos+1;
      BuscarPosicion:= pos;
    end;
```

BuscoPosElemOrd adaptado

Adaptamos "InsertarPos" para escribir "Insertar"

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

```
Procedure INSERTARPOS(var vD: vector_dimL; elem: integer; pos:integer; var exito: boolean );

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

Ya lo vimos!!

```
Procedure Insertar (var vD: vector_dimL; pos: integer; elem:integer);
    var i: integer;
    begin
        for i:= vD.dimL downto pos do
            vD.v [ i +1 ] := vD.v [ i ] ;
        vD.v [ pos ] := elem;
        vD.dimL := vD.dimL + 1;
    End;
```

InsertarPos adaptado

Tipo Vector: Insertar un elemento en un vector ordenado

```
Procedure InsertarElemOrd (var vD: vector_dimL; elem : TipoElem;
                           var exito: boolean);

  Function BuscarPosicion ( vD: vector_dimL; x: integer): integer;
    var pos: integer;
    begin
      pos:=1;
      while (pos <= vD.dimL) and (vD.v[pos] < x) do
            pos:=pos+1;
      BuscarPosicion:= pos;
    end;

  Procedure Insertar (var v:vector_dimL; pos: integer; elem:integer);
    var i: integer;
    begin
      for i:= vD.dimL downto pos do
            vD.v [ i +1 ] := vD.v [ i ] ;
      vD.v [ pos ] := elem;
      vD.dimL := vD.dimL + 1;
    End;

 var pos: integer;

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

Nuevo!!!

Nuevo!!!

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.

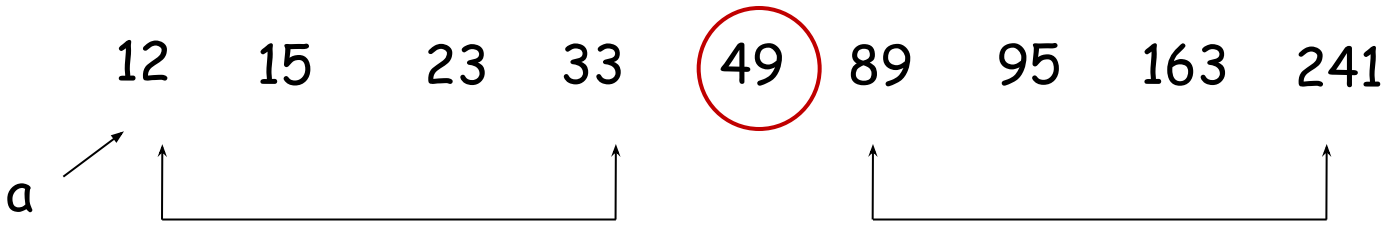
Tipo vector: Búsqueda en Arreglos Ordenados
Método 2 – Búsqueda Dicotómica

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

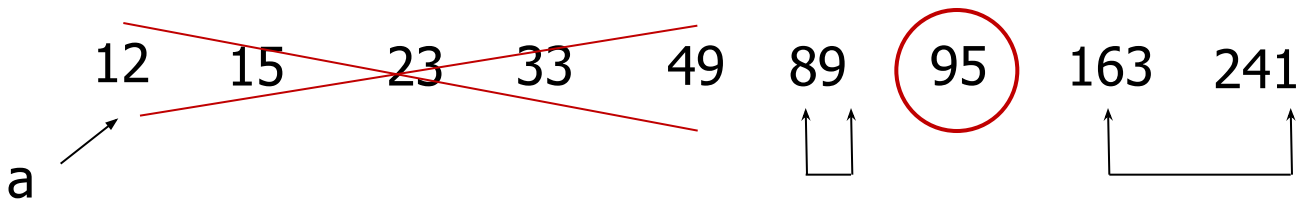
Segunda vez

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

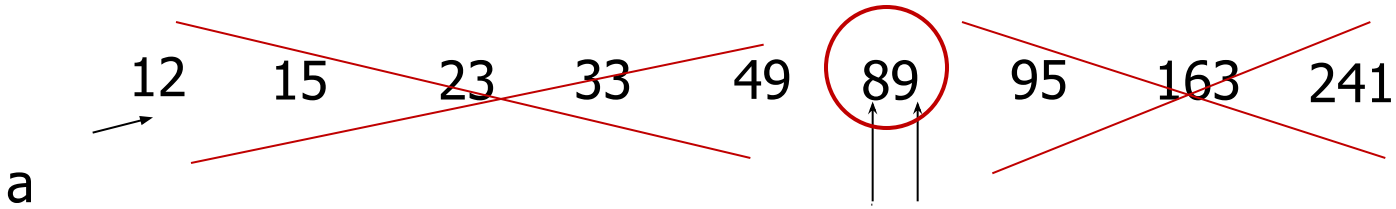
Elemento buscado X= 89

- 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 más 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 vD: vector_dimL ; var j: Indice; x : integer) ;
  Var pri, ult, medio : integer ;
  Begin
    j :=0 ;
    pri:= 1 ;
    ult:= vD.dimL;
    medio := (pri + ult ) div 2 ;
    While  ( pri < = ult ) and ( x <> vD.v [medio]) do begin
          If ( x < vD.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"

¿Qué ocurre cuando el elemento buscado no está en el vector?

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

Ejercitación

Un centro de deportes nos ha encargado un sistema para el manejo de sus clientes. En el centro se ofrecen 5 actividades distintas. De cada cliente se conoce el código de cliente, DNI, apellido, nombre, edad, el número de actividad que realiza, mes y año del último pago. La lectura finaliza cuando llega el DNI 0.
Se pide realizar un módulo para cada uno de los siguientes incisos:

a) Almacenar el precio mensual de las actividades a través de la lectura de los valores.
b) Generar la estructura de datos que almacene los clientes del centro de deportes (como máximo 500), leyendo los datos de los clientes hasta ingresar un DNI -1, **ordenados por código de cliente.**

Una vez almacenada la información de los clientes, implementar:

a) Un módulo que determine e informe la cantidad de clientes que realizan cada actividad y la actividad con mayor recaudación mensual.
b) Un módulo que elimine de la estructura el cliente que está ubicado en una posición que se recibe como parámetro.
c) Un módulo que elimine de la estructura el cliente correspondiente a un código que se recibe como parámetro.
d) Un módulo que elimine todos los clientes que realizan la actividad 3.