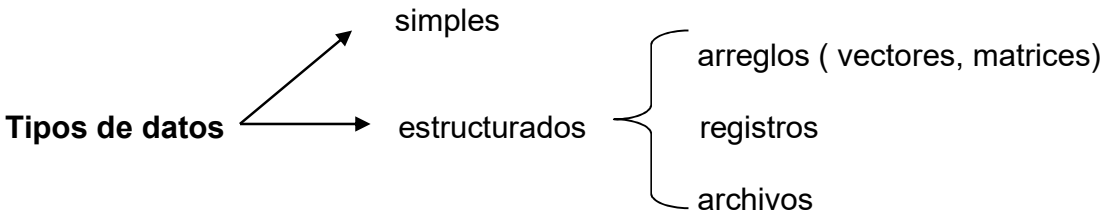
Capitulo 6: Arreglos unidimensionales, lineales o vectores

1. Declaración del tipo vector, acceso.
2. Lectura, escritura, operaciones
3. Generación de un vector a partir de un proceso.
4. Vectores paralelos.
5. Búsqueda de un valor en un vector (desordenado/ordenado)
6. Ordenación
7. Inserción, eliminación, intercalación.
8. Índice con significado (índices aleatorios)



Los tipos de datos simples permiten declarar variables que almacenan un solo valor (entero, carácter, boolean, etc).
Los tipos de datos estructurados permiten almacenar un conjunto de valores y poder acceder en forma individual o grupal (en algunas operaciones) a la totalidad de los mismos. Estos tipos deben ser declarados en la sección Type del programa.

1. Declaración del tipo vector, acceso.

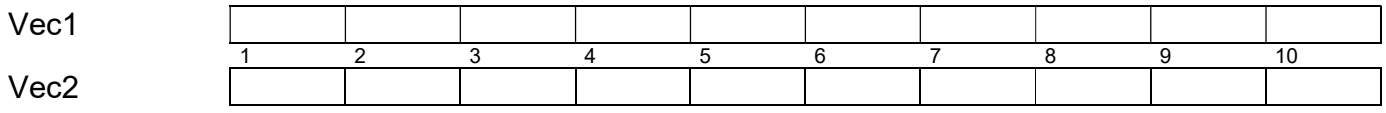
Agrupan un conjunto de valores del mismo tipo, en una única variable. Cuando se describe el tipo se establece:
a. La cantidad máxima de elementos y el rango sus posiciones (para el acceso individual).
b. El tipo de los valores que almacena

```
Type
      TipoVector =  array [1..10]   of    integer;
```

Identificador de tipo — (a) tipo índice — (b) tipo base

El tipo índice debe ser ordinal, el tipo base no tiene restricciones (simple o estructurado)
La declaración del TipoVector asocia este identificador a una estructura de 10 elementos de tipo entero (base). La cantidad de elementos se establece mediante un valor constante (10).
Cada uno de ellos se puede acceder mediante su ubicación : 1..10 (índice)
A partir de este tipo se declaran una o más variables del tipo TipoVector

```
Var
Vec1, Vec2 : TipoVector;
j,k : byte;
```



Este espacio de memoria queda reservado, independientemente de que el programa requiera parte o la totalidad de las componentes de las variables Vec1 y Vec2.
Para acceder a una determinada componente de un vector, se indica la posición entre corchetes.

```
Vec1[2] := 5;
Vec1[1] := Vec1[2] –3;
j:=3;
Vec1[j] := Vec1[j-1] *2;
Vec2[1] := Vec1[3];
Vec1[j + 1] := Vec1[1] + Vec2[j - 2];
For j:=2 to 10 do
   Vec2[j] := 0;
For  k:= 1 to 4 do
   Write(Vec1[k] : 5);
```

Vec1	2	5	10	12						
	1	2	3	4	5	6	7	8	9	10
Vec2	10	0	0	0	0	0	0	0	0	0

El índice puede ser constante, variable o expresión. debe tomar valores dentro del rango con el que se describió el tipo,

Como se observa en las líneas de código que operan sobre los vectores Vec1 y Vec2 es posible acceder a un elemento en particular o (utilizando un ciclo) a un conjunto de elementos.
La compatibilidad que debe tenerse cuenta, tanto en las operaciones como en las asignaciones, esta condicionada al tipo Base.

