VECTORES

Explicación Práctica

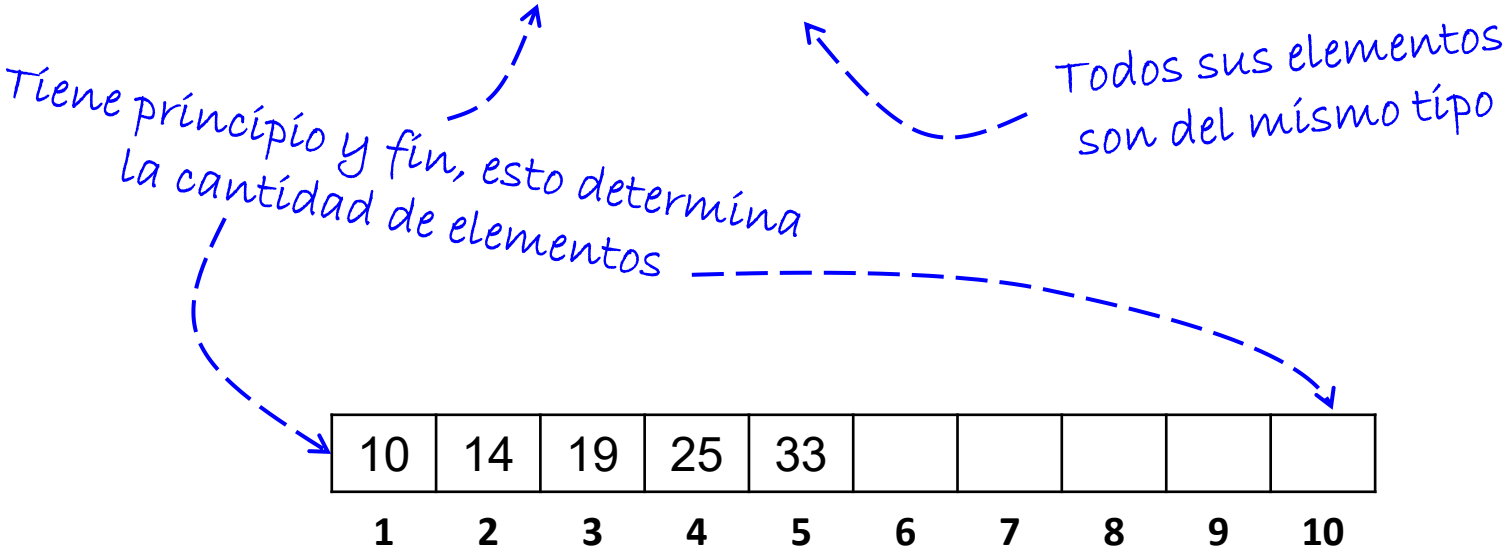
Programación I - 2021

Facultad de Informática y Facultad de Ingeniería - UNLP

Arreglos

Aspectos básicos

Arreglo : colección finita y homogénea de elementos



Arreglos
VECTOR

Declaración de tipo

Se pueden declarar vectores de:

integer, real, char, boolean, subrango, string, registro, vectores.

Declaración de tipo **vector**:

```
nombreTipo = Array [ rango ] of tipoElem;
```

Rango no nulo de un tipo ordinal.
Ejemplo: 'a'.. 'z'
Ejemplo: 0..9

Un tipo permitido previamente definido

Arreglos
Aspectos básicos

En la práctica usamos:

Dimensión física: la asignada al vector en su declaración. Establece la capacidad máxima de elementos

Dimensión lógica: posición del último elemento que se cargó en el vector. *Debe controlar el programador a medida que agrega/quita elementos.*

Ejemplo:

```
Type
    vector = Array [1..10] of integer;
Var
    v: vector;
    dl: integer;
```

Dimensión física = 10
Dimensión lógica = 5

10	14	19	25	33					
1	2	3	4	5	6	7	8	9	10

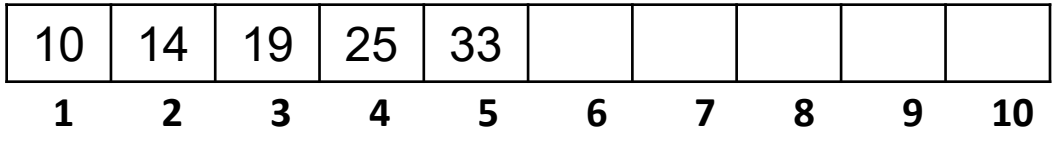
Arreglos

Aspectos básicos

Acceso directo: acceso a un elemento mediante la especificación de la posición del donde se encuentra en el vector.

