



Conceptos de Algoritmos Datos y Programas

CADP – TEMAS



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

CADP – TIPOS DE DATOS

ESTRUCTURADOS



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

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

TIPO DE DATO

SIMPLE

COMPUESTO

DEFINIDO POR EL LENGUAJE

DEFINIDO POR EL PROGRAMADOR

DEFINIDO POR EL LENGUAJE

DEFINIDO POR EL PROGRAMADOR

Integer
Real
Char
Boolean

Subrango

String

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 se que la edad máxima es **77** cómo informo cuantas veces apareció?

Con lo que sabemos hasta ahora tenemos dos alternativas

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.
Cuanto 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.
Cuanto mas valores se lean
el problema es mas grande

SOLUCION?

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
----	----	----	---	----	----	---	----	----	---

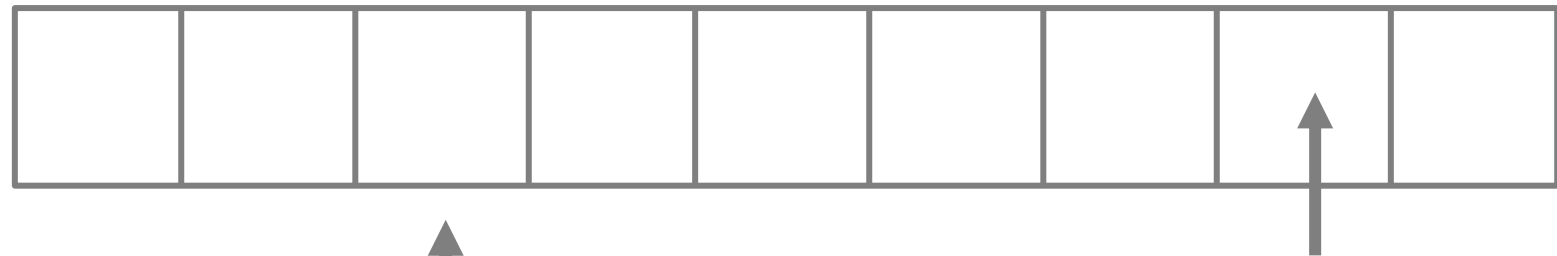
2



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.

ARREGLO



INDICE

COMPONENTE

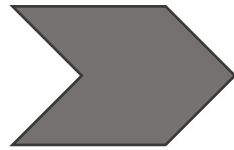


VECTOR (arreglo de una dimensión)

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

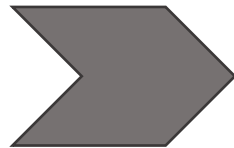
Cómo se declara?

HOMOGENEA



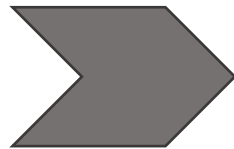
Los elementos pueden ser del mismo tipo .

ESTATICA



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

INDEXADA



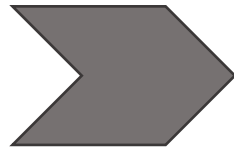
Para acceder a cada elemento de la estructura se debe utilizar una variable '**índice**' que es de tipo ordinal.



VECTOR (arreglo de una dimensión)

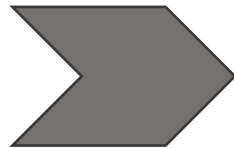
Es una colección de elementos que se guardan consecutivamente en la memoria y se pueden referenciar a través de un índice.

HOMOGENEA



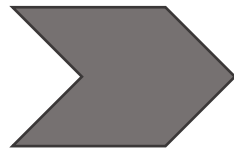
Los elementos pueden ser del mismo tipo .

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.