Tanto la lectura desde teclado como la escritura de un arreglo debe hacerse componente a componente. Es posible asignar una variable de tipo arreglo a otra del mismo tipo (Vec1 := Vec2) , pero no admite otro tipo de operaciones como Vec1 + Vec2. o Vec1 * 5.

Otros ejemplos de tipo arreglo son

```
Const
   MaxElem = 5;

Type
     St10=string[10];
     TVecCh = array [1..MaxElem] of  St10;
     TVecReal = array [´A´..´Z´] of real;
```

Utilidad. Los arreglos permiten mantener en memoria un conjunto de datos y acceder a ellos (en cualquier lugar del programa sin necesidad de volver a leerlos o generarlos. Por ejemplo si se quiere determinar de un conjunto de personas cuantas tienen edad por encima del promedio. Se debe ingresar las edades almacenándolas en un arreglo, sumarlas, dividir la suma por la cantidad de personas y luego comparar cada edad (están en el arreglo) con el promedio.
Ejemplo (25,55,75,45) → Prom=50 → 55,75

2. Lectura, escritura, operaciones

Ejemplo1- A continuación se desarrolla un programa que utilizando funciones y procedimientos para cada ítem:

a. Lee un arreglo de enteros
b. Calcula la suma de sus elementos
c. Cuenta la cantidad de componentes pares
d. Imprime las componentes que se encuentran en ubicaciones pares
e. Imprime el mínimo
f. Permite elegir por medio de un menú el/los proceso/s descriptos

```
Program Ej1;
TYPE
      TV=array[1..100]  of  integer;
VAR
       N : byte;
       A :TV;
       Op : char;

Procedure LeeVector ( Var A : TV; Var N : byte);
Var
  i : byte;
Begin
Write('Ingrese la cantidad  de elementos del vector<=100');
Readln(N);
For i:=1 to N do
     Begin
     Write('Ingrese  el elemento ',i);  Readln(A[i])
     End
End;

Procedure EscVector ( V : TV;  L : byte);
Var   i : byte;
Begin
    For i := 1 to L do
         Write(V[i] : 5)
End;


Function Suma (A : TV; N: byte) : integer;
Var
    Sum : integer;
    i: byte;
Begin
    Sum:=0;
     for i:=1 to N  do
         Sum:=Sum+ A[i];
    Suma := Sum;
End;
```

```
{lee de archivo, son los mismos parametros}
Var
   Arch: text;
Begin
   Assign(Arch, 'Numeros.txt');
   Reset(Arch); N:=0;
   While Not EOF(Arch) do
      Begin
      N:= N + 1;
      Read(Arch, A[N]);
      End;
   Close(Arch)
End;
```

```
Function CuentaPares (A : TV; N: byte) : byte;
Var
     i, Cont : byte;
Begin
    Cont:=0;
     for i:=1 to N  do
        If  Not Odd (A[i] ) then
            Cont:=Cont+ 1;
    CuentaPares := Cont;
End;

Procedure EscPosPares (  A : TV;  N : byte);
Var
  i : byte;
Begin
    i:= 2;
    While  i<= N do
         Begin
         Write(A[i] : 5);
         i:=i +2;
         End
End;

Function Minimo (A : TV; N: byte) : integer;
Var
    Min : integer;
    i:byte;
Begin
    Min:=A[1];
     for i:=2 to N  do
        If  A[i] < Min then
            Min:=A[i];
    Minimo := Min;
End;

Procedure Menu ( Var Op : Char);
Begin
     Writeln(Menú de opciones');
     Writeln('1   -   Suma los elementos del arreglo');
     Writeln('2   -   Cuenta los elementos pares);
     Writeln('3   -   Imprime los elementos de las posiciones pares');
     Writeln('4   -   Calcula el mínimo');
      Writeln('5 - Fin');
      Repeat
          Write(' Ingrese su opción');   Readln(Op);
      Until  ( '1'< = Op) and ( Op < = '5');
End;
```

```
Begin {programa principal}
   LeeVec(A, N);
   EscVector(A, N);
   Repeat
      Menu(Op);
      Case Op of
        ´1´:  writeln('La suma de los elementos del arreglo es:', Suma(A, N));
        ´2´:  writeln('La  cantidad de elementos pares del arreglo es:', CuentaPares(A, N));
        ´3´:  begin
              writeln('Elementos de las posiciones pares del arreglo');   EscPosPares ( A , N )
             end;
        ´4´:  writeln('El mínimo del arreglo es:', Minimo(A, N))
      End ; {case}
 Until   Op = '5';
End.
```

