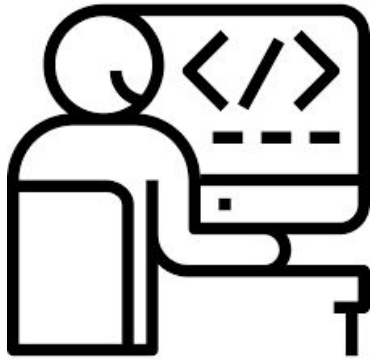


Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de datos ARREGLO
- Definición y características
- Operaciones esenciales

CADP – TIPOS DE DATOS ESTRUCTURADOS

SIMPLE: aquellos que toman un único valor, en un momento determinado, de todos los permitidos para ese tipo.

COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.

- TIPO DE DATO
  - SIMPLE
    - DEFINIDO POR EL LENGUAJE
      - Integer
      - Real
      - Char
      - Boolean
    - DEFINIDO POR EL PROGRAMADOR
      - Subrango
  - COMPUESTO
    - DEFINIDO POR EL LENGUAJE
      - String
    - DEFINIDO POR EL PROGRAMADOR
      - Registros
      - Arreglos

Supongamos que se quiere leer la edad de 10 personas y al finalizar informar cuantas veces apareció la edad máxima.

Edades Leídas

20
77
68
2
77
23
4
15
15
3

Máximo 77

Y ahora que sé que la edad máxima es 77 cómo informo cuantas veces apareció?

Con lo que sabemos hasta ahora tenemos dos alternativas

CADP – TIPOS DE DATOS ARREGLOS

Supongamos que se quiere leer la edad de 10 personas y al finalizar informar cuantas veces apareció la edad máxima.

Solución 1

Ingresar los valores.
Calcular el máximo.
Ingresar los valores nuevamente e imprimir cuáles coinciden con el máximo calculado.

PROBLEMA: se debe garantizar que el usuario ingrese los mismos valores. Cuantos más valores se lean el problema es más grande.

Solución 2

Ingresar los valores y guardar cada valor en una variable.
Calcular el máximo.
Comparar cada variable con el máximo calculado.

PROBLEMA la cantidad de variables a usar, la legibilidad del programa. Cuantos más valores se lean el problema es más grande

SOLUCION?

CADP – TIPOS DE DATOS ARREGLOS

Supongamos que se quiere leer la edad de 10 personas y al finalizar informar cuantas veces apareció la edad máxima.

Disponer de alguna **ESTRUCTURA** donde almacenar los números, para luego calcular el máximo, y así finalmente poder compararlo contra los valores almacenados.

Leer los números y almacenarlos

20	77	68	2	77	23	4	15	15	3

Recorrer la estructura y obtener el máximo

20	77	68	2	77	23	4	15	15	3

77

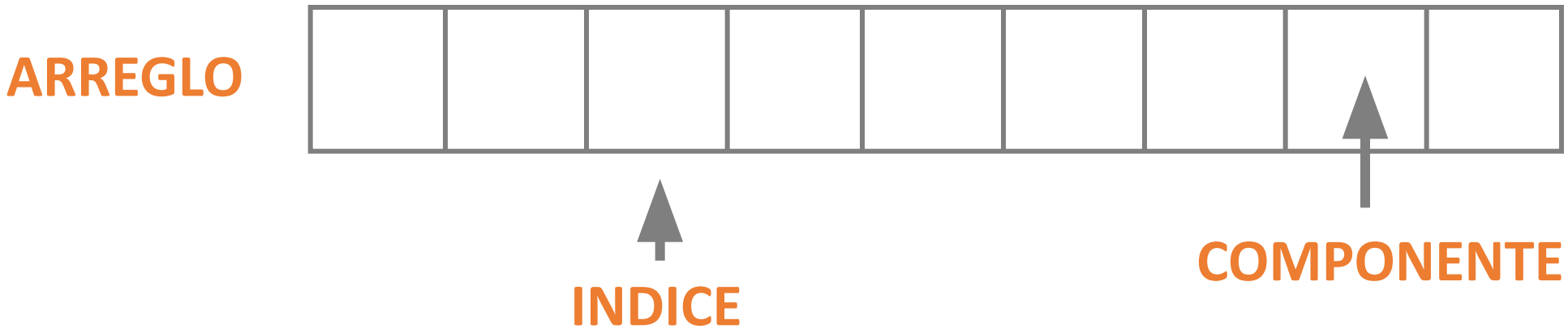
Recorrer la estructura y comparar con el máximo

20	77	68	2	77	23	4	15	15	3

2

ARREGLO

Un arreglo (ARRAY) es una estructura de datos compuesta que permite acceder a cada componente por una variable índice, que da la posición de la componente dentro de la estructura de datos.



Los arreglos de una dimensión se llaman vectores.

VECTOR (arreglo de una dimensión)

Es una colección de elementos que se guardan consecutivamente en la memoria y se pueden referenciar a través de un índice.

Cómo se declara?

HOMOGENEA → Los elementos pueden son del mismo tipo .

ESTATICA → El tamaño máximo no cambia durante la ejecución (se calcula en el momento de compilación)

INDEXADA → Para acceder a cada elemento de la estructura se debe utilizar una variable '**índice**' que es de tipo ordinal.

ARREGLOS - VECTOR

VECTOR

```
Program uno;
Const
   ....
Type

    vector = array [rango] of tipo;

Var
    variable: vector;
```

El **rango** debe ser un tipo ordinal
char, entero, booleano, subrango

Unos ejemplos ...

El **tipo** debe ser un tipo estático
char, entero, booleano, subrango, real
registro, vector

CADP – TIPOS DE DATOS

ARREGLOS - VECTOR

```
type
    numeros = array [1..10]  of real;
    frecuen = array [char]  of real;
    otros = array ['h'..'m']  of integer;
```

Cómo trabajamos?

Var

num: numeros; **num reserva memoria para 10 números reales**

15,25	-7	179,3	0	8,45	10,25	9	8,45	10,5	9
1	2	3	4	5	6	7	8	9	10

nuevo: frecuen; **nuevo reserva memoria para 256 números reales**

15,25	-7,5	179,3							19
A									z

otro: otros; **otro reserva memoria para 6 números enteros**

15	-7	1879	0	8	10
h	i	j	k	l	m

VECTORES - OPERACIONES

Carga de valores

Lectura / Escritura

Recorridos

Agregar elementos al final

Insertar elementos

Borrar elementos

Búsqueda de un elemento

Ordenación de los elementos

CADP – TIPO DE DATOS VECTOR - OPERACIONES

CON LA VARIABLE VECTOR

```
Program uno;
Const
  ....
Type

   vector = array [1..10] of integer;

Var
   v1,v2: vector;

Begin

 ....
 v2:= v1;
 ...
End.
```

La única operación permitida es la asignación entre dos variables del mismo tipo

CADP – TIPO DE DATOS VECTOR - OPERACIONES

CON LOS ELEMENTOS

```
Program uno;
Const
  ....
Type

   vector = array [1..10] of integer;

Var
   v1,v2: vector;

Begin

 ....

End.
```

La única forma de acceder a los elementos es utilizando un índice

`variable [pos]`

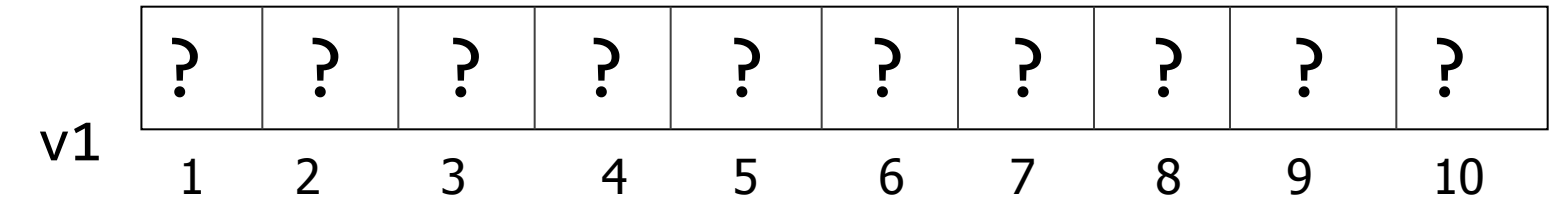
`variable [4]`

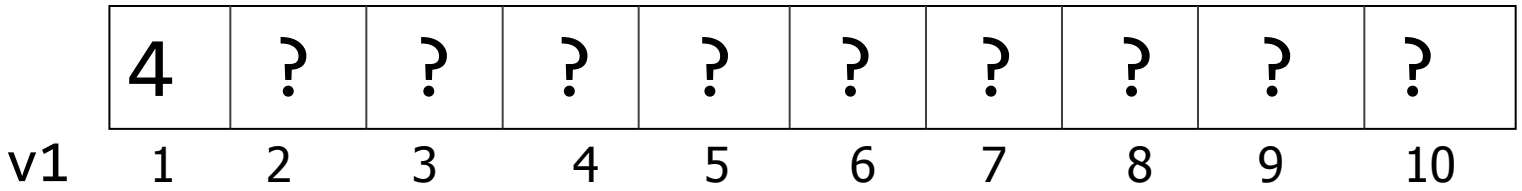
Las operaciones con el elemento son las permitidas para el tipo de datos del elemento

