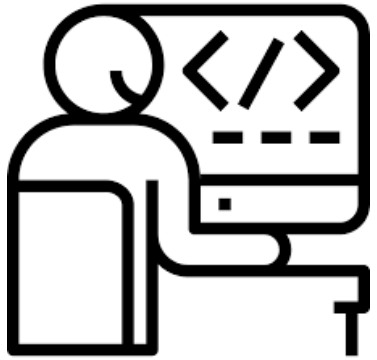


Conceptos de Algoritmos Datos y Programas

CADP – **Temas de la clase de hoy**

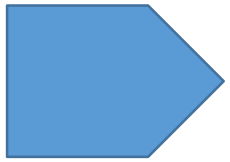
- Tipos de Datos estructurados
- Tipos de Datos Arreglo

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

Realizar un programa que lea 7 números que representan edades, y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. Cómo hacemos? Algunas soluciones…

Edades Leídas

20
98
68
2
98
23
4



Máximo 98

Y ahora que se que la edad máxima es 98 cómo informo cuantas veces apareció?

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

Realizar un programa que lea 7 números que representan edades, y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. Cómo hacemos? Algunas soluciones…

Solución 1

Ingresar los valores.
Calcular el máximo.
Ingresar los valores nueva
imprimir cuáles coincid
calculado.

PROBLEMA: se debe
antizar que el usuario
e los mismos
Cuantos mas
se lean el
a es mas grande.

SOLUCION

Solución 2

Ingresar los valores y guar
una variable.
Calcular el máximo.
Comparar cada variable con el máximo calculado.

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

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

Realizar un programa que lea 7 números que representan edades, y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. Cómo hacemos? Algunas soluciones…

Disponer de alguna ESTRUCTURA donde almacenar los números, para luego calcular el promedio, y así finalmente poder compararlos

Leer los números y almacenarlos

20	98	68	2	98	23	4

Recorrer la estructura y obtener el máximo

20	98	68	2	98	23	4

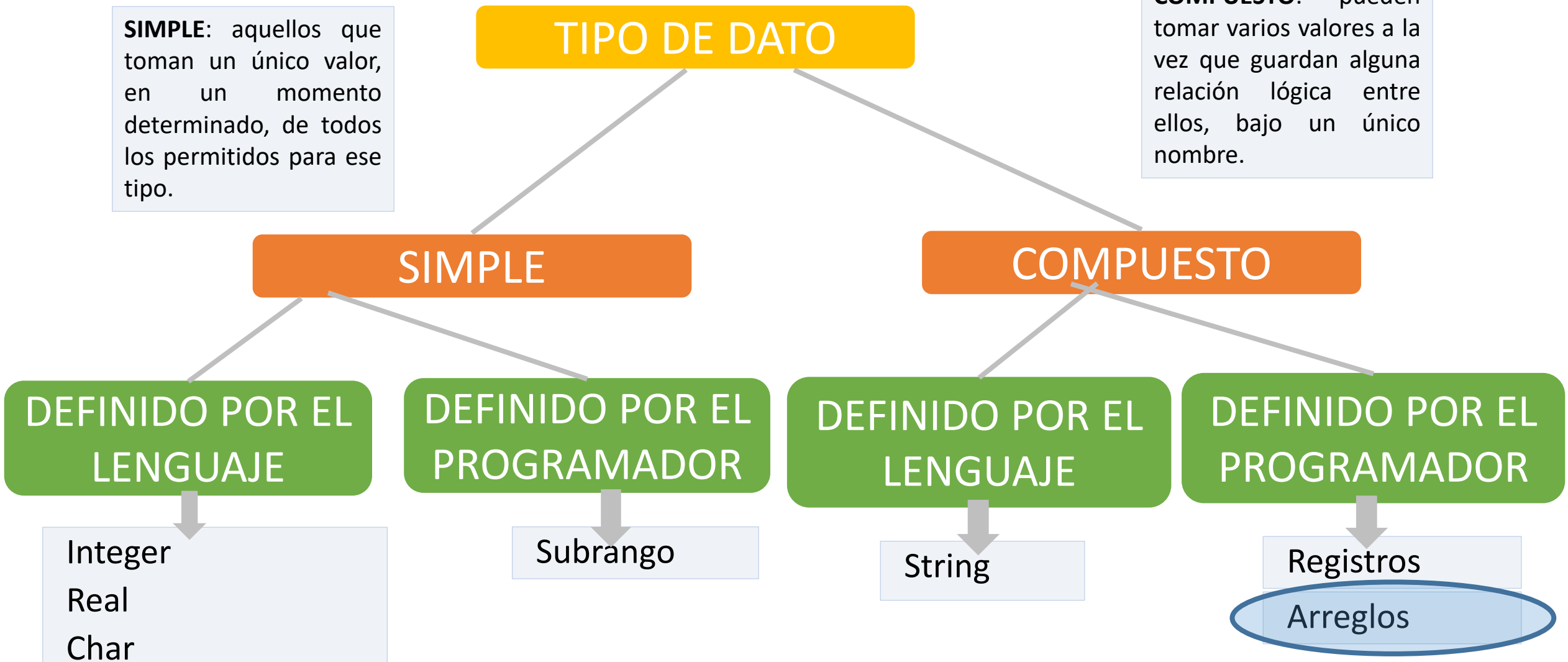
Recorrer la estructura y comparar con el máximo