Desarrollar programas Pascal que resuélva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

Importante :

- Cada proceso (lectura, escritura, cálculo) debe ser implementada mediante un procedimiento o función, según corresponda, correctamente parametrizada. No utilizar variables globales.
- Aunque no se explicite, informar siempre los resultados obtenidos.
- Antes de comenzar a hacer el algoritmo arme un ejemplo y resuelva manualmente.

1.-Ingresar K números que representan montos de ventas de una empresa, ordenados en forma ascendente.
Calcular y mostrar por pantalla el promedio de los mismos y el % de participación de cada uno respecto de la suma total en forma descendente.

2.-Ingresar M números naturales y luego
a. Contar e informar cuántos elementos son pares, impares y nulos.
b. Multiplicar todos los componentes de posición par por un número ingresado por teclado, validando que sea diferente de 0, mostrar por pantalla el conjunto de números resultante.
c. Mostrar por pantalla cuál es el lugar donde aparece el máximo (en caso de que este valor aparezca más de una vez, considerar el primero).
Ejemplo:
números : 3 4 8 1 24 3 1 24 15 24 → Máximo = 24 lugar= 5

3.Generación de un vector a partir de un proceso (que selecciona elemento de otro arreglo)

Ejemplo 2- Ingresar un conjunto de letras mayúsculas y minúsculas (* indica fin de datos). Generar un nuevo arreglo que contenga solo las consonantes. Mostrar el arreglo obtenido o indicar que no hubo consonantes.

```
Program Ej2;
TYPE
     TV=array[1..100]  of  char;
```

```
Procedure LeeVector ( Var V : TV; Var N : byte);
Var
   Car : char;
Begin
    N:=0;
    Write('Ingrese  una letra, * fin de datos ');  Readln(Car);
    While   Car <> '*'do
        Begin
         N:= N + 1;
         V[N]:= Car;
        Write('Ingrese  una letra, * fin de datos ');  Readln(Car);
        End
End;

Function EsVocal(L:char): boolean;
   ……
Procedure GeneraOtro (VL : TV;  N : byte; Var VC : TV;  Var M : byte);
Var
   i : byte;
Begin
    M:= 0;
    For i:=1 to N do
        If    Not EsVocal(VL[i])   Then
            Begin
            M:=M +1;   VC[M] := VL[i];
           End
End;

Procedure EscVector ( V : TV;  L : byte)
Var
    i : byte;
Begin
    For i:=1 to L do
        Write(V[i] : 5)
End;
VAR
       N, M: byte;
       VL,VC :TV;

Begin
LeeVector(VL, N);
GeneraOtro (VL ,  N, VC , M );
If M <> 0  then
  EscVector(VC, M)
Else
  Writeln('No se ingresaron consonantes');
End.
```

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

3.- Ingresar en un arreglo N números enteros, generar dos arreglos VPos y VNeg que contendrán los números positivos y negativos respectivamente. Mostrar el más numeroso, ambos si la cantidad de elementos coinciden

