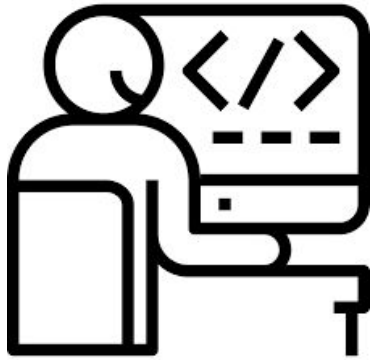


Algoritmos y Programación I

AyPI – **Temas de la clase pasada**

- Tipo de Datos Estructurado
- Clasificación de tipos de Datos Estructurados
- Tipo de datos REGISTRO
- Corte de control

AyPI – **Temas de la clase de hoy**

- Tipo de Dato Arreglo

AyPI – 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 sé que la edad máxima es 98 ¿cómo informo cuántas veces apareció?

AyPI – 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 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

AyPI – 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 la edad máxima, 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

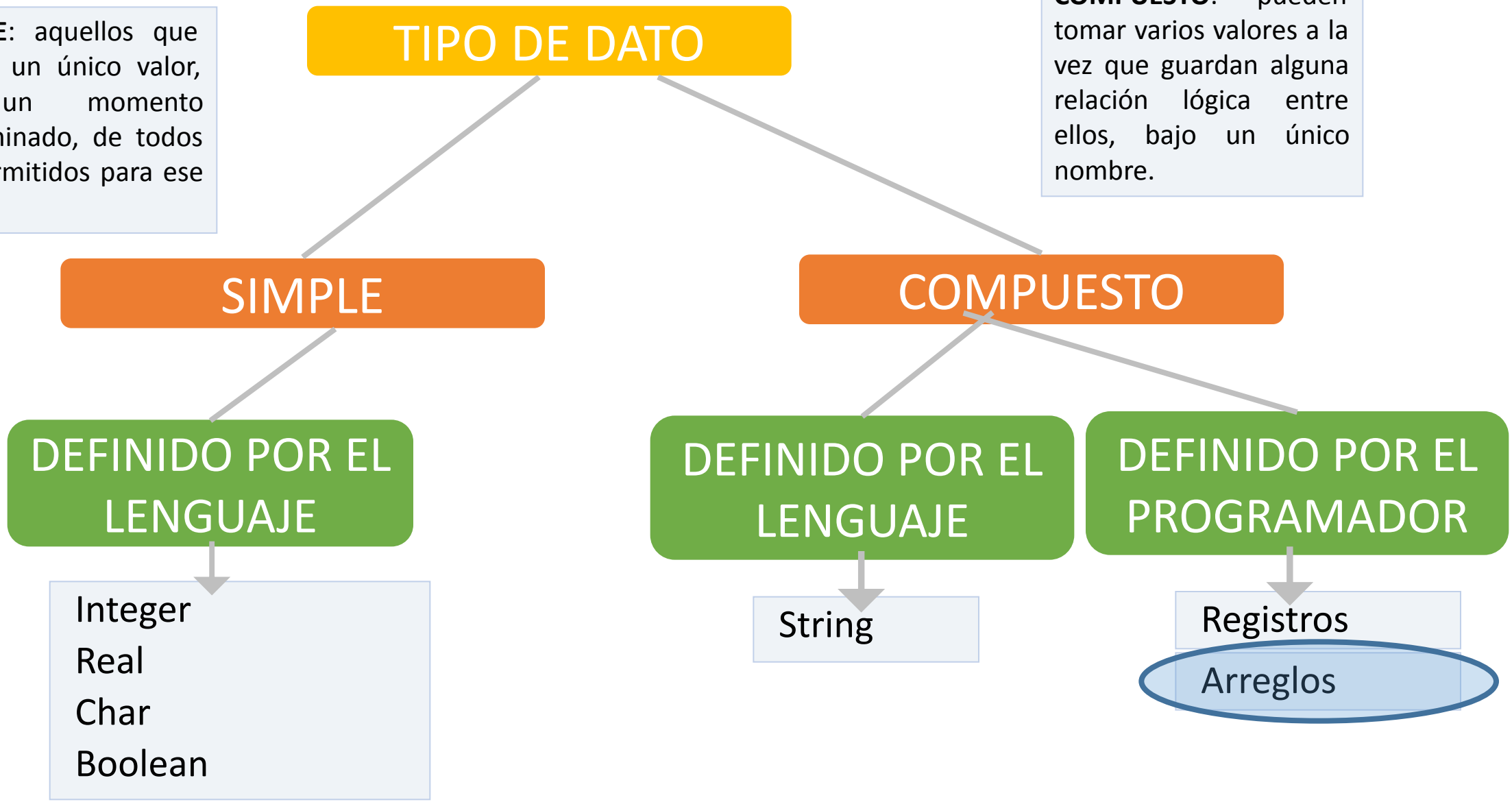
Recorrer la estructura y comparar con el máximo

