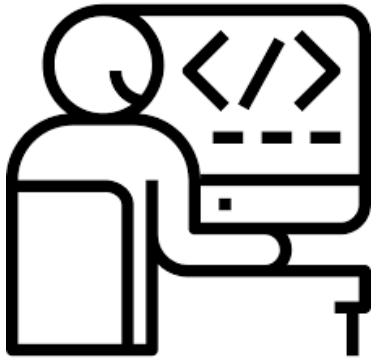


Conceptos de Algoritmos Datos y Programas

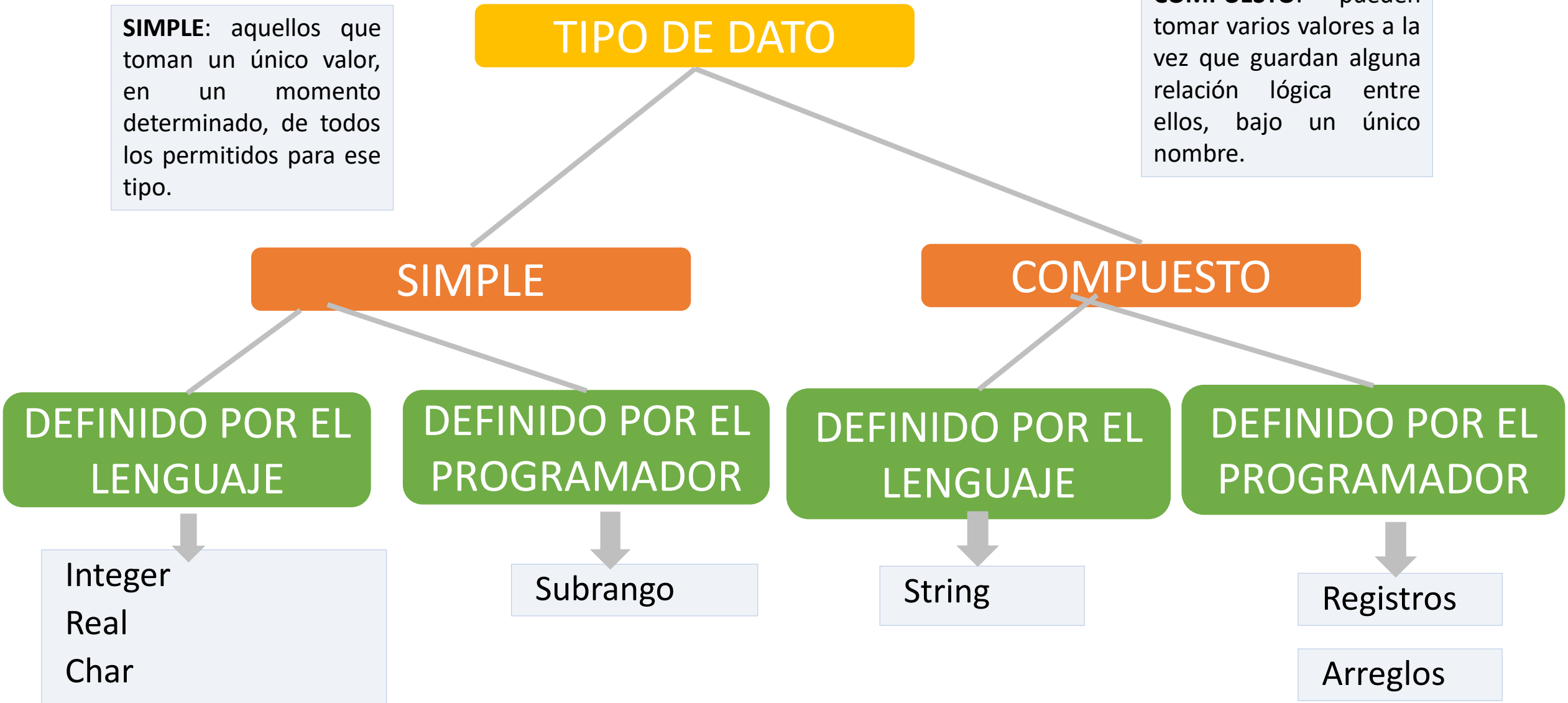
CADP – **TEMAS**

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

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.