4.-Almacenar en una vector K números reales, se sabe que existen elementos nulos. Generar un nuevo arreglo donde cada uno de sus elementos sean la suma de los valores previos a un cero.
Ejemplos: K= 11, (2, 6, 0, 1, 0, 4, 7,-2,0, 8, 4) → (8, 1, 9)
(0, 2, 6, 0, 1, 0, 4, 7, -2, 3, 0) →(0, 8, 10)

4.Vectores paralelos

A partir de las declaraciones

```
Type
    St25 = string[25];
    TVNom = array[1..50] of St25;
    TVEdad = array[1..50] of byte;

Var
    VNom  : TVNom;
    VEdad : TVEdad;
```

Si quiero almacenar nombre y edad de un conjunto de personas, utilizo los vectores VNom y VEdad en forma "paralela".

Se dice que dos o más arreglos son paralelos, si tienen la misma cantidad de componentes y están relacionados de forma tal que sus componentes en la i-esima posición forman una unidad de información.

VNom[1] y VEdad[1] nombre y edad de la primera persona
VNom[2] y VEdad[2] nombre y edad de la segunda persona
…….
VNom[i] y VEdad[i] nombre y edad de la *i-esima* persona

Ejemplo3-En un archivo se han grabado, en cada línea, el nombre (10 caracteres) y la edad de un conjunto de personas. Se pide ingresar dichos datos en dos vectores para luego listar los nombres de las personas cuya edad este por debajo del promedio

```
Program Ej3;
Type
    St25 = string[25];
    TVNom = array[1..50] of St25;
    TVEdad = array[1..50] of byte;
```

```
Procedure LeeParalelo( Var VNom:TVNom; Var VEdad: TVEdad; Var N:Byte);
      Var
        Arch : text;
      Begin
         Assign(Arch, 'Personas.txt'); Reset (Arch );
         N:=0;
          while not eof(Arch) do
              Begin
              N:= N + 1;
              Readln (Arch,  VNom[ N], VEdad [ N ] );
              End;
        Close(Arch);
     End;

Function Promedio (V : TVEdad; N: byte) : word;
Var
    Sum : word;
    i: byte;
Begin
    Sum:=0;
     for i:=1 to N  do
         Sum:=Sum+ V[i];
    Suma := Sum DIV N;
End;

Procedure Listado( VNom: TVNom; VEdad: TVEdad; N: Byte; Prom: word);
      Var
         i: Byte
      Begin
         For i:= 1 to N do
            If VEdad[ i ] > Prom Then
               Writeln ( VNom[ i ] );
      End;


Var
      VNom : TVNom;     VEdad : TVEdad;
       Prom,  N : byte;

Begin
LeeParalelo(  Vnom,  Vedad, N);
Prom:= Promedio(Vedad, N);
Listado( Vnom, Vedad, N, Prom);
End.
```

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

5.- Leer de un archivo de texto sobre tres arreglos los datos de un conjunto de autos, por cada uno:
- Patente
- Año
- Precio

Se pide mediante un menú que permita la repetición de las opciones con diferentes valores, calcular:
a. Para un año dado, precio mínimo (puede no existir)
b. Para un precio dado cantidad de vehículos por debajo de dicho valor
c. Para un rango de años dado [Año1, Año2] precio promedio de los autos en dicho rango (puede no existir)

6.- Ingresar los elementos de un vector A y a continuación ingresar los de un vector B, si la cantidad de componentes ingresadas para cada uno de los vectores no es la misma indicar con un cartel y finalizar, de lo contrario calcular:
a. La suma y la diferencia de los vectores.
b. El producto escalar.
La fórmula del producto escalar es la que sigue:

$$Producto = \sum_{i=1}^{N} (A_i * B_i)$$ (N indica la cantidad de componentes de A y B)

5.Búsqueda de un valor en un vector (desordenado/ordenado)

Siendo V un arreglo de N elementos enteros.
Responda las preguntas y describa el resultado que devuelve la función si se invoca con los parámetros actuales dados en los incisos a y b