20	98	68	2	98	23	4

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

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

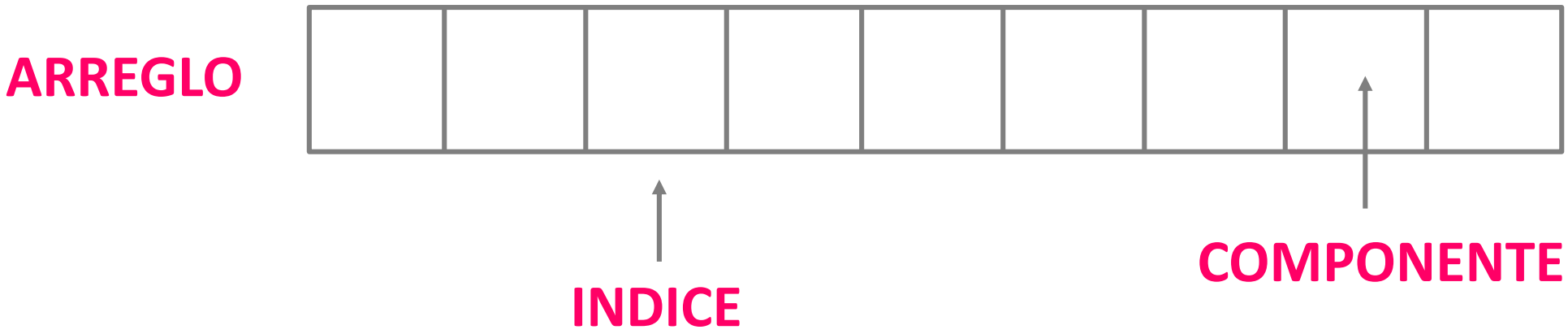
COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.



CADP – TIPOS DE DATOS ESTRUCTURADOS 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.



CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

VECTOR

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.

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

VECTOR

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

CARACTERISTICAS

Los elementos son del mismo tipo. Precisamente por ser **estática**, permite el acceso rápido a sus componentes a través de la variable **índice** (que tiene que ser de tipo ordinal) y que puede verse como el desplazamiento desde la posición inicial de comienzo de la estructura.

Cómo se declaran?

Cómo se usan?