20	98	68	2	98	23	4

AyPI – **TIPOS DE DATOS ESTRUCTURADOS** Arreglos

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

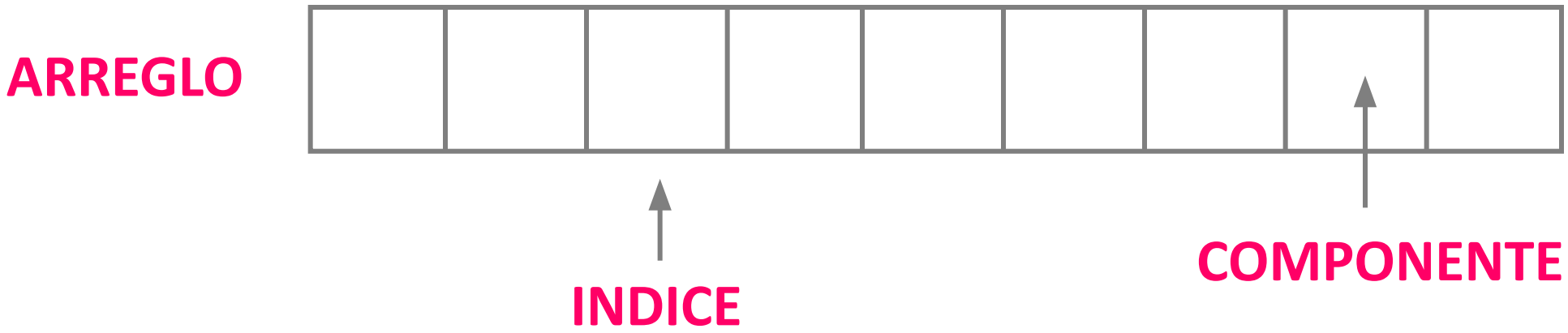
COMPUESTO: pueden tomar varios valores a la vez que guardan alguna relación lógica entre ellos, bajo un único nombre.



AyPI – 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.



AyPI – **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.

HOMOGÉNEA

Los elementos pueden son del mismo tipo .

ESTÁTICA

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.

AyPI – **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.

CARACTÉRISTICAS

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?