¿Que hace la siguiente función?

```
Function Busca ( V:TV; N:byte; x:integer ):byte;
Var
  i: byte;
Begin
i:=1:
While x <> V[ i ] do
   i:= i+1;
Busca := i
End;
```

a. V = (-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = -2
b. V =(-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = 8

Si el elemento buscado esta en el vector, la búsqueda termina cuando lo encuentra. ¿y si no se encuentra en el vector? Hay que controlar que el índice no supere el límite.

```
Function Busca ( V:TV; N:byte; x:integer ):byte;
Var
  I: byte;
Begin
i:=1:
While (i<= N) and (x <> V[i]) do
    I:= i+1;
If i<=N then
  Busca := i
Else
  Busca:= 0;
End;
```

a. V = (2,4,5,-2,3,8,15,9) ; N = 8; X = -2
b. V = (2,4,5,-2,3,8,15,9) ; N = 8; X = 1

Si en vez de la posición donde se encuentra, se requiere saber si está o no (boolean)

```
Function Esta ( V:TV; N:byte; x:integer ):boolean;
Var
 i: byte;
Begin
i:=1:
While (i< N) and (x <> V[i]) do
    i:= i+1;
Esta := V[i] = x;
End;
```

a. V = (2,4,5,-2,3,8,15,9) ; N = 8; X = -2
b. V = (2,4,5,-2,3,8,15,9) ; N = 8; X = 1

¿qué cambió con respecto al algoritmo anterior y por qué?

Si el arreglo esta ordenado no debe recorrerlo hasta el final para determinar que x no esta en el arreglo. ¿ Como se detecta esta situación?

```
Function Busca ( V:TV; N:byte; x:integer ):byte;
Var
  i: byte;
Begin
i:=1:
While (i< N) and (x > V[ i ]) do
    i:= i+1;
If V[ i ] = x then
  Busca := i
Else
  Busca:= 0;
End;
```

c. V = (-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = -2
d. V =(-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = 1

La siguiente función optimiza la búsqueda en un arreglo ordenado ¿ por qué?

```
Function BusquedaBinaria ( V:TV; N:byte; x:integer ):boolean;
Var
   Medio, Pri, Ult: byte;
Begin
    Pri := 1;
    Ult := N;
    Medio:=(Pri + Ult) DIV 2:
    While (Pri <Ult) and (x <> V[ Medio ]) do
        Begin
        If  x < V[ Medio ] then
            Ult := Medio - 1
       Else
            Pri:= Medio + 1;
     Medio:=(Pri + Ult) DIV 2:
    End;
    BusquedaBinaria:=  x = V[ Medio]
End;
```

a. V = (-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = -2
b. V =(-2, 2, 3,4,5, 8, 9, 15) ; N = 8; X = 1

Proponga un cambio para que la función devuelva la posición donde encuentra x y cero en caso de no encontrarlo.

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

7.- A partir de ejercicio 5 y para una patente dada, mostrar los datos del auto (indicar si no se encontró).

8.- Verificar si todos los elementos de un arreglo de K enteros son pares.

9.- Verificar si un arreglo de reales esta ordenado ascendentemente.

10.- Ingresar un número real X y a continuación N números reales, luego evaluar si X se encuentra entre los N números (no están ordenados).
Informar:
- Si lo encuentra:
  a. el mensaje "el número pertenece al conjunto"
  b. la cantidad de veces que figura
  c. los lugares que ocupa dentro del conjunto de números
- Si no lo encuentra el mensaje "el número no pertenece al conjunto".

11.- Idem 10 pero suponer que los números están ordenados en forma ascendente. ¿debe recorrer siempre el arreglo hasta el final?