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

VECTOR

```
Program uno;

Type
  vector = array [rango]   of   tipo;
```

Nombre del tipo

El rango debe ser de tipo ordinal.
Integer
Char
Boolean
Subrango

El tipo debe ser estático
Integer
Real
Char
Boolean
Subrango
Registro
Vector

CADP – **TIPOS DE DATOS ESTRUCTURADOS** Vector

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

CADP – **VECTOR**

Operaciones

Carga de valores

Lectura / Escritura

Recorridos

Agregar elementos al final

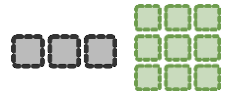
Insertar elementos

Borrar elementos

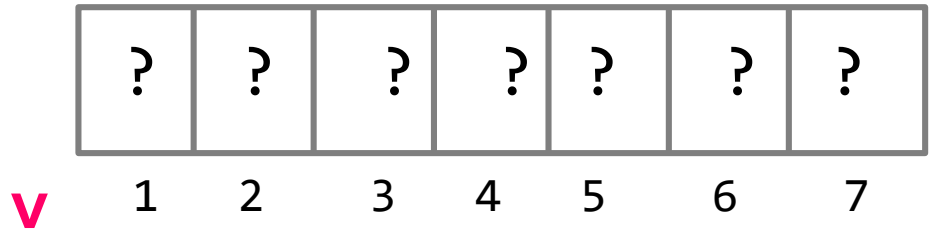
Búsqueda de un elemento

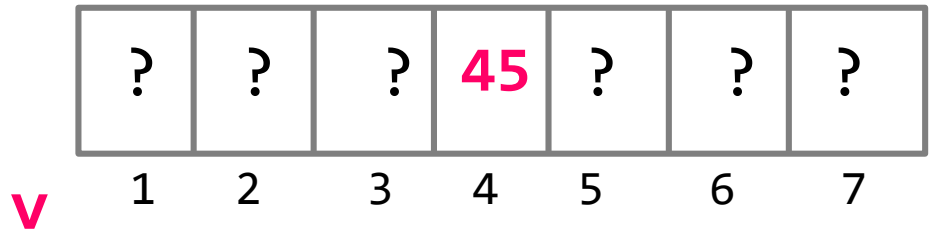
Ordenación de los elementos

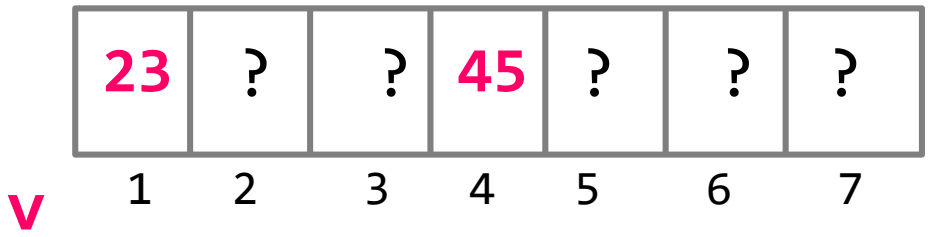
Operaciones – Carga de valores



```
Program uno;
Const
 tam = 7;
Type
   vector = array [1..tam] of   integer;
Var
 v:vector;

Begin

 v [4]:= 45;

 v [1]:= 23;
```





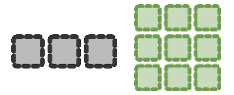


Cómo se podría representar el tamaño?

Cómo se carga de manera completa?

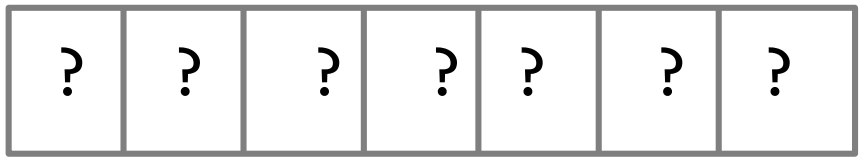
CADP – **VECTOR**

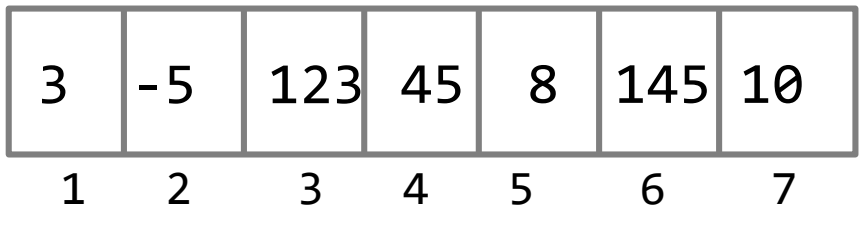
Operaciones – Carga de valores



```
Program uno;
 Const
  tam = 7;
 Type
   vector = array [1..tam]   of   integer;
Var
 v:vector;
 i,valor:integer;

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

No se puede hacer read(v)



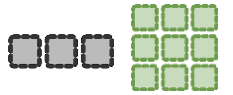