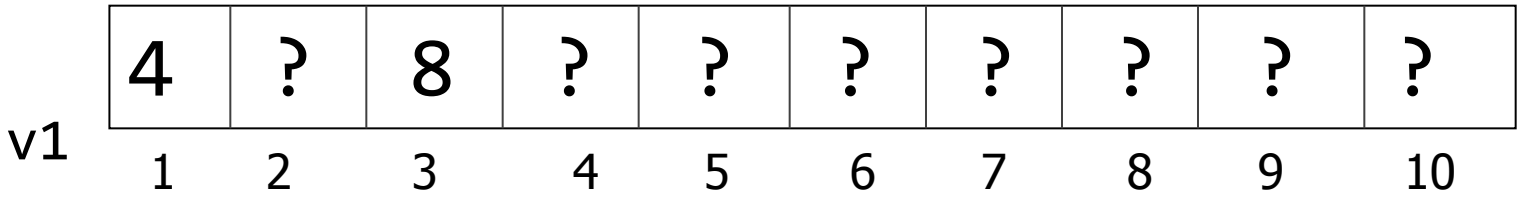
CADP – TIPO DE DATOS VECTOR - CARGA

```
Program uno;
Const
  ....
Type

   vector = array [1..10] of integer;

Var
   v1: vector;

 Begin
  v1[1]:= 4;

  v1[3]:= 8;
 End.
```

Cómo se carga completo?







CADP – TIPO DE DATOS VECTOR - CARGA

```
Program uno;

 Const
  tam = 10;

Type
   vector = array [1..tam]   of   integer;
Var
 v1:vector;
 i,valor:integer;

 Begin
  for i:= 1 to tam do
   begin
    read (valor);
    v1[i]:= valor;
   end;
 End.
```

No se puede hacer read(v1)

v1

?	?	?	?	?	?	?	?	?	?
1	2	3	4	5	6	7	8	9	10

Cómo se modulariza?

v1

-1	18	4	0	5	57	-2	3	8	4
1	2	3	4	5	6	7	8	9	10

ALTERNATIVA

```
Begin
 for i:= 1 to tam do
  begin
   read(v1[i]);
  end;
End.
```

CADP – TIPO DE DATOS VECTOR - CARGA

```
Procedure carga (var v: vector);

  var
   i,valor:integer;

  begin
   for i:= 1 to tam do
    begin
      read (valor);
      v[i]:= valor;
    end;
  end;
```

v

?	?	?	?	?	?	?	?	?	?
1	2	3	4	5	6	7	8	9	10

v

-1	18	4	0	5	57	-2	3	8	4
1	2	3	4	5	6	7	8	9	10

Puede ser una función?

Se puede utilizar tam?

Cómo muestro los datos?

ALTERNATIVA

```
Procedure carga (var v: vector);
  var
   i:integer;
  begin
   for i:= 1 to tam do
      read (v[i]);
end;
```

CADP – TIPO DE DATOS VECTOR - MUESTRA

```
Program uno;

 Const
  tam = 10;
Type
   vector = array [1..tam]   of   integer;
Var
 v1:vector;
 i,valor:integer;

Begin
 carga (v1);
 for i:= 1 to tam do
    begin
     valor:= v1[i];
     write (valor);
    end;
 End.
```

No se puede hacer write(v1)

v1

?	?	?	?	?	?	?	?	?	?
1	2	3	4	5	6	7	8	9	10

v1

-1	18	4	0	5	57	-2	3	8	4
1	2	3	4	5	6	7	8	9	10

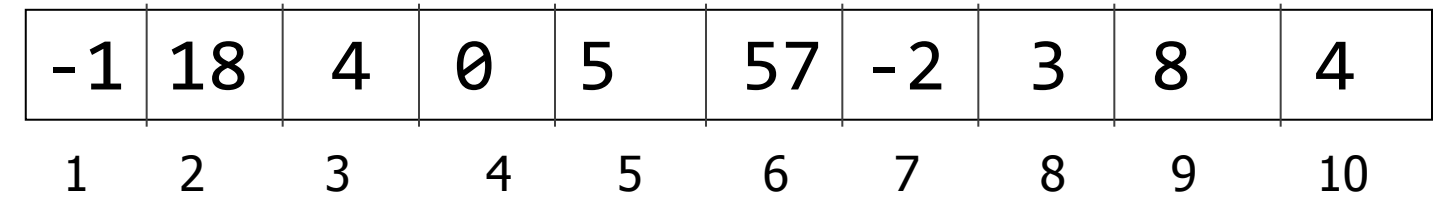
Cómo se modulariza?

ALTERNATIVA

```
Begin
 for i:= 1 to tam do
  begin
   write(v1[i]);
  end;
End.
```

CADP – TIPO DE DATOS VECTOR - CARGA

```
Procedure imprimir (v: vector);
var
 i,valor:integer;

begin
 for i:= 1 to tam do
  begin
    valor:= v[i];
    write(valor);
  end;
end;
```



Puede ser una función?

ALTERNATIVA

```
Procedure imprimir (v: vector);
  var
   i:integer;
  begin
   for i:= 1 to tam do
      write (v[i]);
end;
```

Cómo solucionamos nuestro problema inicial?

CADP – TIPO DE DATOS VECTOR -

Escriba un programa que lea 10 números enteros y al finalizar informe cuantas veces apareció el número máximo.

- Cómo se carga el vector? Estructura de control?
- Cómo calculo el máximo? Qué tipo de módulo elijo?
- Cómo calculo cuántas veces aparece el valor máximo? Qué tipo de módulo elijo?

CADP – TIPO DE DATOS VECTOR -

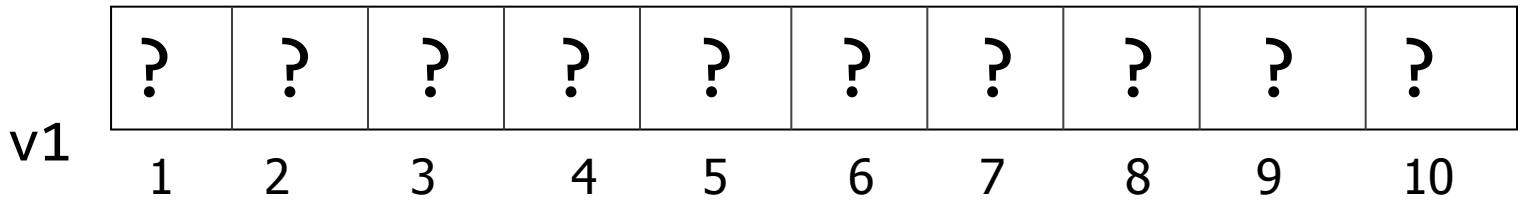
Escriba un programa que lea 10 números enteros y al finalizar informe cuantas veces apareció el número máximo.

Cargar vector (v)

Calcular el máximo (v , max)

Verificar cuantas veces aparece max en el vector v

CADP – TIPO DE DATOS VECTOR -




```
Program uno;

Const
 tam = 10;

 Type
   vector = array [1..tam] of integer;
 Var
  v1:vector;
  i,max,cant:integer;

 Begin
  carga (v1);
  max:= máximo (v1);
  cant:= verificar (v1,max);
  write (cant);
 End.
```

CADP – TIPO DE DATOS VECTOR -

```
procedure carga (var v:vector);
  Var
     i,valor:integer;

   Begin
     for i:= 1 to tam do
       begin
        read (valor);
        v[i]:= valor;
       end;
   End;
```

ALTERNATIVA

```
procedure carga (var v:vector);
Var
   i :integer;

 Begin
   for i:= 1 to tam do
     read (v[i]);

 End;
```

CADP – TIPO DE DATOS VECTOR -

```
function máximo (v:vector):integer;

  Var
     i,max,valor:integer;

     Begin
       max:= -9999;
       for i:= 1 to tam do
         begin
          valor:= v[i];
          if (valor >= max) then
            max:= v[i];
         end;

       maximo:= max;
     End;
```

v

-1	18	4	0	5	57	-2	3	57	4
1	2	3	4	5	6	7	8	9	10

ALTERNATIVA

```
function máximo (v:vector):integer;

  Var
   i,max:integer;

  Begin
    max:= -9999;
    for i:= 1 to tam do
      begin
        if (v[i] >= max) then
          max:= v[i];
      end;
    maximo:= max;
  End;
```

CADP – TIPO DE DATOS VECTOR -

```
function verificar (v:vector; valor:integer):integer;

Var
   i,cant,aux:integer;

   Begin
     cant:= 0;
     for i:= 1 to tam do
       begin
        aux:= v[i];
        if (valor = aux) then
          cant:= cant + 1;
       end;

     verificar:= cant;
   End;
```

v

-1	18	4	0	5	57	-2	3	57	4
1	2	3	4	5	6	7	8	9	10

57

ALTERNATIVA

```
function verificar(v:vector; valor:integer):integer;
  Var
   i,cant:integer;

  Begin
    cant:= 0;
    for i:= 1 to tam do
      begin
         if (valor = v[i]) then
           cant:= cant + 1;
      end;

    verificar:= cant;
  End;
```

CADP – TIPO DE DATOS VECTOR -

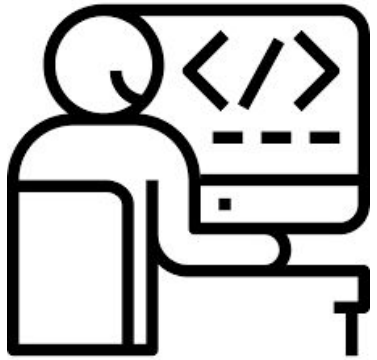
```
Program uno;

Const
 tam = 10;

Type
  vector = array [1..tam] of integer;
Var
 v1:vector;
 i,max,cant:integer;


Begin
 carga (v1);
 max:= máximo (v1);
 cant:= verificar (v1,max);
 write (cant);
End.
```

ALTERNATIVA 1

```
Begin
 carga (v1);
 max:= máximo (v1);
 write(verificar (v1,max));
End.
```

ALTERNATIVA 2

```
Begin
 carga (v1);
 write(verificar (v1,máximo(v1)));
End.
```



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de datos ARREGLO
- Recorridos totales
- Recorridos parciales

RECORRIDOS

Consiste en recorrer el vector de manera total o parcial, para realizar algún proceso sobre sus elementos.