AyPI – TIPOS DE DATOS ESTRUCTURADOS Arreglos

VECTOR

```
Program uno;

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

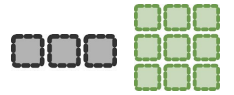
Nombre del tipo

El rango debe ser de tipo ordinal.
Integer
Char
Boolean

El tipo debe ser estático
Integer
Real
Char
Boolean
Registro
Vector

AyPI – 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

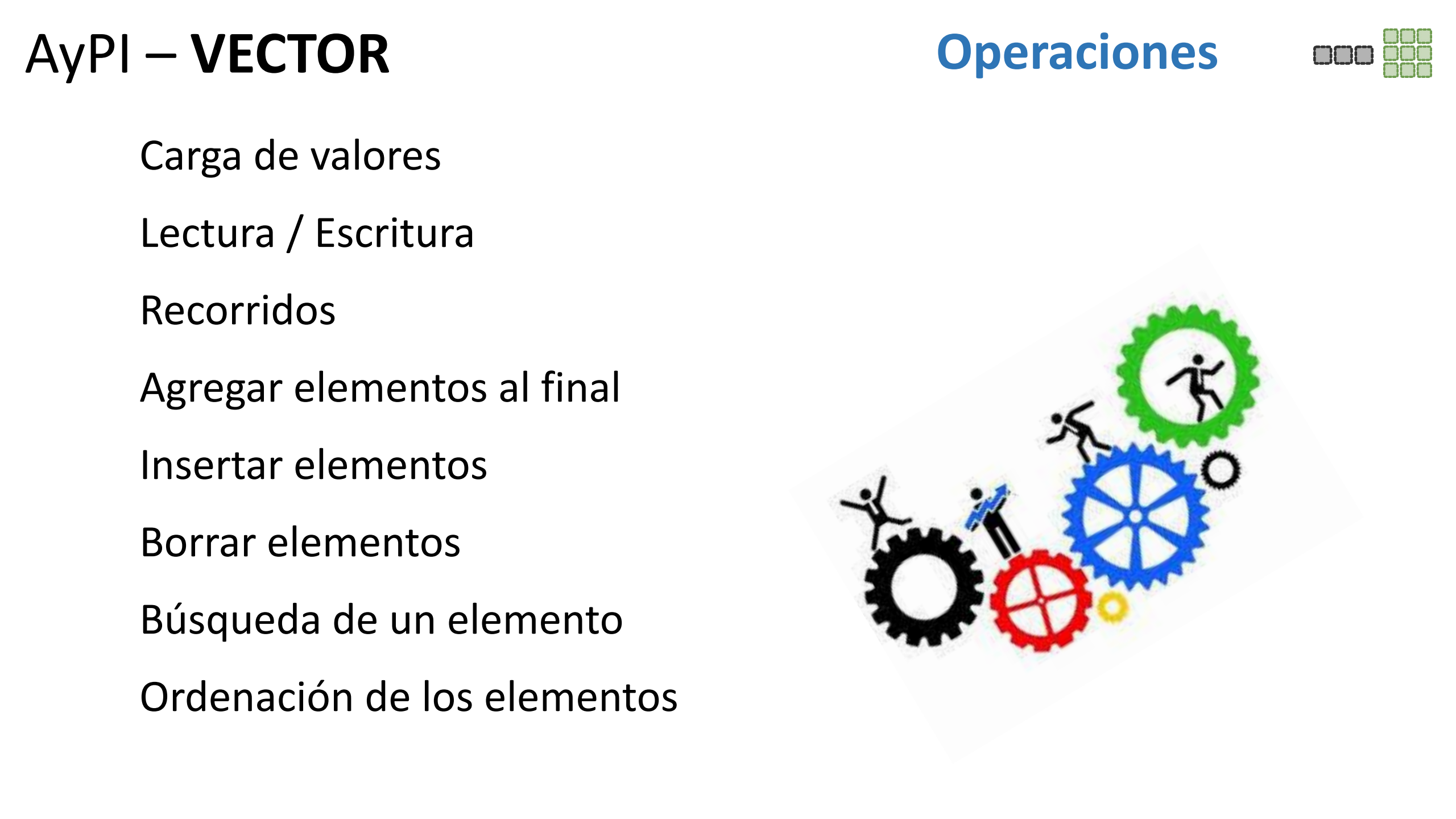
```
 otro: otros;
```

otro reserva memoria para 6 números enteros

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

AyPI – **VECTOR**

Operaciones



Carga de valores

Lectura / Escritura

Recorridos

Agregar elementos al final

Insertar elementos

Borrar elementos

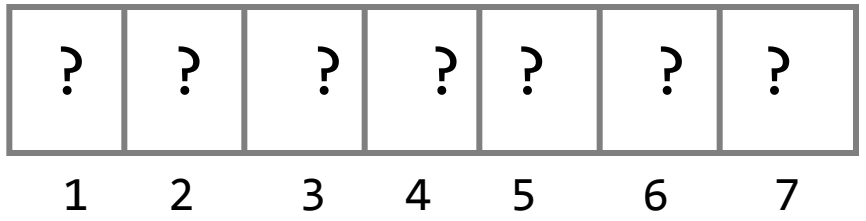
Búsqueda de un elemento

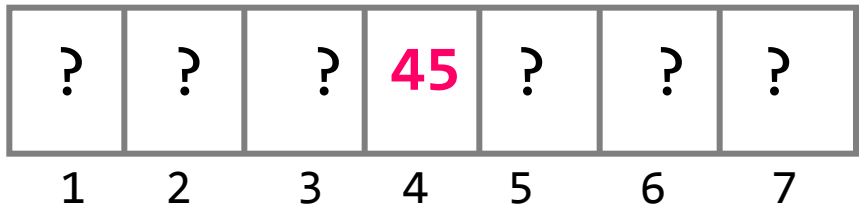
Ordenación de los elementos

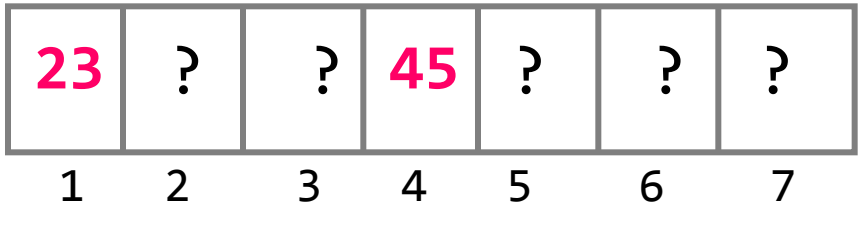
AyPI – **VECTOR**

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?

AyPI – **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)

?	?	?	?	?	?	?
1	2	3	4	5	6	7

v

3	-5	123	45	8	145	10
1	2	3	4	5	6	7

v

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

End.
```