6.Ordenacion de los elementos de un vector

Algunos procesos , como la búsqueda, sobre vectores dependen de que los elementos se encuentren, o no, ordenados. Siempre es posible ordenarlo, existen numerosos métodos de ordenación y su eficiencia y rapidez es proporcional a la cantidad y al orden parcial, que presenten sus elementos .
Algunos métodos no aprovechan el orden de partida y repiten procesos de comparación e intercambio una cantidad fija de veces, otros detectan situaciones de orden parcial o total y evitan comparaciones y repeticiones inútiles
Si quiero ordenar en forma ascendente, V1 presenta sus elementos mejor dispuestos que V2

V1 = (2, 3 , 12, 7, 8, 21, 9, 15) V2 = (21, 18, 10, 15, 6, 2, 3, 1)

El **método de burbujeo** recorre el arreglo comparando dos elementos consecutivos V [i] y V[i + 1]
Si están desordenados, V[i] > V[i+1], los intercambia. Este proceso provoca que al finalizar una "pasada" o "barrida" sobre el arreglo, el elemento mayor se desplace a la derecha, ocupando el sitio que le corresponde. Estas barridas se repiten hasta que no se registren cambios (K = 0)
Otra característica de este método es que los elementos de la derecha van quedando ordenados en forma creciente (como mínimo a ese orden parcial se incorpora un elemento en cada pasada), por lo tanto en la próxima barrida la cantidad de elementos a comparar se reduce, al menos, en uno. Puede ocurrir (debido al orden inicial) que a los elementos ya ordenados de la derecha, se incorpore más de un elemento, esto es detectado para que en la próxima barrida no se revisen elementos ya ordenados. Esto se logra guardando en K la posición del último cambio, en la próxima barrida se avanza hasta K –1.

```
Procedure Burbujeo (Var V : TV ; N : byte);
Var
    Aux : real;
   i, K, Tope :byte;
Begin
   Tope:= N ;
    Repeat
       K : = 0:
       For i := 1 to  Tope -1 do
           If V[i] > V[i+1] then
             Begin
             Aux:= V[i];     V[i]:= V[i+1];     V[i+1]:=Aux;
             K:= i ;
             End:;
       Tope:= K ;
    Until K <=1;
End:
```

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

12.- A partir de los dos arreglos del ejercicio 5 que almacenan Patente y Precio de un conjunto de autos se pide desarrollar un procedimiento que los ordene por patente. Luego en otro procedimiento mostrar ambos arreglos.

7. Inserción, eliminación de un elemento. Intercalación de dos arreglos ordenados

- Describa qué representa cada parámetro
- ¿Cuáles se modifican? ¿Cuáles son de entrada , salida o entrada/salida?.

7.1. Eliminar el elemento que se encuentra en la posicion Pos

```
Procedure Elimina ( Var V: Tv; Var N : Byte; Pos:Byte);
Var
  i: Byte;
Begin
   For i:= Pos to N-1 do
      V[ i ] := V[ i+1 ];
   N:=N-1;
End;
```

Realice un seguimiento con el siguiente ejemplo:
N = 5 V= (2,4,7,1,3)
para las posiciones:
a. Pos = 5
b. Pos = 3 V = ?
c. Pos = 1 N = ?

7.2.1 .Insertar el elemento X en la posición Pos

```
Procedure Inserta( Var V:TV; Var N:Byte; Pos: Byte; X:Integer);
Var
  i: Byte;
Begin
   For i := N downto Pos do
     V[ i+1 ] := V[ i ];
   V[ Pos ]:= X;     N := N+1;
End;
```

Realice un seguimiento con el siguiente ejemplo:
N = 5 V= (2,4,7,1,3) X = 5
para las posiciones:
a. Pos = 6
b. Pos = 3 V = ?
c. Pos = 1 N = ?

7.2.2. En un arreglo ordenado, insertar un elemento X (debe determinar la posición), busca de derecha a izquierda y realiza el corrimiento simultaneamente

```
Procedure InsertaOrdenado ( Var V: TV; Var N: Byte; X: Real);
Var
   J: Byte;

Begin
   J:= N;
   While  ( J>0 ) and ( V[ J ] > X ) do
      Begin
      V[ J+1 ] := V [ j ] ; J := J-1;
      End;
    V[ J+1 ] := X ;
     N := N+1;
End;
```

Realice un seguimiento con el siguiente ejemplo:
N = 5 V= (1, 2, 3, 4, 7)
para los valores de X:
a. X = 8
b. X = 5 V = ?
c. X= 0 N = ?