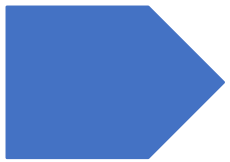


ARREGLOS

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 se 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 mas valores se lean el problema es mas 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 mas valores se lean el problema es mas 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

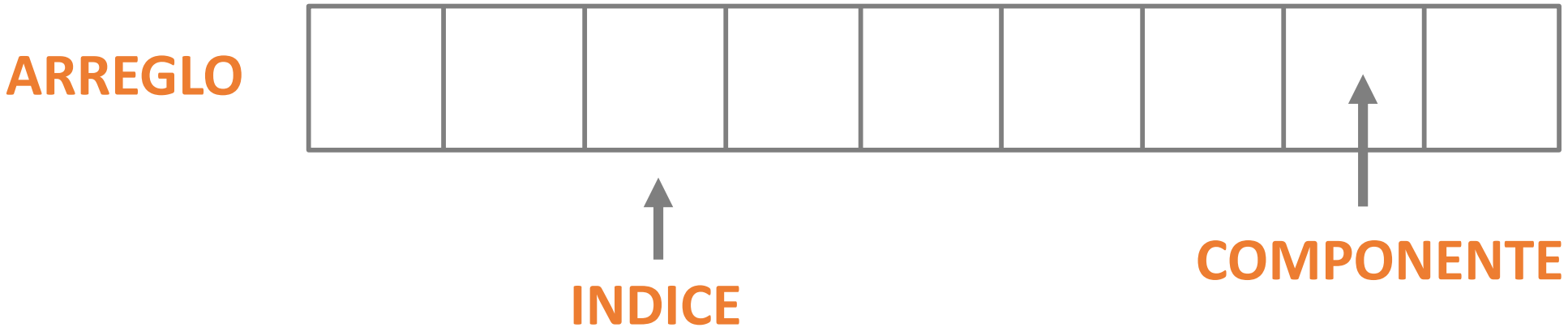
20	77	68	2	77	23	4	15	15	3

2

ARREGLOS

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.



ARREGLOS

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 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 (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.

HOMOGENEA → Los elementos pueden son del mismo tipo .

ESTATICA → El tamaño 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

```
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

```
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

ARREGLOS - OPERACIONES

Carga de valores

Lectura / Escritura

Recorridos

Agregar elementos al final

Insertar elementos

Borrar elementos

Búsqueda de un elemento

Ordenación de los elementos

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]`

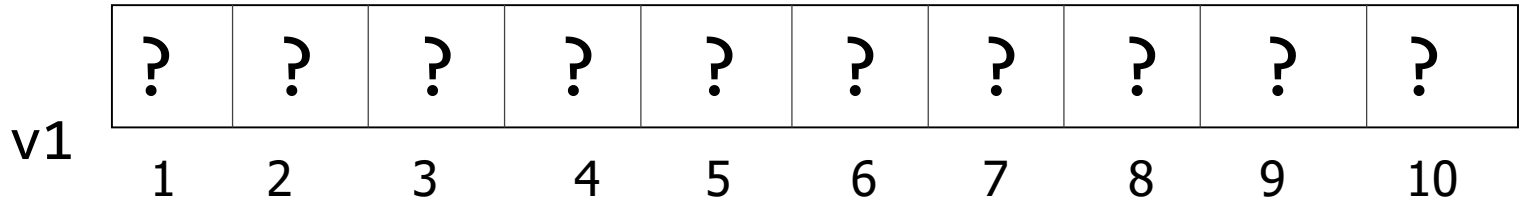
`variable [4]`

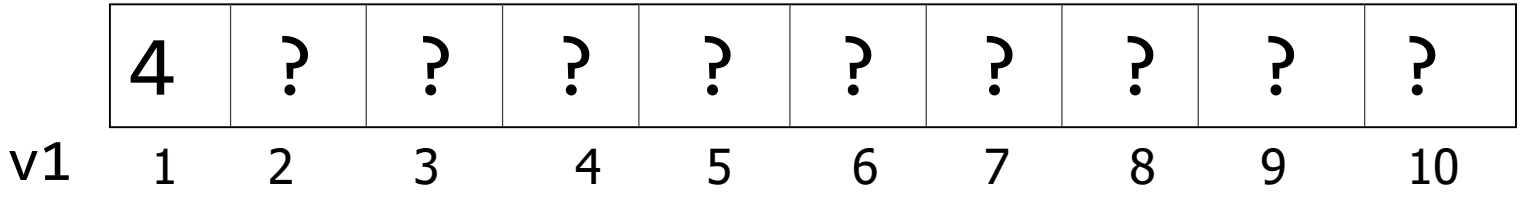
Las operaciones con el elemento son las permitidas para el tipo de datos del elemento