Ejemplo:

```
Type
  vector = Array [1..10] of integer;
Var
  v: vector;
  dl: integer;
```



Acceso al elemento en posición 5: v[5]

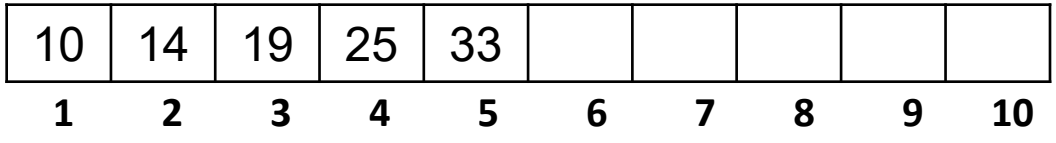
Arreglos

Aspectos básicos

Acceso directo: acceso a un elemento mediante la especificación de la posición del donde se encuentra en el vector.

Ejemplo:

```
Type
  vector = Array [1..10] of integer;
Var
  v: vector;
  dl: integer;
```



Acceso al elemento en posición 5: v[5]

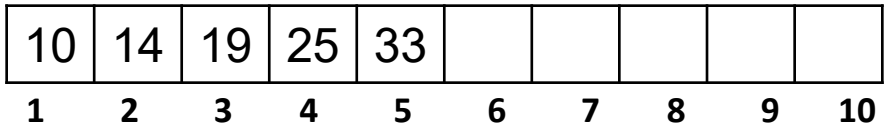
¿Qué operaciones puedo hacer con v[5]?

Arreglos
Aspectos básicos

Acceso directo: acceso a un elemento mediante la especificación de la posición del donde se encuentra en el vector.

Ejemplo:

```
Type
  vector = Array [1..10] of integer;
Var
  v: vector;
  dl: integer;
```



¿Cómo imprimir todo el arreglo?

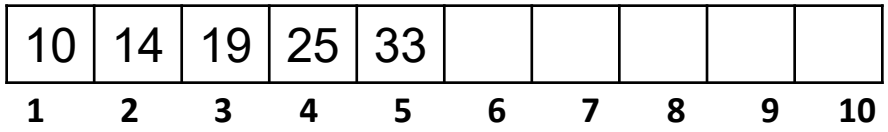
Arreglos

Aspectos básicos

Acceso directo: acceso a un elemento mediante la especificación de la posición del donde se encuentra en el vector.

Ejemplo:

```
Type
  vector = Array [1..10] of integer;
Var
  v: vector;
  dl: integer;
```



¿Cómo imprimir todo el arreglo?

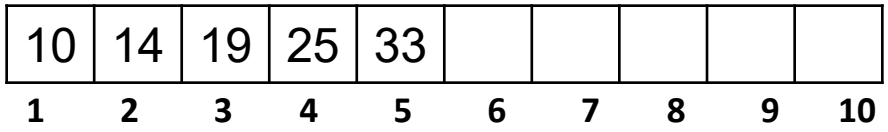
```
...
For i:= 1 to dl do
  write (v[i]);
...
```

Arreglos
Aspectos básicos

Acceso directo: acceso a un elemento mediante la especificación de la posición del donde se encuentra en el vector.

Ejemplo:

```
Type
  vector = Array [1..10] of integer;
Var
  v: vector;
  dl: integer;
```



¿Cómo imprimir todo el arreglo?

```
...
For i:= 1 to dl do
  write (v[i]);
...
```

Pensar en módulo...
¿qué parámetros debería pasar?

VECTOR

Ejercicio 1

Realizar un programa que cargue un arreglo de 1500 números enteros positivos.

Finalizada la carga informar:

A) Los números que son múltiplos de 2

B) Los números que incluyen todos los dígitos impares

¿Dónde almaceno los números?

10	3	50	8	21	3	33	29	8	...
1	2	3	4	5	6	7	8	9	.....

¿Necesito llevar la dimensión lógica?