Qué estructura de control implica cada uno?

RECORRIDO - TOTAL

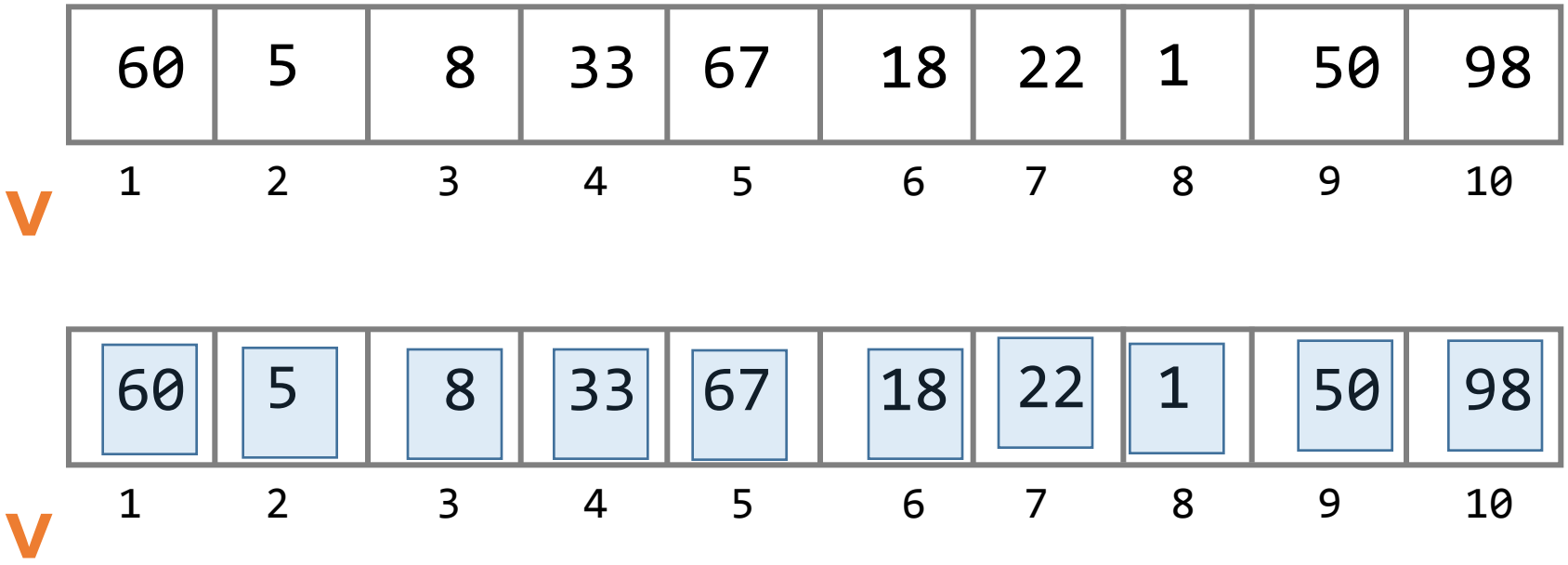
Implica analizar todos los elementos del vector, lo que lleva a recorrer completamente la estructura.

RECORRIDO - PARCIAL

Implica analizar los elementos del vector, hasta encontrar aquel que cumple con lo pedido. Puede ocurrir que se recorra todo el vector

CADP – TIPOS DE DATOS ARREGLOS

Realice un programa que llene un vector de 10 elementos enteros positivos y luego informe la cantidad de números múltiplos de 3. Suponga que los nros leídos son positivos.



Cant 0

Cant 1

Cant 2

Cant 3

Qué estructura de control?

Qué modularizo?

Cómo lo implemento?

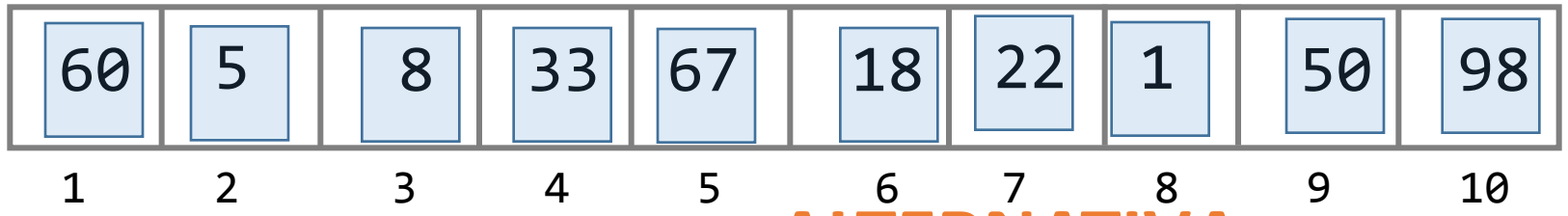
```
Program uno;
Const
 tam=10;
 multi=3;

Type
  vector = array [1..tam]   of   integer;

Var
 v1:vector;
 cant:integer;

Begin
 cargar (v1);
 cant:= múltiplos (v1);
 write (“La cantidad de múltiplos de”, multi, “es”, cant);
End.
```

```
Procedure cargar (var v:vector);
Var
  i,valor:integer;

Begin
  for i:= 1 to tam do
   begin
     read(valor);
     v[i]:= valor;
   end;
End;
```

ALTERNATIVA

```
Procedure cargar (var v:vector);
Var
  i:integer;

Begin
  for i:= 1 to tam do
   begin
     read(v[i]);
   end;
End;
```

```
function multiplos (v:vector):integer;

Var
  i,cant,resto: integer;

Begin
  cant:=0;
  for i:= 1 to tam do
   begin
     resto:= V[i] MOD multi;
     if (resto = 0) then
       cant:= cant + 1;
   end;
  multiplos:= cant;
End;
```



ALTERNATIVA

```
function multiplos (v:vector):integer;
Var
  i,cant: integer;

Begin
  cant:=0;
  for i:= 1 to tam do
   begin
     if ((v[i] MOD multi) = 0) then
       cant:= cant + 1;
   end;
  multiplos:= cant;
End;
```

CADP – TIPOS DE DATOS ARREGLOS

Realice un programa que cargue un vector de 10 elementos enteros positivos y luego informe la primer posición donde aparece un múltiplos de 3. Suponga que los nros leídos son positivos y que existe al menos un múltiplo de 3.

v

61	5	8	33	67	18	22	1	50	98
1	2	3	4	5	6	7	8	9	10

POS 4

v

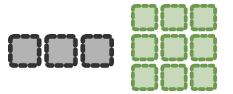
61	5	8	33	67	18	22	1	50	98
1	2	3	4	5	6	7	8	9	10

Qué estructura de control?

Qué modularizo?

Cómo lo implemento?

CADP – **VECTOR** RECORRIDO - PARCIAL

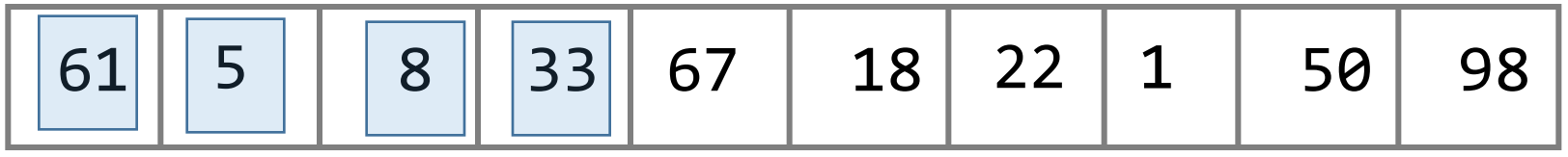


```
Program uno;
Const
 tam=10;
 multi=3;

Type
  vector = array [1..tam]   of   integer;

Var
 v:vector;
 pos:integer;

Begin
 cargar (v);
 pos:= posicion (v);
 write (“La posición del primer múltiplo de”, multi, “es”, pos);
End.
```

CADP – **VECTOR**

RECORRIDO PARCIAL

```
function posicion (v: vector): integer;
var
  pos,resto:integer;
  seguir:boolean;

begin
  seguir:= true; pos:=1;
  while (seguir = true) do
    begin
      resto:= v[pos] MOD multi;
      if (resto = 0) then
       seguir:= false
      else
       pos:= pos + 1;
    end;
  posicion:= pos;
end;
```



v

Por qué se inicializa pos en 1?