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

13.- Dado un arreglo de enteros, eliminar el máximo elemento, suponer único.
14.- Ingresar un conjunto de números ordenados almacenando en un arreglo los números sin repetición
15.- A partir de un arreglo A de reales desordenado, se pide generar otro arreglo B con los mismos elementos de A ordenados, utilizando el procedimiento *InsertaOrdenado()*

7.3.Intercalación, fusión o mezcla de dos arreglos ordenados , resultando un tercero ordenado

Se deben comparar los elementos de ambos arreglos de a pares, comenzando con las primeras componentes de cada uno, copiando en el tercer arreglo la menor y avanzando sobre el arreglo del cual proviene dicha componente. En la siguiente comparación se enfrentan nuevamente dos componentes y se repite el proceso hasta que uno o ambos arreglos hayan sido procesados en su totalidad. En caso que uno de ellos tenga elementos sin copiar al tercer arreglo (por ser los mayores) , en un ciclo se terminan de pasar.

```
Procedure Intercalacion (V1, V2 : TV ; N, M : integer; Var V3 : TV; Var K : integer);
Var
   t, i,j : integer
```

```
Begin
i:= 1; j:=1; K := 0;
While (i<= N) and (j<=M)do
      Begin
      K:=K +1;
      If V1[i] < V2[j] then
         Begin
         V3[K]:= V1[i]; i:= i + 1 ;
         end
      else
         If V1[i] > V2[j] then
               Begin
               V3[K]:= V2[j]; j:= j =+1 ;
               end
         else
               Begin
               V3[K]:= V1[i];  i:= i + 1 ;   j:= j =+1 ;
               end
      end;
for t := i to N do
      begin
      K:=K +1;
      V3[K]:= V1[t]
      End;
for t:=j to M do
      begin
      K:=K +1;
      V3[K]:= V2 [t]
      End;
End;
```

Realizar un seguimiento del procedimiento Intercalación con los siguientes juegos de datos:

a) N =6 M =3

i										
V1	2	8	10	19	20	22				
J										
V2	1	5	9							
K										
V3										
	1	2	3	4	5	6	7	8	9	10

b) N =4 M =6

i										
V1	1	3	10	19						
J										
V2	1	5	9	10	11	19				
K										
V3										
	1	2	3	4	5	6	7	8	9	10

Otras aplicaciones utilizando arreglos unidimensionales

- **Indice con significado**
- **Arreglos constantes**

El ministerio de educación quiere evaluar el nivel de las escuelas de la provincia. Para ello en cada escuela se toma un examen a todos los alumnos del ciclo inicial (1° a 9° año) Se considera SATISFACTORIO el nivel de una escuela si en cada uno de los nueve años resulto aprobado el 70% o más del alumnado. Se pide hacer un programa que lea desde un archivo los datos de una escuela, son pares : AÑO, NOTA (desordenados) y determine si el resultado fue SATISFACTORIO.
Además liste el total de alumnos que rindieron de la siguiente forma:

Primer año	99
Segundo año	99
.....	
Noveno año	99
Total	999

*Solución : es necesario utilizar por cada año dos contadores, uno para el total de alumnos que rindieron y otro para el total de aprobados. Utilizamos dos arreglos Total y Aprob de 9 elementos cada uno (9 contadores). El año al cual pertenece el alumno determina la posición de la componente que se incrementa, se dice que el **índice tiene un ´significado´**. Se utiliza un **arreglo constante** para almacenar los nombres de los años del EGB y generar el listado pedido*

```
Program  NivelEscuela;
Type
    TV1 = array [1..9] of word;
   TV2 = array [1..9] of string[10];
Const
    Cursos:TV2 = (´Primer´, ´Segundo´ , ....., ´Noveno´);
Var
   Total , Aprob:  TV1;
Procedure IniciaVec(Var V: TV1);
Var
  i :byte;
Begin
  For i := 1 to 9 do
     V[i]:=0;
end;

Procedure Cuantos (Var Total, Aprob: TV1);
Var
  Anio, Nota: byte;   Arch: text;
Begin
   Assign(Arch, 'Escuela.txt'); Reset (Arch);
   While not EOF(Arch) do
     begin
     Readln (Arch,Anio, Nota);     Total[Anio]:= Total[Anio] + 1;
     If Nota >= 7 then
        Aprob[Anio] := Aprob[Anio] + 1;
     End;
Close(Arch)
end;
```