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

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

Carga de valores

Lectura / Escritura

Recorridos

Agregar elementos al final

Insertar elementos

Borrar elementos

Búsqueda de un elemento

Ordenación de los elementos





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**



```
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;

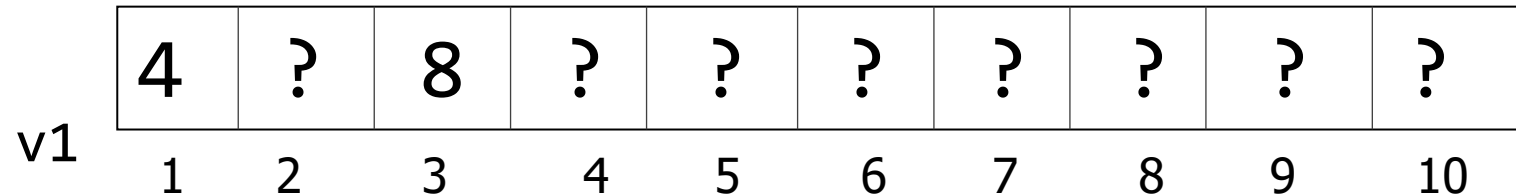
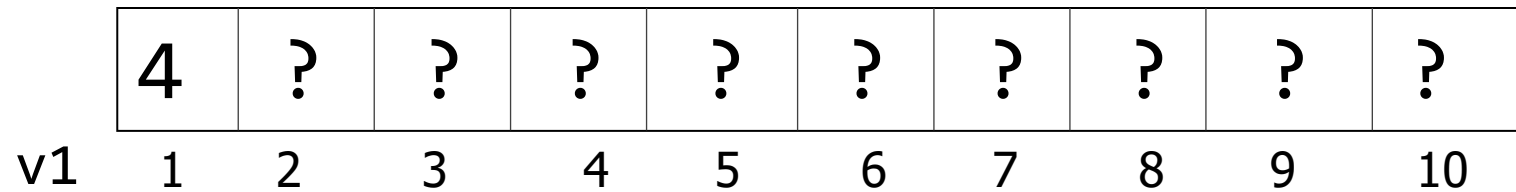
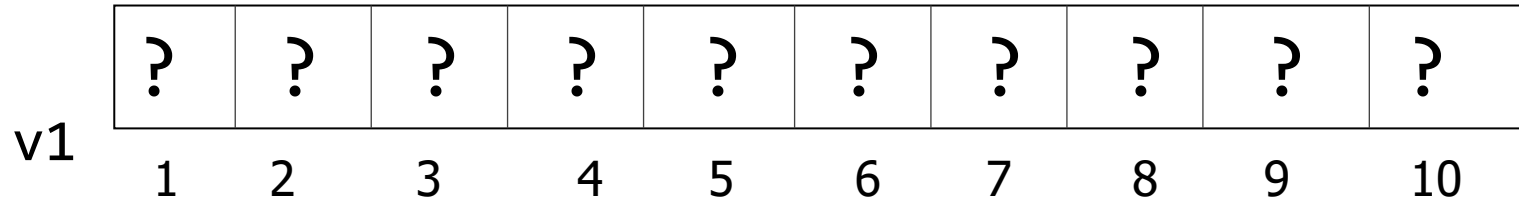
Begin

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;

v1

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



No se puede hacer
read(v1)

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

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
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

ALTERNATIVA

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

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

Puede ser una función? Se puede utilizar tam?

Cómo muestro los datos?

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;
```

v

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

ALTERNATIVA

```
Procedure imprimir (v: vector);
```

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

Puede ser
una función?

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

CADP – TIPO DE DATOS VECTOR -



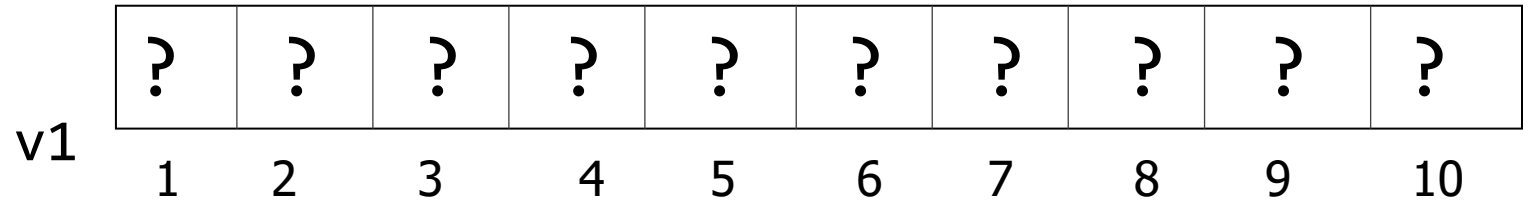
Program uno;

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;
```

v

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

```
Begin
```

```
  max:= -9990;
```

```
  for i:= 1 to tam do
```

```
    begin
```

```
      valor:= v[i];
```

```
      if (valor >= max) then
```

```
        max:= v[i];
```

```
      end;
```

```
  maximo:= max;
```

```
End;
```

```
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;
```

ALTERNATIVA

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

```
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;
```

ALTERNATIVA