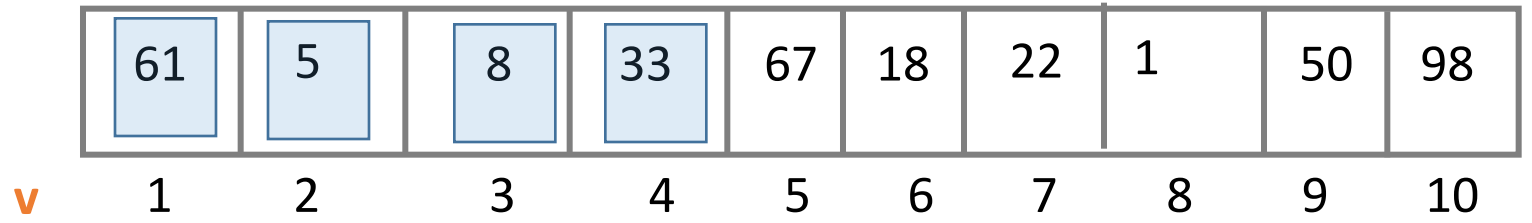
Por qué pos se incrementa en el else?

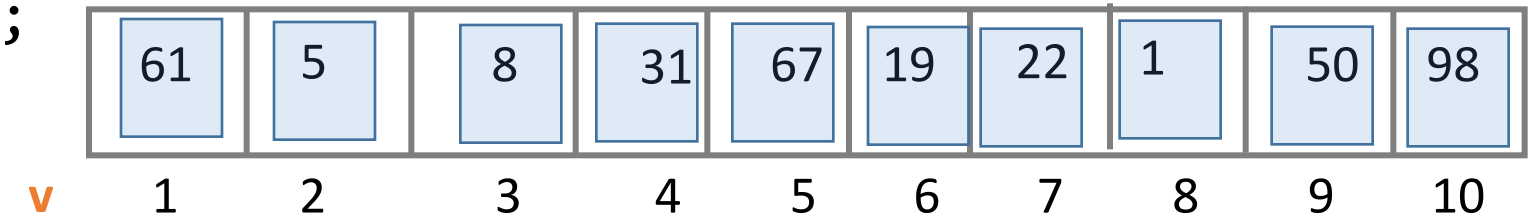
Qué cambio si el enunciado no asegura que haya al menos un múltiplo de 3?

CADP – **VECTOR**

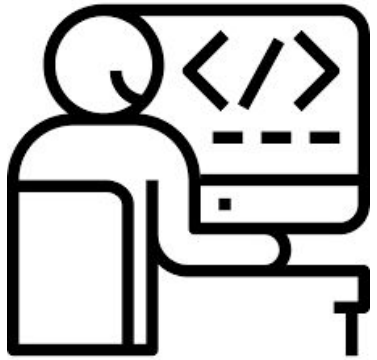
RECORRIDO PARCIAL

```
function posicion (v: vector): integer;
var
  pos,resto:integer;
  seguir:boolean;

begin
  seguir:= true; pos:=1;
  while ((pos<= tam) and (seguir = true)) do
    begin
      resto:= v[pos] MOD multi;
      if (resto = 0) then
       seguir:= false
      else
       pos:= pos + 1;
    end;
    if (seguir = false) then posicion:= pos
                        else posicion:= -1;
end;
```





Es necesario la última condición del if?



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de datos ARREGLO
- Dimensión física y dimensión lógica

VECTORES - OPERACIONES

Carga de valores

Lectura / Escritura

Recorridos

Dimensión física y lógica

Agregar elementos al final

Insertar elementos

Borrar elementos

Búsqueda de un elemento

Ordenación de los elementos

Supongamos que se existe un vector cargado de 10 elementos como máximo, pero por alguna circunstancia se cargaron sólo los primeros 4 valores.

20	77	68	2	?	?	?	?	?	?

a

Supongamos que sin saber que esto ocurrió se imprime el contenido del vector:

```
for i:= 1 to 10 do
   write (a[i]);
```

Que se obtendrá con la impresión?

DIMENSION FISICA

Se especifica en el momento de la declaración y determina su ocupación máxima de memoria.

La cantidad de memoria total reservada no variará durante la ejecución del programa.

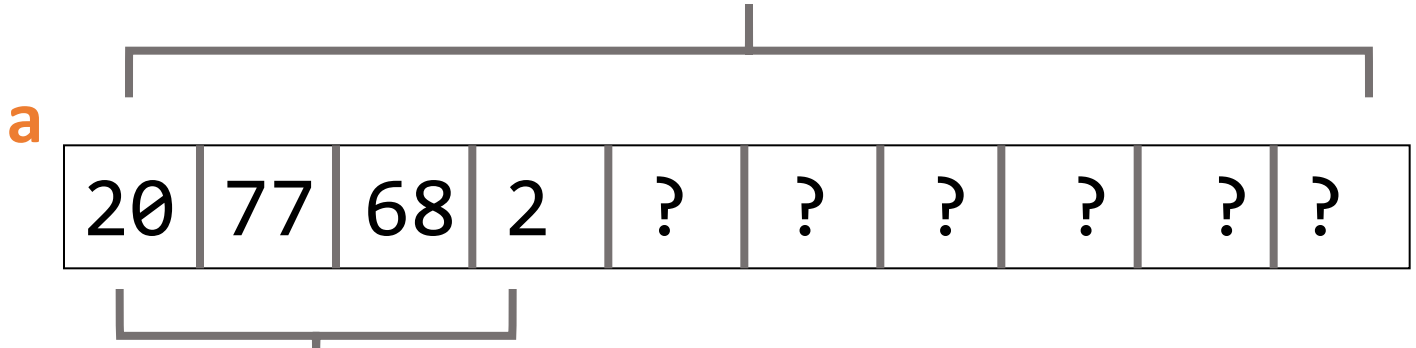
DIMENSION LOGICA

Se determina cuando se cargan contenidos a los elementos del arreglo. Indica la cantidad de posiciones de memoria ocupadas con contenido real.

Nunca puede superar la dimensión física.

VECTORES

DIMENSION FISICA



DIMENSION LOGICA

Es la cantidad de elementos reales que se guardan en el arreglo.
Puede modificarse durante la ejecución del programa
Nunca puede ser mayor a la dimensión física (se debe controlar)

Es la cantidad máxima de elementos que se pueden guardar en el arreglo.
No puede modificarse durante la ejecución del programa

Cuándo se determina cada una?

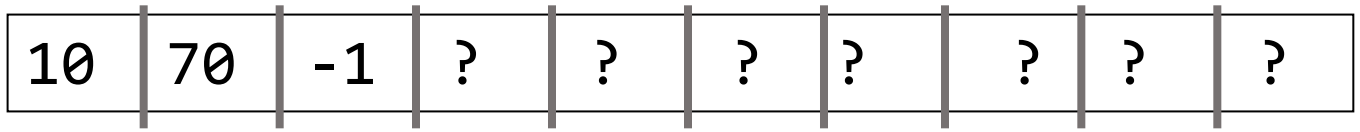
Donde se declaran?

CADP – TIPOS DE DATOS VECTORES

Realizar un programa que cargue un arreglo con números enteros hasta leer el número 50, como máximo deben almacenarse 10 números.
Luego de terminar la carga informe cuál es el número más grande de los leídos.

10
70
-1
50

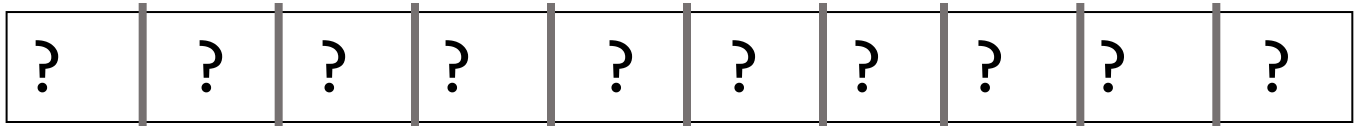
a



DF = 10
DL= 3

50

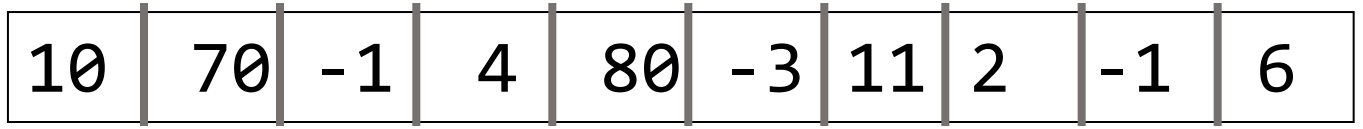
a



DF = 10
DL= 0

10
70
-1
4
80
-3
11
2
-1
6
...