```
Begin
 for i:= 1 to tam do
  begin
   read(v[i]);
  end;
End.
```

Cómo se modulariza?

CADP – **VECTOR**

Operaciones – Carga de valores



Puede ser una función?

Se conoce tam?

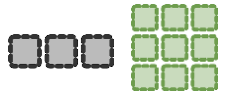
```
Procedure cargar (var datos: vector);

 Var
  i,valor:integer;

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

OPCION 2

```
Procedure cargar (var datos: vector);

 Var
  i:integer;

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

CADP – **VECTOR**

Operaciones – Carga de valores

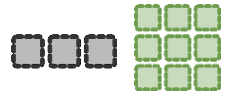


```
Program uno;
Const
 tam=7;

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

procedure cargar (var datos:vector);
  begin
   …
  end;

Var
 v:vector;

Begin
 cargar (v);
End.
```

Cómo muestro los datos?

CADP – **VECTOR**

Operaciones – Mostrar valores



```
Procedure imprimir (datos: vector);

 Var
  i,valor:integer;

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

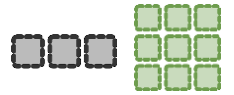
Puede ser una función?

OPCION 2

```
Procedure imprimir (datos: vector);

 Var
  i:integer;

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

CADP – **VECTOR**

Operaciones – Muestra de valores



```
Program uno;
Const
 tam=7;
Type
   vector = array [1..tam]   of   integer;

procedure cargar (var datos:vector);
  begin
   …
  end;
procedure imprimir (datos:vector);
  begin
   …
  end;
Var
 v:vector;
Begin
 cargar (v);
 imprimir (v);
End.
```

Ahora como solucionamos el problema planteado inicialmente?

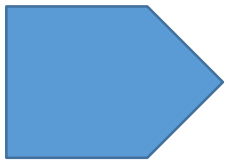
CADP – **VECTOR**

Operaciones – Muestra de valores

Realizar un programa que lea 7 números que representan edades, y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. Cómo hacemos? Algunas soluciones…

Edades Leídas

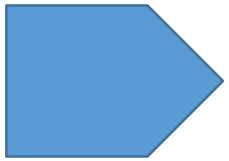
20
98
68
2
98
23
4



Máximo calculado
98

Edades Almacenadas

20
98
68
2
98
23
4



Informo 2

Qué módulos hacemos?

Qué necesita recibir cada módulo?

Qué devuelve cada módulo?

Cargar el vector
Calcular el máximo
Contar cuantas veces aparece el máximo en el vector

CADP – **VECTOR**

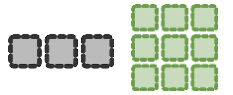
EJERCICIO

Realizar un programa que lea 7 números que representan edades, y luego de realizar la lectura se quiere informar cuántos veces apareció la edad más grande. Cómo hacemos? Algunas soluciones…

```
Porgram uno;
Const
 tam=7;
Type
 vector = array[1..tam]of integer;

//módulos

Var
 v:vector;
Begin
 cargar(v);
 max:= máximo(v);
 cant:= cantidad (v,max);
 write (“La cantidad de veces que aparece ”, max, “es”,cant);
End.
```

CADP – **VECTOR**

EJERCICIO

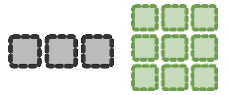


```
Procedure cargar (var datos:vector);
Var
  i,valor:integer;

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

OPCION 2

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

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

EJERCICIO



```
function maximo (datos:vector):integer;
Var
  i,max: integer;

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

Es obligatorio usar max?

CADP – **VECTOR**

EJERCICIO

```
function cantidad (datos:vector; maxi:integer):integer;
Var
  i,cant: integer;