*Los arreglos Aprob y Total son **paralelos**,*

```
Function Satisfactorio(Total, Aprob: TV1) : boolean;
Var
i : byte;
Begin
   i:= 1;
   While (i<=9) and (Aprob[i]/Total[i] >= 0.7 ) do
           i := i + 1;
  Satisfactorio := i>9;
end;
```

Otra forma ineficiente de verificarlo

```
 ……
Var
i, cont : byte;
Begin
  for i:= 1 to 9 do
    If  Aprob[i]/Total[i] >= 0.7 then
        Cont := Cont + 1;
  Satisfactorio := Cont = 9;
end;
```

```
Procedure Escribe (Total: TV1);
Var
  Sum : word;     i : byte;
Begin
  Sum := 0;
  for i:= 1 to 9 do
     Begin
     Sum := Sum + Total[i],
     Writeln(Cursos[i], Total[i]);
     end;
   Writeln(´Total´, Sum);
end;

Begin {P.P.}
IniciaVec(Total);  IniciaVec(Aprob);
Cuantos(Total, Aprob);
If Satisfactorio(Total, Aprob) then
    Writeln(´Nivel stisfactorio´);
Escribe(Total);
end.
```

*Los arreglos Cursos y Total son **paralelos**,*

¿Qué cambiaría si además se requiere el nombre del curso con mayor % de aprobados?

Otro ejemplo:

Leer un conjunto de números, determinar cuantos terminan con 0, 1, 2,.....9

```
Program  Resto10;
Type
   TV = array [0..9] of word;

Procedure Inicia(Var Vec: TV);
Var
  i :byte;
Begin
  For i := 0 to 9 do
     Vec[i]:=0;
end;
```

```
Procedure Cuantos (Var Vec: TV);
Var
  Nro: word;  Indice : byte;
  Arch:text;
Begin
 Assign(Arch, 'Numeros.txt') ; reset(Arch);
 While not Eof(Arch) do
      begin
      Readln (Arch, Nro);
      Indice:= Nro mod 10;
      Vec[Indice] := Vec[Indice] + 1;
      end;
 Close(Arch)
end;

Procedure Escribe (Vec: TV);
.....
Var
   Vec:TV;
Begin {P.P.}
IniciaVec(Vec);
Cuantos(Vec);
Escribe(Vec);
end.
```

Desarrollar programas Pascal que resuelva los problemas propuestos utilizando funciones y procedimientos. Proponer juegos de datos y verificar su funcionamiento.

16.- El área de RRHH de una empresa desea conocer cuántos de sus empleados cumplen años cada uno de los 12 meses. Se conoce Nombre y mes (1..12) de los N empleados, se pide ingresar dicha información para emitir el siguiente listado:

Mes	Cantidad
Enero	100
Febrero	350
Marzo	270
.....	.....

Además indicar el mes con mayor cantidad de cumpleaños.
¿Cómo cambiaría la solución si se considere el nombre del mes es en vez del valor 1..12?

17.- En un taller de reparación de vehículos, se ha registrado en un archivo los siguientes datos de las unidades:

- Tipo (1-particular, 2-carga, 3-transporte de pasajeros, 4-oficial, 5-servicios)
- Costo de la reparación

Se pide leer la información para calcular e informar para cada tipo, el monto recaudado y el porcentaje que representa del total.