a



DF = 10
DL= 10

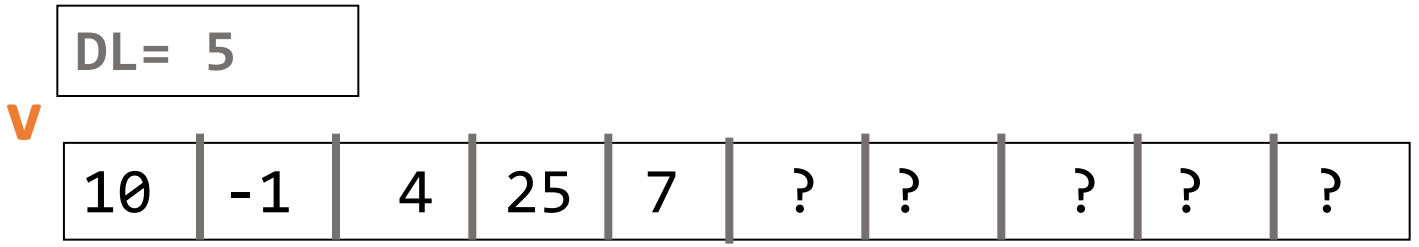
CADP – TIPOS DE DATOS VECTORES

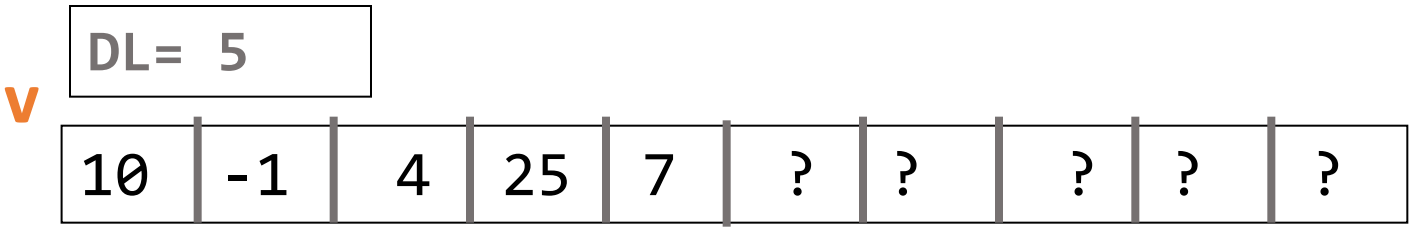
Realizar un programa que cargue un arreglo con números enteros hasta leer el número 50, a lo sumo se cargan 10 números.
Luego de terminar la carga informe cuál es el número mas grande de los leídos.

```
Program uno;
Const
 DF = 10
Type
   valores = array [1..DF] of integer;
Var
   v: valores;
   max:integer;
   dL:integer;
Begin
  cargarValores ( v , dL );
  max:= maximo (  v , dL );
  write (max);
End.
```

La **dimensión física** es una constante.

La **dimensión lógica** es una variable y toma valor cuando se carga el vector.





```
Procedure cargarValores (var a: números; var  dimL:integer);

 Var
  num:integer;

 Begin
    dimL:=0;
    read (num);
    while (num <> 50) do
     begin
      a[dimL]:= num;
      read(num);
     end;
  End;
```

Es correcto?

Cómo dimL, está inicializado en 0, la primera vez se accede a la posición a[0] y no es válida

```
Procedure cargarValores (var a: números; var  dimL:integer);

  Var
   num:integer;

  Begin
   dimL:=1;
   read (num);
   while (num <> 50) do
    begin
     a[dimL]:= num;
     read(num);
    end;
  End;
```

Es correcto?

Cómo dimL, nunca se incrementa, entonces carga siempre en la misma posición a[1]

```
Procedure cargarValores (var a: números; var  dimL:integer);

  Var
   num:integer;

  Begin
   dimL:=1;
   read (num);
   while (num <> 50) do
    begin
     a[dimL+1]:= num;
     read(num);
    end;
   End;
```

Es correcto?

Cómo dimL, nunca se incrementa, entonces carga siempre en la misma posición a[1]

```
Procedure cargarValores (var a: números; var  dimL:integer);

  Var
   num:integer;

  Begin
   dimL:=1;
   read (num);
   while (num <> 50) do
    begin
     a[dimL]:= num;
     dimL:= dimL+1;
     read(num);
    end;
  End;
```

Es correcto?

Si el primer número leído es 50, no entra al while, y como dimL está inicializado en 1, entonces devuelve que se cargó un elemento

```
Procedure cargarValores (var a: números; var  dimL:integer);

  Var
   num:integer;

  Begin
   dimL:=0;
   read (num);
   while (num <> 50) do
    begin
     dimL:= dimL+1;
     a[dimL]:= num;
     read(num);
    end;
End;
```

Es correcto?

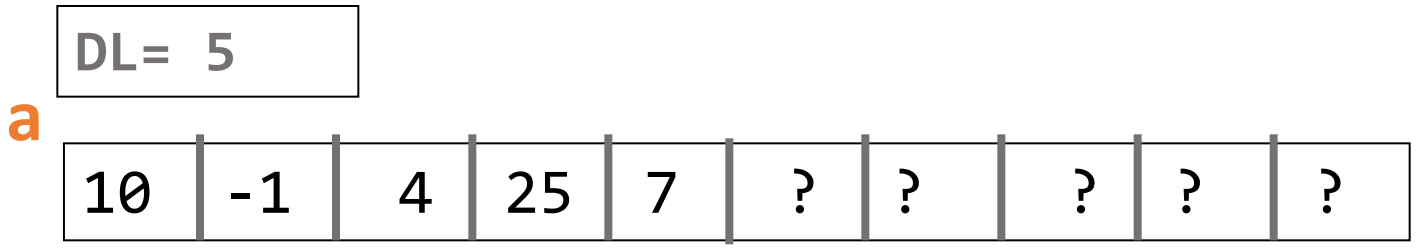
Qué pasa si leo mas de 10 números (el valor 50 no apareció y ya leí 10 valores)

```
Procedure cargarValores (var a: números; var  dimL:integer);

  Var
   num:integer;

  Begin
   dimL:=0;
   read (num);
   while ((dimL < dF) and (num <> 50)) do
    begin
     dimL:= dimL+1;
     a[dimL]:= num;
     read(num);
    end;
  End;
```

Es correcto?

SI!!!!

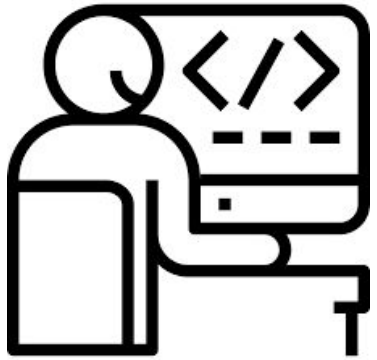
```
function maximo (a: números; dimL:integer):integer;

  Var
   max,i:integer;

  Begin
   max:=-9999;
   for i:= 1 to dF do
    begin
     if (a[i]>= max) then max:= a[i];
    end;

   maximo:= max;
  End;
```



Es correcto?

NO! Sólo hay que recorrer hasta la cantidad de elementos cargados realmente

```
function maximo (a: números; dimL:integer):integer;

Var
 max,i:integer;

Begin
 max:=-9999;
 for i:= 1 to dimL do
  begin
   if (a[i]>= max) then max:= a[i];
  end;

 maximo:= max;
End;
```



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de datos ARREGLO
- Agregar elementos
- Insertar elementos
- Eliminar elementos

VECTORES - OPERACIONES

Carga de valores

Lectura / Escritura

Recorridos

Dimensión física y lógica

Agregar elementos

Insertar elementos

Borrar elementos

Búsqueda de un elemento

VECTORES - AGREGAR

Agregar en un vector significa que el vector contendrá un elemento más, ubicado al atrás del último elemento cargado.

Puede pasar que esta operación no se pueda realizar si el vector está lleno (la dimensión lógica = dimensión física)

Dl = 4 5

45

a

34	10	-1	5						
1	2	3	4	5	6	7	8	9	10

Qué pasos considero?

VECTORES - AGREGAR

Agregar en un vector significa que el vector contendrá un elemento más, ubicado al atrás del último elemento cargado.

Puede pasar que esta operación no se pueda realizar si el vector está lleno (la dimensión lógica = dimensión física)

1- **Verificar** si hay espacio (cantidad de elementos actuales es menor a la cantidad de elementos posibles)

2- **Agregar** al final de los elementos ya existentes el elemento nuevo.

3- **Incrementar** la cantidad de elementos actuales.

Cómo se implementa?

VECTORES - AGREGAR

Dado un vector de números enteros (10 elementos como máximo) realice un programa que lea un nuevo número e invoque a un módulo que agregue el elemento en el vector.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
    VN: numeros;
    dimL, valor:integer;
    ok:boolean;

Begin
  cargar (VN,dimL);
  read(valor);
  agregar(VN,dimL,ok,valor);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 4

VN

4	-1	10	3	?	?	?	?	?	?

valor = 7 dimL = 5 ok = true

VN

4	-1	10	3	7	?	?	?	?	?