¿Cómo se modulariza?

AyPI – **VECTOR**

Operaciones – Carga de valores

¿Puede ser una función?

¿Se conoce tam?

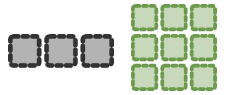
```
Procedure cargar (var datos: vector);
Var
   i,valor:integer;

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

OPCIÓN 2

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

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

AyPI – **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?

AyPI – **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;
```

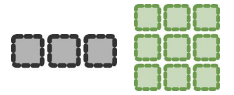
¿Puede ser una función? **NO**

OPCIÓN 2

```
Procedure imprimir (datos: vector);

 Var
  i:integer;

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

AyPI – **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.
```

¿Cómo solucionamos el problema planteado inicialmente?

AyPI – **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

AyPI – **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…

```
Program 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.
```

AyPI – **VECTOR**

EJERCICIO

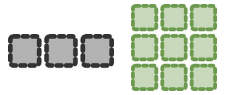
```
Procedure cargar (var datos:vector);
Var
  i,valor:integer;

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

OPCIÓN 2

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

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

AyPI – **VECTOR**

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;
  maximo:= max;
End;
```

¿Es necesario usar max? SÍ

AyPI – **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 necesario usar cant? **SÍ**

¿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.

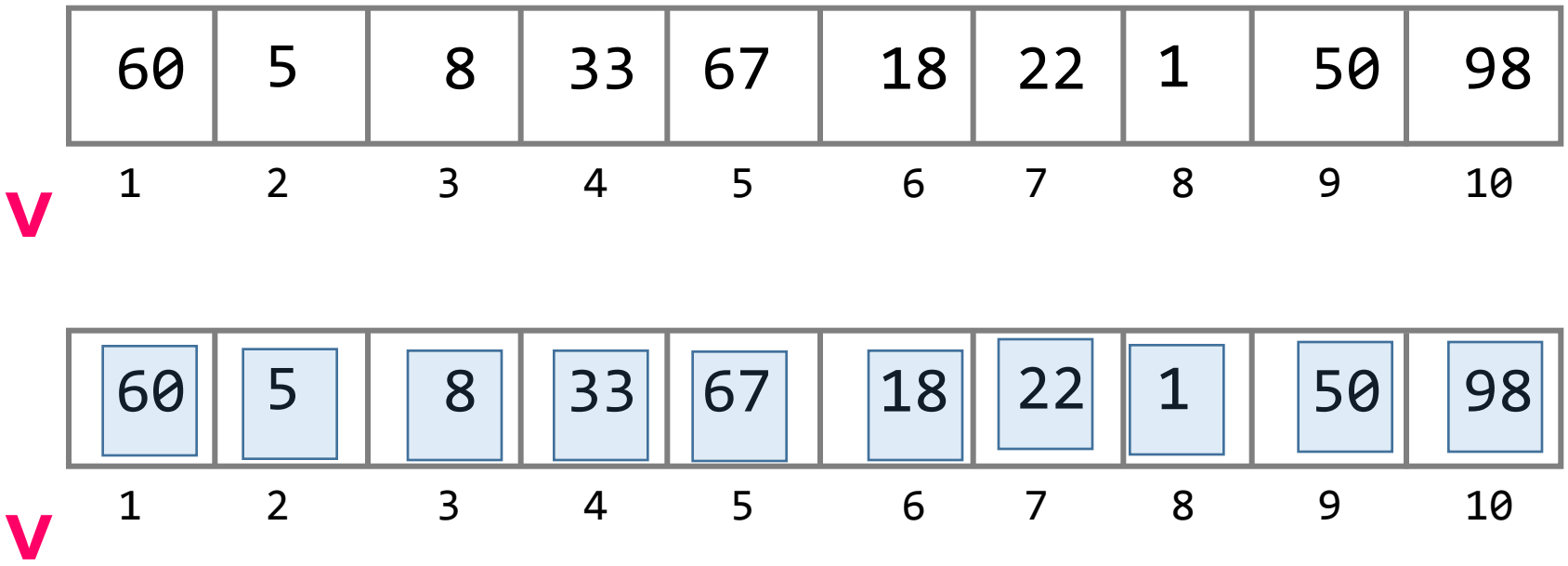
RECORRIDO - PARCIAL

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

AyPI – **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 números leídos son positivos.



Cant 0

Cant 1

Cant 2

Cant 3

¿Cómo lo implemento?

AyPI – **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.
```

AyPI – **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;
```

OPCIÓN 2

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

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

AyPI – **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;
```

OPCIÓN 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;
```

AyPI – **VECTOR**

RECORRIDO - PARCIAL

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

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

V

POS 4

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

V

AyPI – **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.