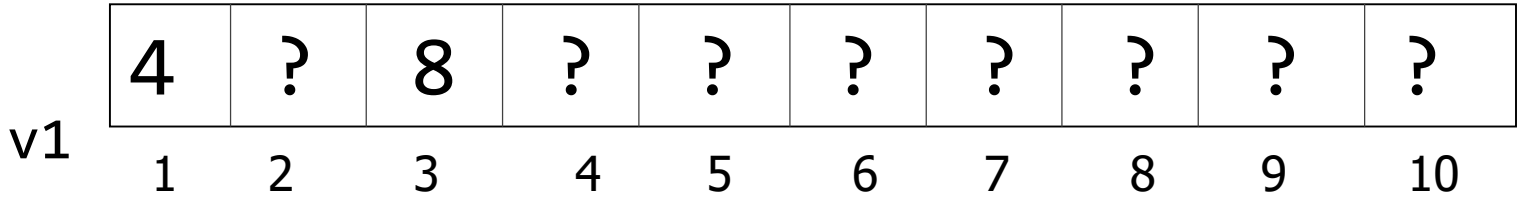
CADP – TIPO DE DATOS VECTOR - CARGA

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

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;
```

v

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

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.

Cargar vector (v)
Calcular el máximo (v , max)
Verificar cuantas veces aparece max en el vector v

Cómo cargo el vector?

Cómo calculo el máximo?

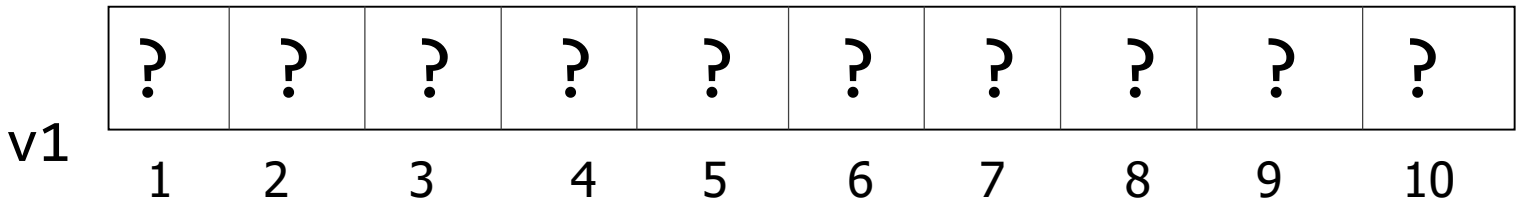
Cómo verifico cuántas veces apareció el maximo?

v

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

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

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

CADP – TIPO DE DATOS VECTOR -

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

  Var
     i,max,valor:integer;

     Begin
       max:= -9990;
       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;
```