A y B en un único recorrido LUEGO de la carga. ¿Ideas?

```
Program Digitos;
type
    digitos=0..9;
    conjunto=set of  digitos;
    rango=1..1500;
    numeros = array [rango] of integer;
   { Acá se declaran los módulos}
var
  v: numeros; i: rango;
Begin
  cargar (v);
  for i:= 1 to 1500 do begin
    if EsPar ( v[i] ) then
      write( v[i], 'es par' );
    if AparecenImp(v[i]) then
      write (v[i] ,'tiene todos los dig imp');
  end;
end.
```

```
procedure cargar (var a: numeros );
var i: rango;
begin
  for i:= 1 to 1500 do read(a[i]);
End;
```

```
Function EsPar ( num: integer): boolean;
begin
  EsPar:= ((num mod 2) = 0);
End;
```

```
Function AparecenImp(num:integer):boolean;
var
  resto: digitos; c: conjunto;
Begin
  c:= [1,3,5,7,9];
  while (num <> 0) do begin
    resto:= num mod 10; // Obtengo digito
    c:= c - [resto] ; // Lo saco del conjunto
    num:= num div 10; // Achico número
  end;
  AparecenImp:= (c = []);
End;
```

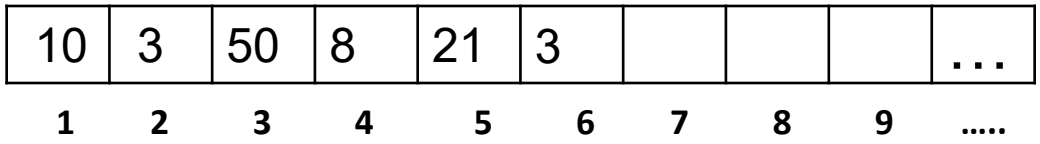
VECTOR
Ejercicio 2

Realice las modificaciones necesarias al ejercicio 1 para el caso que la carga de números termine cuando se lee el número 999.

Tener en cuenta que en el arreglo se pueden cargar como máximo 1500 números.

Necesito llevar la dimensión lógica

Se leen 10 3 50 8 21 3 999



¿Dimensión física ? 1500

¿Dimensión lógica? 6

Cargar: ¿Cuando finaliza la carga del arreglo?
¿Estructura de control?

Programa Ppal: Modificaciones al recorrido

```
Program Digitos;
type
    rango=1..1500;
    digitos=0..9;
    conjunto=set of  digitos;
    numeros = array[rango] of integer;
   { Acá se declaran los módulos}
var
    v: numeros; i: rango;
    DIMLOG:integer;
Begin
 cargar (v, DIMLOG);
 for i:= 1 to  DIMLOG do begin
     if EsPar ( v[i] ) then
        write(v[i], 'es par' );
     if AparecenImp(v[i]) then
        write (v[i] ,'tiene todos los dig imp');
  end;
end.
```

```
procedure cargar (var a: numeros;
                  var dimlog: integer);
var num: integer
Begin
  dimlog:=0;
  read(num);
  while (num <> 999) and (dimlog < 1500) do
  begin
    dimlog:= dimlog + 1;
    a[dimlog] := num;
    read(num);
  end;
End;
```

Ejercicio 3

Hacer un programa que lea una secuencia de números enteros terminada en 0. Informar la cantidad de veces que aparece cada dígito del 0 al 9 entre todos los números leídos.

Ejemplo: se leen los números: 457 9875 5 24879 0

{Se debe informar por cada digito cuantas veces aparecieron cada uno de ellos}

***0** aparece 0 veces;*
***1** aparece 0 veces;*
***2** aparece 1 veces;*
***3** aparece 0 veces;*
***4** aparece 2 veces;*
***5** aparece 3 veces;*
***6** aparece 0 veces;*
***7** aparece 3 veces;*
***8** aparece 2 veces;*
***9** aparece 2 veces;*

¿Debo almacenar los números leídos?

¿Cómo cuento las veces que aparece cada dígito?

0	0	1	0	2	3	0	3	2	2
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

¿Necesito llevar la dimensión lógica?