```

AyPI – **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?

AyPI – **VECTOR**

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?

AyPI – **VECTOR**

VECTOR DE REGISTROS

Se leen datos de 100 inmuebles. Realizar un programa que almacene los datos en un vector, y luego procese dicha información para informar, para cada letra del alfabeto, la cantidad de inmuebles cuya Localidad comienza con dicha letra.

Localidad: “La Plata” ...	Localidad: “La Plata” ...	Localidad: “Gonnet” ...	Localidad: “Villa Elisa” ...	Localidad: “La Plata” ...	...	Localidad: “La Plata” ...	Localidad: “Los Hornos” ...
1	2	3	4	5	....	99	100

V

A	B	C	…	G	…	L	…	V	…	Z
0	0	0		1		4		1		0

AyPI – **VECTOR**

VECTOR DE REGISTROS

Se leen datos de 100 inmuebles. Realizar un programa que almacene los datos en un vector, y luego procese dicha información para informar, para cada letra del alfabeto, la cantidad de inmuebles cuya Localidad comienza con dicha letra.

```
Program uno;
const dimF = 100;
type
  fecha = record
    dia: integer;
    mes:integer;
    año:integer;
  end;
  inmueble = record
       tipo: string;
       cantHab: integer;
       cantBaños:integer;
       precio: real;
       localidad: string;
       fechaPub:fecha;
    end;
    vInmuebles = array[1..dimF] of inmueble;
    vLetras = array['A'..'Z'] of integer;
```

```
function leerInmueble (): inmueble;
  ...
procedure cargarInmuebles (var vI: vInmuebles);
  ...
procedure inicializar (var vL: vLetras);
  ...
procedure procesarDatos(vI: vInmuebles;
                        vL: vLetras);
  ...


var
    datos: vInmuebles;
    letras : vLetras;
begin
  cargarInmuebles (datos);
  inicializar(letras);
  procesarDatos(datos,letras);
end.
```

AyPI – **VECTOR**

VECTOR DE REGISTROS

Se leen datos de 100 inmuebles. Realizar un programa que almacene los datos en un vector, y luego procese dicha información para informar, para cada letra del alfabeto, la cantidad de inmuebles cuya Localidad comienza con dicha letra.

```
procedure cargarInmuebles (var vI: vInmuebles);
var i: integer;
begin
  for i:= 1 to dimF do
     vI[i]:= leerInmueble();
end;
```

```
procedure inicializar (var vL: vLetras);
var c: char;
begin
  for c:= 'A' to 'Z' do
    vL[c]:= 0;
end;
```

```
procedure procesarDatos(vI: vInmuebles;
                        vL: vLetras);

var i: integer;
    c, letra: char;

begin
  for i:= 1 to dimF do
  begin
     letra:= vI[i].localidad[1];
     vL[letra]:= vL[letra] + 1;
  end;
  for c:= 'A' to 'Z' do
      writeln(c, ' ' , vL[c]);
end;
```