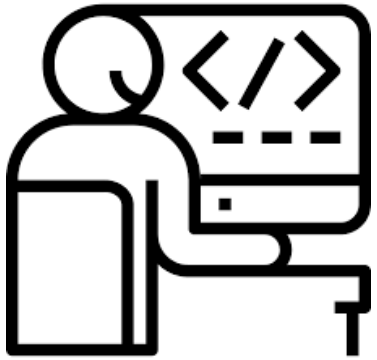
v

-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;
```



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.

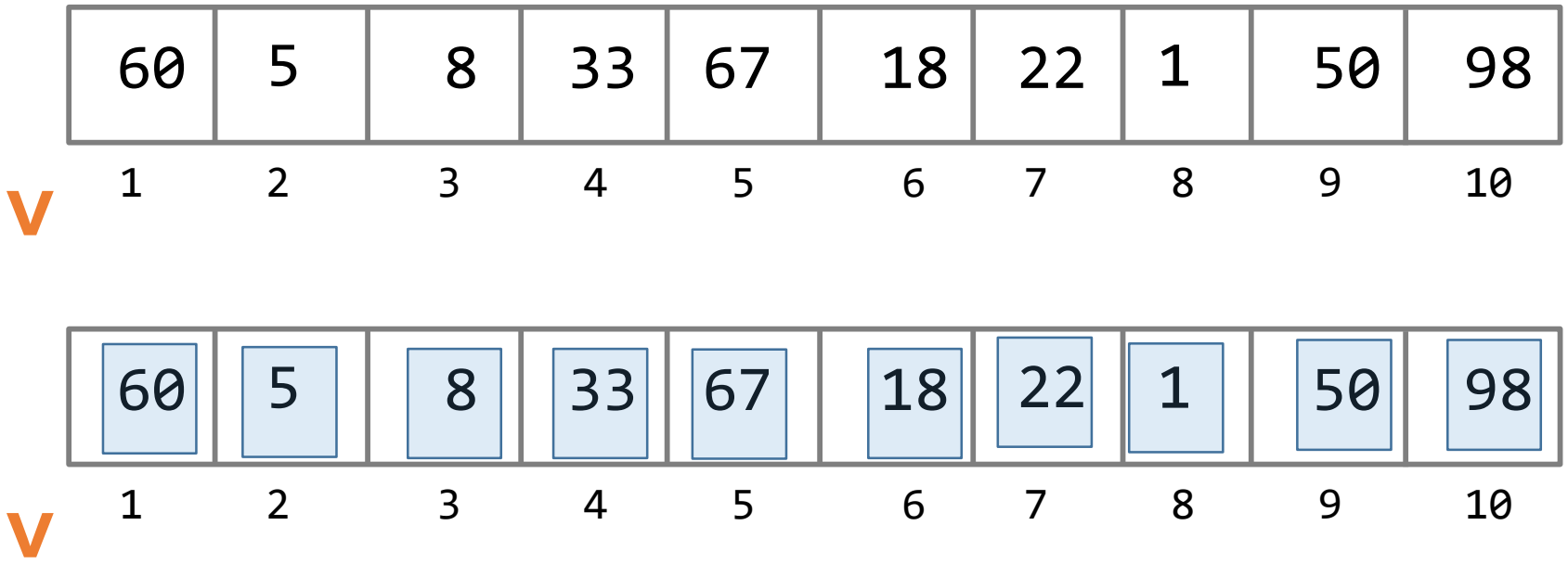
RECORRIDO - PARCIAL

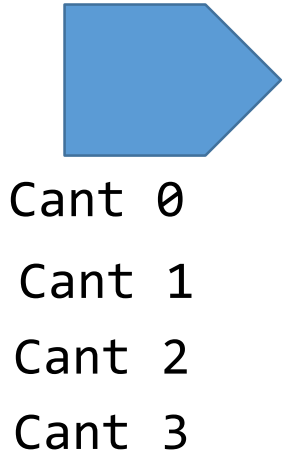
Implica analizar los elementos del vector, hasta encontrar aquel que cumple con lo pedido. Puede ocurrir que se recorra todo el vector

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.





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.
```

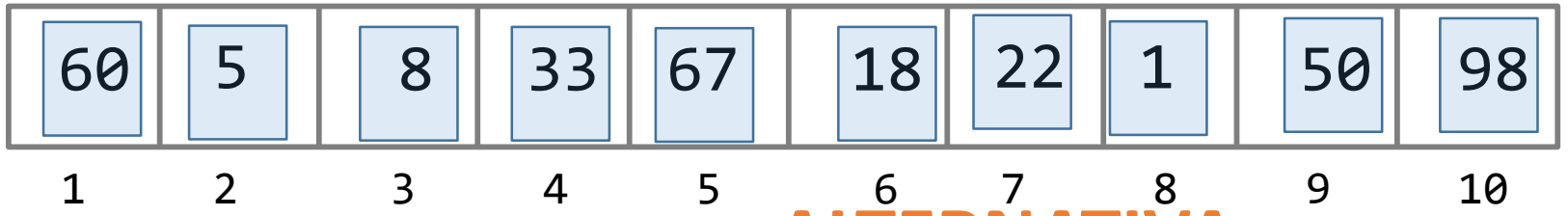
```
Procedure cargar (var v:vector);
Var
  i,valor:integer;

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;
```

ARREGLOS - RECORRIDOS

```
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

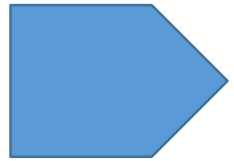
```
function multiplos (v:vector):integer;
Var
  i,cant: integer;

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

ARREGLOS

Realice un programa que llene 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

60	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?