VECTORES - AGREGAR

```
Procedure agregar (var a :números; var dL:integer; var pude:boolean; num:integer);

 Begin
   pude:= false;
                    Verifico si hay espacio
   if ((dl + 1) <= física) then
    begin
      pude:= true;          Registro que se pudo realizar
      dL:= dL + 1;          Incremento la dimensión lógica
      a[dL]:= num;          Agrego elelemento
    end;
 end.
```

CADP – TIPOS DE DATOS

VECTORES -INSERTAR

Significa agregar en el vector un elemento en una posición determinada. Puede pasar que esta operación no se pueda realizar si el vector está lleno o si la posición no es válida

Dl = 4

45

pos = 2

a

34	10	-1	5						
1	2	3	4	5	6	7	8	9	10

Qué pasos considero?

VECTORES -INSERTAR

Significa agregar en el vector un elemento en una posición determinada. Puede pasar que esta operación no se pueda realizar si el vector está lleno o si la posición no es válida

1- **Verificar** si hay espacio (cantidad de elementos actuales es menor a la cantidad de elementos posibles)

2- **Verificar** que la posición sea válida (esté entre los valores de dimensión definida del vector y la dimensión lógica).

3- **Hacer lugar** para poder insertar el elemento.

4- **Incrementar** la cantidad de elementos actuales.

Cómo se implementa?

CADP – TIPOS DE DATOS

VECTORES - INSERTAR

Dado un vector de números enteros (10 elementos como máximo) realice un programa que lea un nuevo número y una posición e invoque a un módulo que inserte el elemento en el vector en la posición leída.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
 VN: numeros;
 dimL, valor,pos:integer;
 ok:boolean;
Begin
 cargar (VN,dimL);
 read(valor); read(pos);
 insertar(VN,dimL,ok,valor,pos);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 4

VN

4	-1	10	3	?	?	?	?	?	?

valor = 7 pos= 2 dimL = 5 ok = true

VN

4	7	-1	10	3	?	?	?	?	?

CADP – TIPOS DE DATOS

VECTORES - INSERTAR

```
Procedure insertar (var a :números; var dL:integer; var pude:boolean;
                                num:integer; pos: integer);
Var
 i:integer;

Begin
  pude:= false;

  if ((dL + 1) <= física) and (pos>= 1) and (pos <= dL) )then begin

   for i:= dL downto pos do
        a[i+1]:= a[i];

   pude:= true;
   a[pos]:= num;
   dL:= dL + 1;
  end;
end;
```

Verifico si hay espacio y si la posición es válida

Corro los elementos empezando desde atrás hasta la posición a insertar para hacer el hueco donde se va a insertar el elemento

Registro que se pudo realizar
Inserto el elemento
Incremento la dimensión lógica

VECTORES -ELIMINAR

Significa borrar (lógicamente) en el vector un elemento en una posición determinada, o un valor determinado.

Puede pasar que esta operación no se pueda realizar si la posición no es válida, o en el caso de eliminar un elemento si el mismo no está

pos = 2

DL = 4

a

34	10	-1	5						
1	2	3	4	5	6	7	8	9	10

Qué pasos considero?

VECTORES -ELIMINAR

Significa borrar (lógicamente) en el vector un elemento en una posición determinada, o un valor determinado. Puede pasar que esta operación no se pueda realizar si la posición no es válida, o en el caso de eliminar un elemento si el mismo no está

1- Verificar que la posición sea válida (esté entre los valores de dimensión definida del vector y la dimensión lógica).

2- Hacer el corrimiento a partir de la posición y hasta el final.

3- Decrementar la cantidad de elementos actuales

Cómo se implementa?

CADP – TIPOS DE DATOS

VECTORES - ELIMINAR

Dado un vector de números enteros (10 elementos como máximo) realice un programa que lea una posición e invoque a un módulo que elimine el elemento en el vector en la posición leída.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
 VN: numeros;
 dimL,pos:integer;
 ok:boolean;
Begin
 cargar (VN,dimL);
 read(pos);
 eliminar(VN,dimL,ok,pos);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 4

VN

4	-1	10	3	?	?	?	?	?	?

pos= 2 dimL = 3 ok = true

VN

4	10	3	3	?	?	?	?	?	?

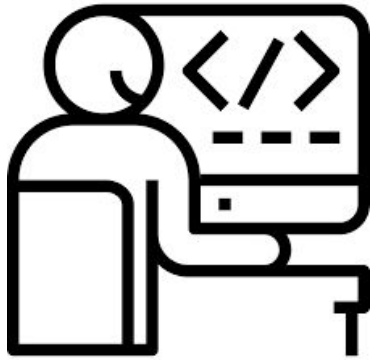
VECTORES - ELIMINAR

```
Procedure eliminar (var a :números; var dL:integer; var pude:boolean;pos: integer);
Var
 i:integer;

Begin
  pude:= false;
  if ((pos>= 1) and (pos <= dL) )then begin
    for i:= pos to (dL-1) do
       a[i]:= a[i+1];
    pude:= true;
    dL:= dL - 1;
  end;
end;
```

Verifico si la posición es válida

Corro los elementos empezando desde la posición hasta la dimensión lógica-1 para "tapar" el elemento a eliminar

Registro que se pudo realizar
Decremento la dimensión lógica



Conceptos de Algoritmos Datos y Programas

CADP – **TEMAS**

- Estructura de datos ARREGLO
- Búsqueda en un vector desordenado
- Búsqueda en un vector ordenado

VECTORES - OPERACIONES

Carga de valores

Lectura / Escritura

Recorridos

Dimensión física y lógica

Agregar elementos

Insertar elementos

Borrar elementos

Búsqueda de un elemento

VECTORES - BUSQUEDAS

Significa recorrer el vector buscando un valor que puede o no estar en el vector. Se debe tener en cuenta que no es lo mismo buscar en un vector ordenado que en uno que no lo este.

Vector Desordenado

- Se debe recorrer todo el vector (en el peor de los casos), y detener la búsqueda en el momento que se encuentra el dato buscado o en el que se terminó el vector.

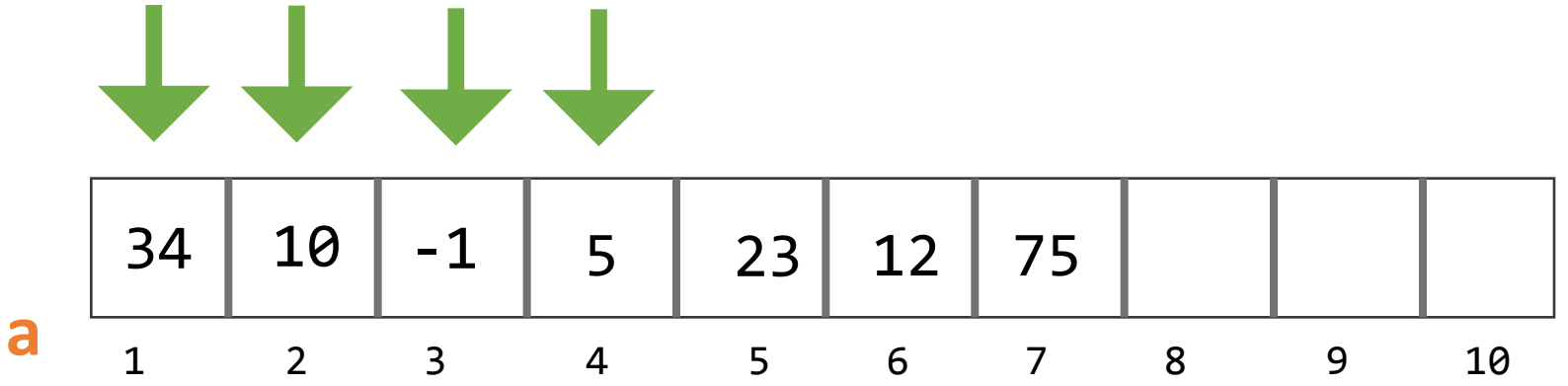
Vector Ordenado

- Se debe recorrer el vector teniendo en cuenta el orden:
  - BUSQUEDA MEJORADA
  - BUSQUEDA BINARIA

Vector Desordenado

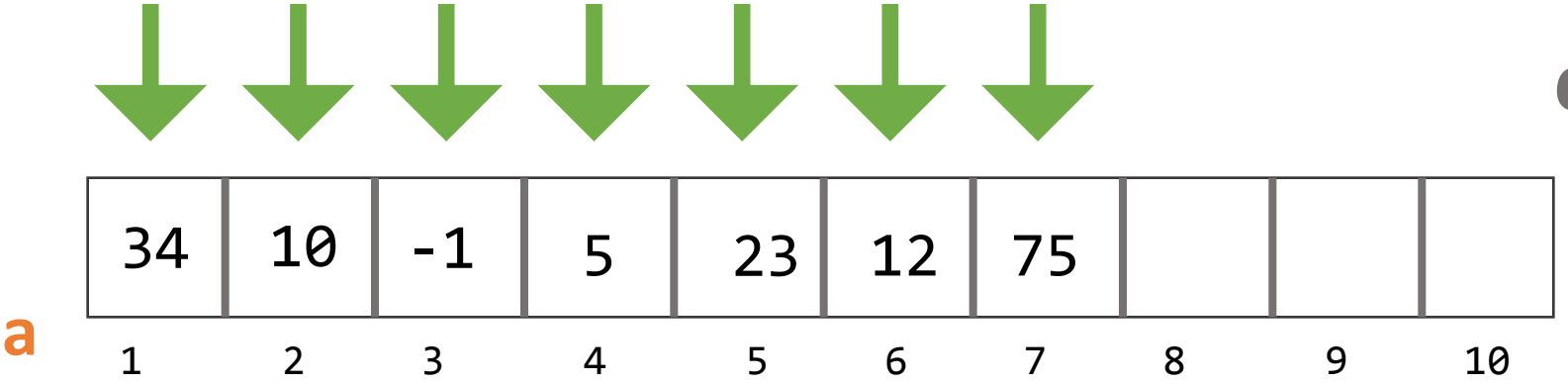
Dl = 7

5



Dl = 7

25



Qué pasos considero?