Begin
  cant:=0;
  for i:= 1 to tam do
   begin
     if (datos[i] = maxi) then
       cant:= cant + 1;
   end;
  cantidad:= cant;
End;
```

Es obligatorio usar cant?

Cómo puedo reescribir el programa aprovechando las ventajas de las funciones?

Escribir un programa que lea 5 números, los almacene y luego informe la multiplicación de todos.

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

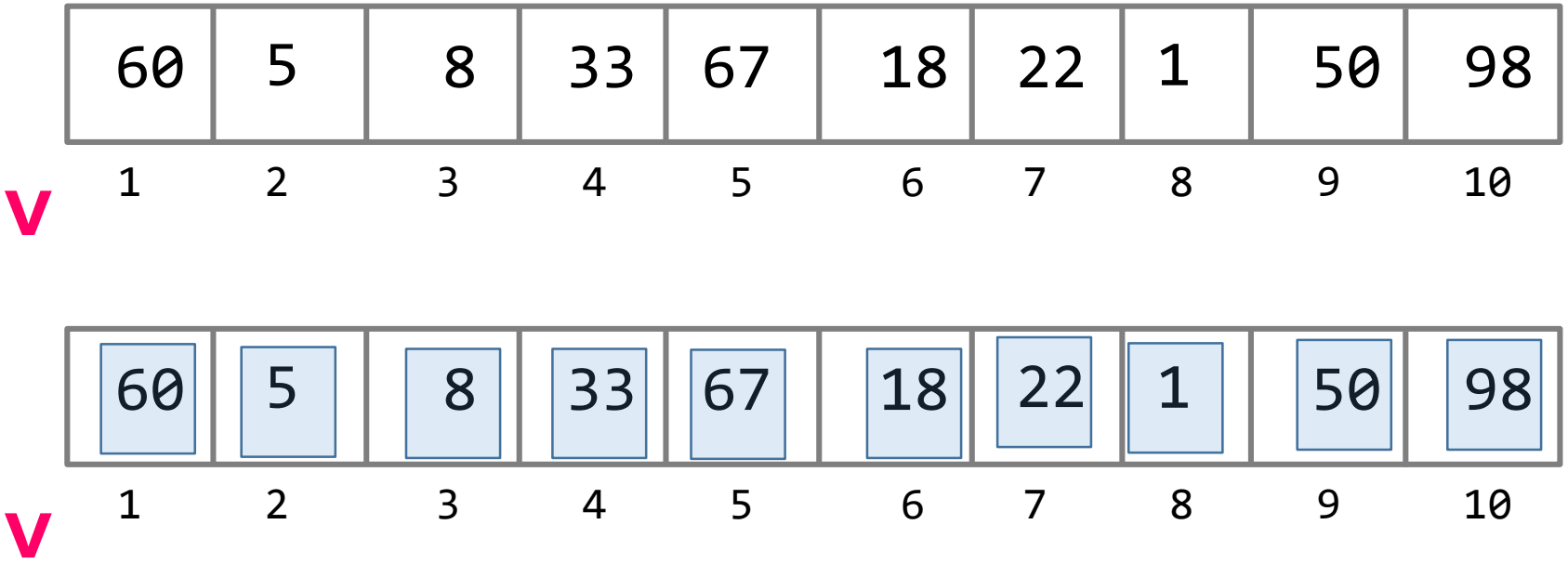
Implica analizar todos los elementos del vector, lo que lleva a recorrer completamente la estructura.

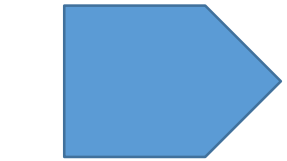
RECORRIDO - PARCIAL

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

RECORRIDO - TOTAL

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

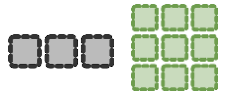
Cant 1

Cant 2

Cant 3

Cómo lo implemento?

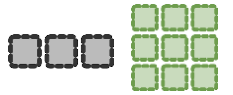
CADP – **VECTOR** RECORRIDO - TOTAL



```
Program uno;

Const
 tam=10;
 multi=3;

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

Var
 v:vector;
 cant:integer;