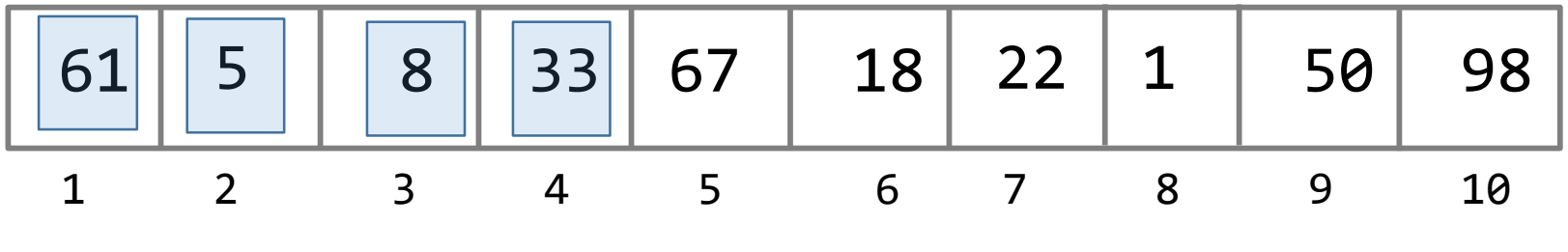
CADP – **VECTOR** RECORRIDO - PARCIAL

```
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;
```



Por qué se inicializa pos en 1?

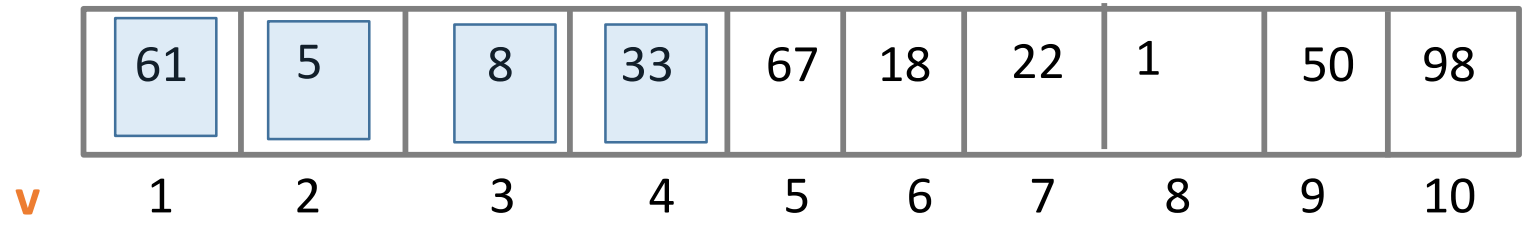
Por qué pos se incrementa en el else?

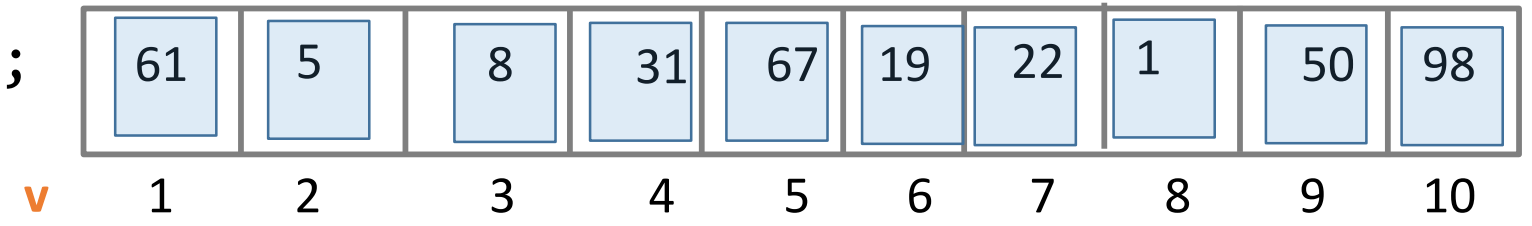
Qué cambio si el enunciado no asegura que haya al menos un múltiplo de 3?

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?