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

Se debe recorrer todo el vector (en el peor de los casos), y detener la búsqueda en el momento que se encuentra el dato buscado o en el que se terminó el vector.

1- Inicializar la búsqueda desde la posición 1 (pos).

2- Mientras ((el elemento buscado no se igual al valor en el arreglo[pos]) y (no se termine el arreglo))
   2.1 Avanzo una posición

3- Determino porque condición se ha terminado el while y devuelvo el resultado.

Cómo se implementa?

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

Dado un vector de números enteros (10 elementos como máximo) realice un programa que lea un nuevo número y determine si el valor se encuentra en el vector.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
    VN: numeros;
    dimL, valor:integer;
    ok:boolean;

Begin
  cargar (VN,dimL);
  read(valor);
  res:= buscar(VN,dimL,valor);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 5

VN

4	-1	10	3	7	?	?	?	?	?

valor = 10 dimL = 5 ok = true

VN

4	-1	10	3	7	?	?	?	?	?

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

```
function buscar (a :números; dL:integer; valor:integer): boolean;
 Var
   pos:integer;

 Begin
   pos:=1;
   while ( (pos <= dL ) and (a[pos] <> valor) ) do
      begin
        pos:= pos + 1;
      end;
    buscar:= (a[pos] = valor);
end.
```

Es correcto?

Si el elemento no está, pos en este caso quedaría en 11, y en la última línea de la función estaría asignando el resultado de comparar a[11] = valor

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

```
function buscar (a :números; dL:integer; valor:integer): boolean;
 Var
   pos:integer;

 Begin
   pos:=1;
   while ((a[pos] <> valor) and (pos <= dL) ) do
      begin
        pos:= pos + 1;
      end;
    buscar:= (a[pos]=valor);
end.
```

Es correcto?

Si el elemento no está, pos en este caso quedaría en 11, y en en el while se pregunta a[11] y no es válido

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

```
function buscar (a :números; dL:integer; valor:integer): boolean;
 Var
   pos:integer;

 Begin
   pos:=1;
   while ((pos <= dL) and (a[pos] <> valor)  ) do
      begin
        pos:= pos + 1;
      end;
    buscar:= (pos <= dL);
 end.
```

Es correcto?

Si pos no es <= dL no significa que haya estado el elemento

CADP – TIPOS DE DATOS VECTORES – BUSCAR DESORDENADO

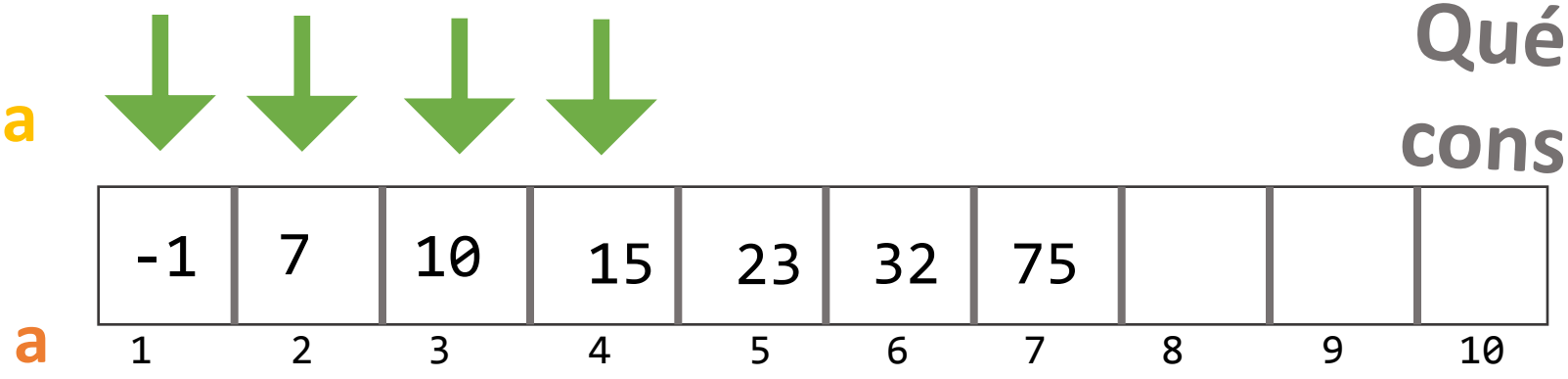
```
function buscar (a :números; dL:integer; valor:integer): boolean;
 Var
   pos:integer;
   esta:boolean;

 Begin
   esta:= false;
   pos:=1;
   while ( (pos <= dL) and (not esta) ) do
      begin
        if (a[pos]= valor) then esta:= true
        else
           pos:= pos + 1;
      end;
    buscar:= esta;
 end.
```

VECTORES - BUSQUEDAS

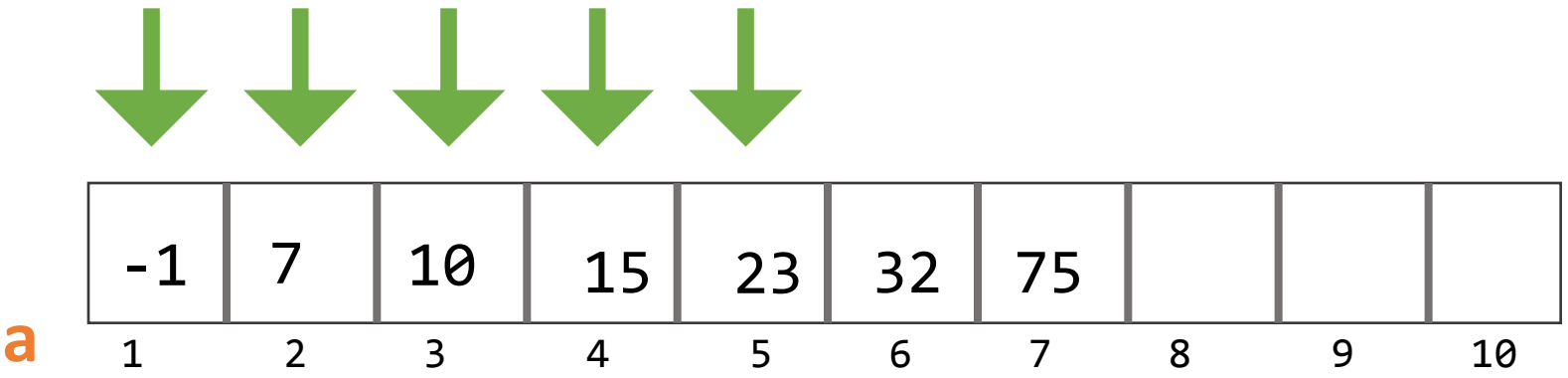
Vector Ordenado
Búsqueda Mejorada

Qué pasos considero?

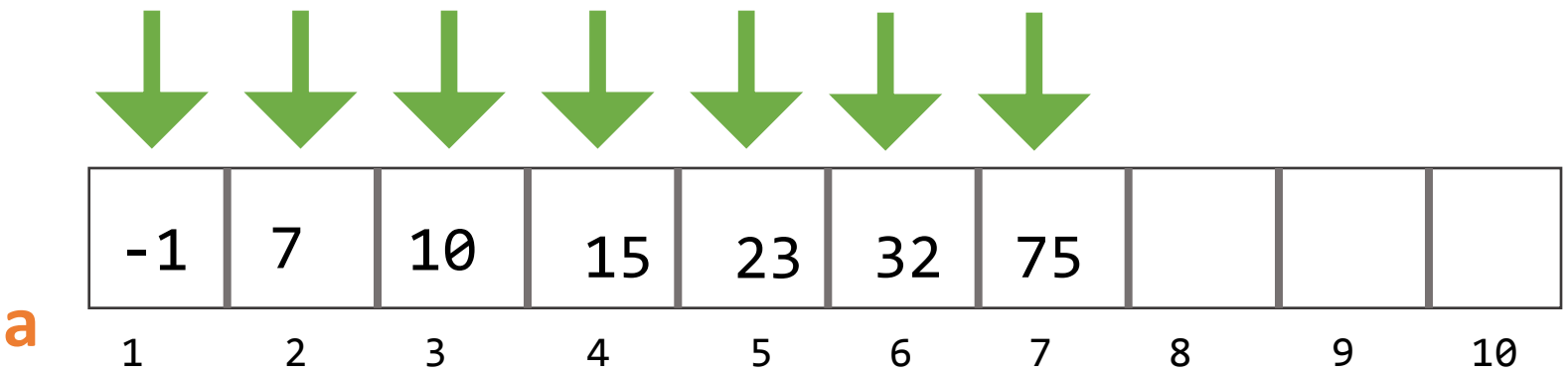
```
Dl = 7
15
```




```
Dl = 7
16
```




```
Dl = 7
80
```



BUSQUEDA MEJORADA

1- Inicializar la búsqueda desde la posición 1 (pos).