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  {Acá se declaran los módulos}
var
  losnros: numeros;
  num: integer;
begin
  inicializar (losnros);
  read (num);
  while (num <> 0) do begin
    descomponer (losnros, num);
    read (num);
  end;
  informo(losnros);
end.
```

```
procedure inicializar (var a: numeros );
var i: rango;
begin
  for i:= 0 to 9 do
    a[i]:= 0;
End;
```

```
procedure descomponer(var a:numeros; num:integer);
var
  resto: rango;
begin
  while (num <> 0) do begin
    resto:= num mod 10; {Obtengo digito}
    {Incremento contador asociado al digito}
    a[resto]:= a[resto] + 1;
    num:= num div 10; {Achico número}
  end;
end;
```

```
procedure informo ( a: numeros );
Var
  i: rango;
Begin
  for i:= 0 to 9 do
    writeln(i, ‘ = ’,a[i]);
End;
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada digito.

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  {Acá se declaran los módulos}
var
  losnros: numeros;
  num: integer;
begin
  inicializar (losnros);
  read (num);
  while (num <> 0) do begin
    descomponer (losnros, num);
    read (num);
  end;
  informo(losnros);
end.
```

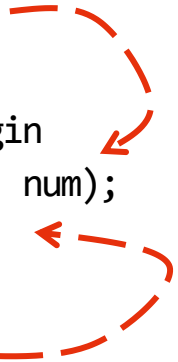
```
procedure inicializar (var a: numeros );
var i: rango;
begin
  for i:= 0 to 9 do
    a[i]:= 0;
End;
```

```
procedure descomponer(var a:numeros; num:integer);
var
  resto: rango;
begin
  while (num <> 0) do begin
    resto:= num mod 10; {Obtengo digito}
    {Incremento contador asociado al digito}
    a[resto]:= a[resto] + 1;
    num:= num div 10; {Achico número}
  end;
end;
```

```
procedure informo ( a: numeros );
Var
  i: rango;
Begin
  for i:= 0 to 9 do
    writeln(i, ' = ',a[i]);
End;
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada digito.

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  {Acá se declaran los módulos}
var
  losnros: numeros;
  num: integer;
begin
  inicializar (losnros);
  read (num);
  while (num <> 0) do begin
    descomponer (losnros, num);
    read (num);
  end;
  informo(losnros);
end.
```



```
procedure inicializar (var a: numeros );
var i: rango;
begin
  for i:= 0 to 9 do
    a[i]:= 0;
End;
```

```
procedure descomponer(var a:numeros; num:integer);
var
  resto: rango;
begin
  while (num <> 0) do begin
    resto:= num mod 10; {Obtengo digito}
    {Incremento contador asociado al digito}
    a[resto]:= a[resto] + 1;
    num:= num div 10; {Achico número}
  end;
end;
```

```
procedure informo ( a: numeros );
Var
  i: rango;
Begin
  for i:= 0 to 9 do
    writeln(i, ‘ = ’,a[i]);
End;
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada digito.

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  {Acá se declaran los módulos}
var
  losnros: numeros;
  num: integer;
begin
  while (num <> 0) do begin
    inicializar (losnros);
    descomponer (losnros, num);
    informo (losnros);
    read (num);
  end;
end.
```

1. Modifique para informar para cada número la cantidad de veces que aparece cada digito.

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  {Acá se declaran los módulos}
var
  losnros: numeros;
  num: integer;
begin
  while (num <> 0) do begin
    inicializar (losnros);
    descomponer (losnros, num);
    informo (losnros);
    read (num);
  end;
end.
```

2. Modifique para informar el dígito que más veces apareció para cada número

```
Program Digitos;
type
  rango=0..9;
  numeros = array [rango] of integer;
  { Acá se declaran los módulos }
var
  losnros : numeros;
  num: integer;
begin
  read (num);
  while (num <> 0) do begin
    inicializar (losnros);
    descomponer (losnros, num);
    write('el dig que mas aparece es', DigitoMaximo(losnros));
    read (num);
  end;
end.
```

```
Function DigitoMaximo(a:numeros): rango;
Var
     i, digmax: rango; max: integer;
Begin
     max:= -1;
     for i:= 0 to 9 do
       if (a[i] > max ) then begin
         max:= a[i];
         digmax:= i;
       end;
     DigitoMaximo:= digmax;
 End;
```

2. Modifique para informar el dígito que más veces apareció para cada número