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

CADP – **VECTOR**

EJERCICIO



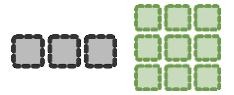
```
Procedure cargar (var datos:vector);
Var
  i,valor:integer;

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

OPCION 2

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

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

CADP – **VECTOR**

EJERCICIO



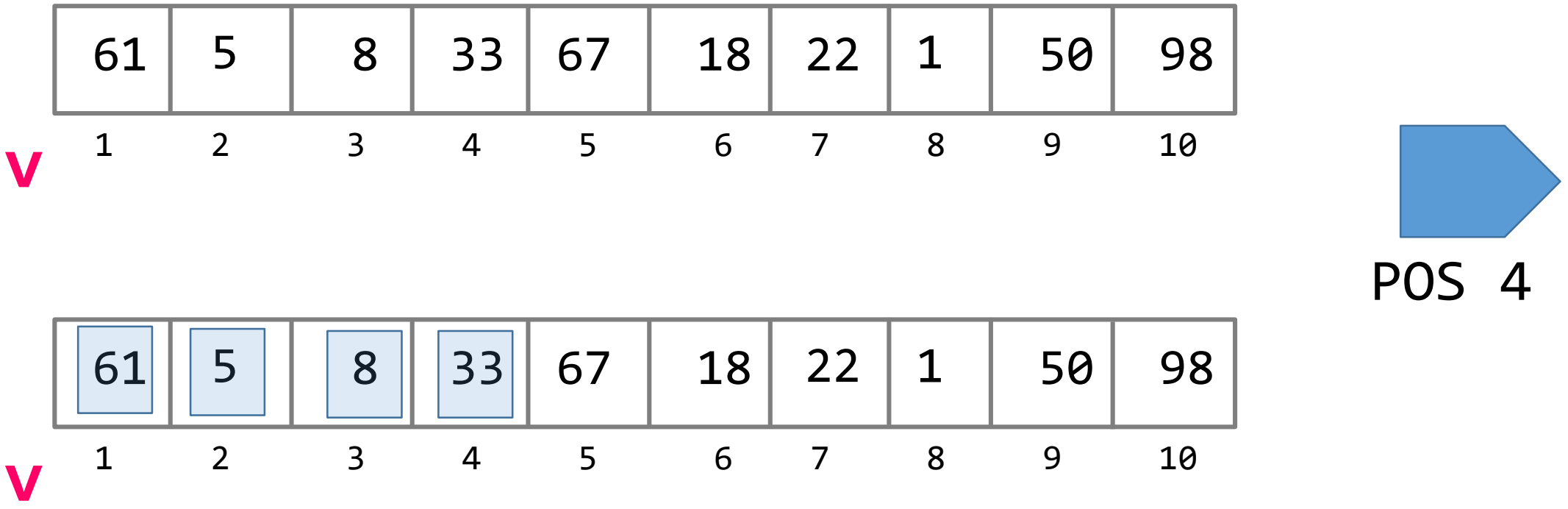
```
function multiplos (datos:vector):integer;
Var
  i,cant,resto: integer;

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

OPCION 2

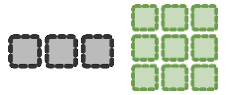
```
function multiplos (datos:vector):integer;
Var
  i,cant: integer;

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

CADP – **VECTOR** RECORRIDO - PARCIAL

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.



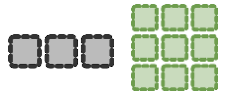
CADP – **VECTOR** RECORRIDO - PARCIAL



```
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 (datos: vector): integer;
var
  pos,resto:integer;
  seguir:boolean;

begin
  seguir:= true; pos:=1;
  while (seguir = true) do
    begin
      resto:= datos[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?

RECORRIDO PARCIAL

```
function posicion (datos: vector): integer;
var
  pos,resto:integer;
  seguir:boolean;
begin
  seguir:= true; pos:=1;
  while ((pos<= tam)and (seguir = true)) do
    begin
      resto:= datos[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?

Con qué tipo de problemas del curso de nivelación se compara este problema?