2- Mientras ((el elemento buscado sea menor al valor en el arreglo[pos]) y (no se termine el arreglo))
   2.1 Avanzo una posición

3- Determino porque condición se ha terminado el while y devuelvo el resultado.

Cómo se implementa?

CADP – TIPOS DE DATOS

Dado un vector de números enteros (10 elementos como máximo) ordenado realice un programa que lea un número e invoque a un módulo que retorne si el número se encuentra en el vector.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
 VN: numeros;
 dimL,pos:integer;
 ok:boolean;
Begin
  cargar (VN,dimL);
  read(valor);
  ok:= existe(VN,dimL,valor);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 4

VN

4	-1	10	3	?	?	?	?	?	?

valor= -1 dimL = 4 ok = true

VN

4	-1	10	3	?	?	?	?	?	?

VECTORES - INSERTAR

```
Function existe (a:números; dL:integer; valor:integer):boolean;

Var
  pos:integer;

Begin
  pos:=1;

  while ( (pos <= dL) and (a[pos]< valor)) do
     begin
       pos:= pos + 1;
     end;

  if ( (pos <= dL) and (a[pos]= valor)) then buscar:=true
  else buscar:= false;
end.
```

Importa el orden en la condición del while?

Alcanza con preguntar por sólo una de las dos condiciones?

VECTORES - BUSQUEDAS

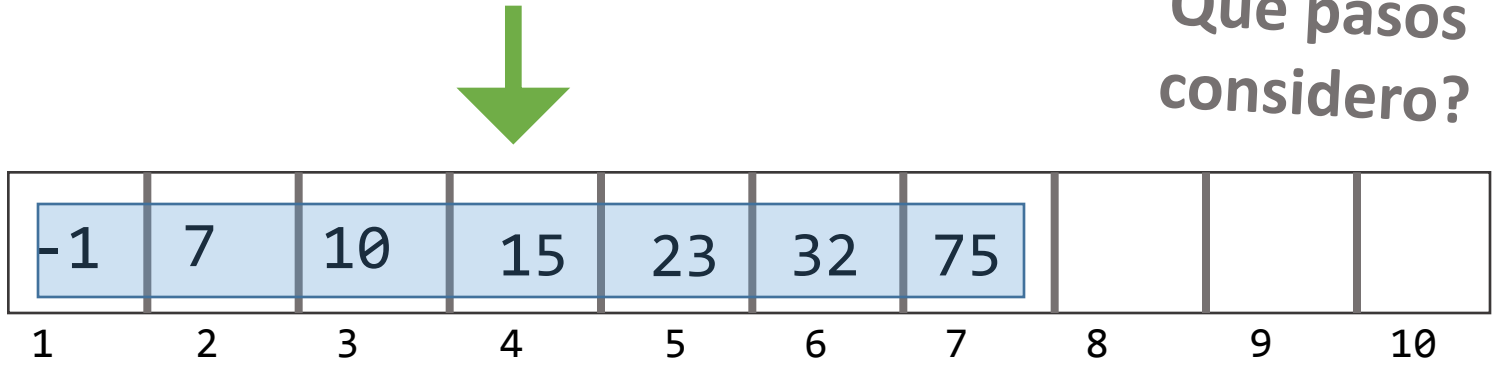
Vector Ordenado
Búsqueda DICOTOMICA

Qué pasos considero?

Dl = 7

10

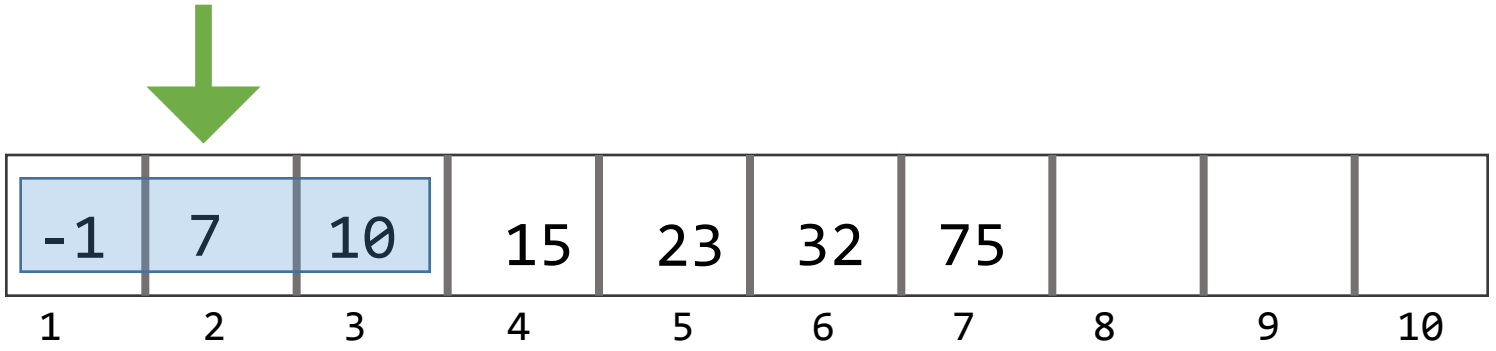
Inf 1
Sup 7
Medio 4



Dl = 7

10

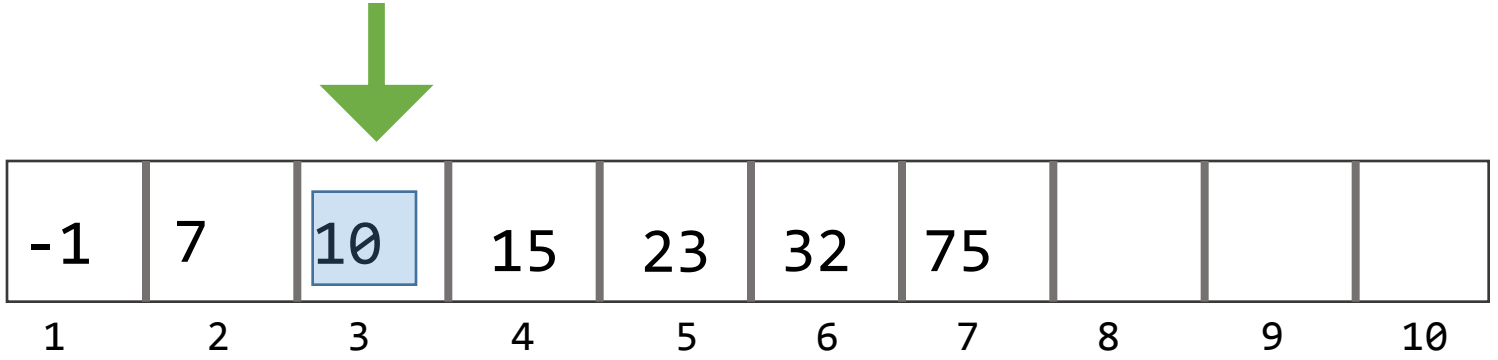
Inf 1
Sup 3 (medio-1)
Medio 2



Dl = 7

80

Inf 3(medio+1)
Sup 3
Medio 3



CADP – TIPOS DE DATOS VECTORES – BUSCAR ORDENADO

BUSQUEDA DICOTOMICA

Cómo se implementa?

1- Se calcula la posición media del vector (teniendo en cuenta la cantidad de elementos)

2- Mientras ((el elemento buscado sea <> arreglo[medio]) y (inf <= sup))

    Si ((el elemento buscado sea < arreglo[medio]) entonces
      Actualizo sup
    Sino
      Actualizo inf
    Calculo nuevamente el medio

3- Determino porque condición se ha terminado el while y devuelvo el resultado.

CADP – TIPOS DE DATOS

Dado un vector de números enteros (10 elementos como máximo) ordenado realice un programa que lea un número e invoque a un módulo que retorne si el número se encuentra en el vector.

```
Program uno;
  const
    fisica = 10;
  type
    numeros= array [1..fisica]  of integer;

var
 VN: numeros;
 dimL,pos:integer;
 ok:boolean;
Begin
  cargar (VN,dimL);
  read(valor);
  ok:= dicotomica(VN,dimL,valor);
End.
```

VN

?	?	?	?	?	?	?	?	?	?

dimL = 4

VN

4	-1	10	3	?	?	?	?	?	?

valor= -1 dimL = 4 ok = true

VN

4	-1	10	3	?	?	?	?	?	?

VECTORES - INSERTAR

```
Function dicotomica (a:números; dL:integer; valor:integer):boolean;
  Var
     pri, ult, medio : integer;
     ok:boolean

  Begin
      ok:= false;
      pri:= 1 ;  ult:= dL;  medio := (pri + ult ) div 2 ;

      While  ( pri < = ult ) and ( valor <> vec[medio]) do
        begin
          if ( valor < vec[medio] ) then
            ult:= medio -1 ;
          else pri:= medio+1 ;
          medio := ( pri + ult ) div 2 ;
        end;
      if (pri <=ult) and (valor = vec[medio]) then ok:=true;
    end;
   dicotomica:= ok;
end.
```