

República Bolivariana de Venezuela
Ministerio de Educación Superior
Instituto Universitario de Tecnología Valencia
Departamento de Informática



GUIA TEORICA N° 5
LENGUAJE DE PROGRAMACION I
APUNTES: ARREGLOS.

«Fíate del Señor de todo tu corazón,
y no te apoyes en tu propia prudencia.
Reconócelo en todos tus caminos,
y Él enderezará tus veredas.»
Proverbios 3:5-6

Profesor : DELY M. GIL A.

VALENCIA, 2007

ESTRUCTURAS DE DATOS

Es una colección de datos caracterizados por su organización y por las operaciones que se definen en él.

Dependiendo de la forma de almacenamiento en memoria pueden las estructuras de datos pueden ser DINÁMICAS Y ESTÁTICAS.

Las estructuras de datos ESTÁTICAS son aquellas en que el espacio ocupado en memoria se definen en tiempo de compilación y no puede ser modificado durante la ejecución del programa; por el contrario, en las estructuras de datos DINÁMICAS el espacio ocupado en memoria puede ser modificado en tiempo de ejecución.

ARREGLOS (ARRAY)

Un array es una estructura de datos en la que se almacena una colección de datos del mismo tipo (por ejemplo las notas de los alumnos) . Al tipo se le llama *tipo base* del arreglo. Los datos individuales se llaman *elementos del arreglo*.

TIPOS DE ARREGLOS

Los arrays pueden ser:

- Unidimensionales, también llamados Vectores o listas
- Bidimensionales , denominados tablas o matrices.
- Multidimensionales, con tres o más dimensiones.

CARACTERÍSTICAS DE LOS ARREGLOS

Un array se caracteriza por :

1. Almacenar los elementos del array en posiciones de memoria continua
2. Tener un único nombre de variable que representa a todos los elementos (Notas), y éstos a su vez se diferencian por un índice o subíndice. Notas[1], ..., Notas[n]
3. Acceso directo o aleatorio a los elementos individuales del array.

FORMATO DE DECLARACIÓN DE UN ARRAY

nombre_array = array [*tipo subíndice*] of *tipo*

nombre_array identificador válido

tipo subíndice puede ser de tipo ordinal:

integer, *boolean* o *char*, un tipo *enumerado* o un tipo *subrango*, pero no *REAL*. Existe un elemento por cada valor del tipo subíndice

tipo describe el tipo de cada elemento del vector
todos los elementos de un vector son del mismo tipo

DECLARACIÓN DE UN ARRAY

Las variables de tipo *array* se declaran en la sección *Const*, *Type* o *Var*.

CONST arr_Num : Array[1..2] of Integer = (-5.5);

Declaración en *Type*:

Type

nombres : array[1..30] of string[30];

calif : array[1..30] of real;

numero : array[0..100] of 1..100;

Var

nom : nombres;

califica : calif;

num : numero;

DECLARACIÓN DE UN ARRAY

Declaración en *Var*:

Var

nombres : array[1..30] of string[30];

calif : array[1..30] of real;

numero : array[0..100] of 1..100;

SUBÍNDICE DE UN ARRAY

El subíndice o índice de un array debe ser de tipo simple: entero (integer, Short, Byte, Word), lógico, carácter o enumerado pero NO REAL. Ejemplo :

1..10	integer
True..False	boolean
'C'..'N'	char
azul..marron	enumerados

Type

Letra	= 'A'..'Z';
Conjunto	= array[Letra] of integer;
ConjuntoB	= array[Letra] of boolean;
DiasSemanas	= (Lunes ,Martes, Miercoles, Jueves, Viernes;Sabado, Somingo);
ConjuntoDias	= array[DiasSemanas] of integer;

SUBÍNDICE DE UN ARRAY

Const

Longitud	= 40;
Altura	= 30;

Type

Horizontal	= 1..Longitud;
Vertical	= 1..Altura;
Linea	= Array[Horizontal] of Char;
Tabla	= Array[Vertical] of Linea;

Var

Reticula	: Tabla;
Rango	: Linea;

TIPO DE DATO DE UN ARRAY

El tipo de dato de un array puede ser cualquier dato válido, Ejemplos:

a) Var

Estado	: array[1..100] of boolean;
{Datos enumerados}	

b) TYPE

LucesTrafico	= (Rojo,Verde,Ambar);
--------------	-----------------------

VAR

Trafico	: Array[1..4] of LucesTrafico;
---------	--------------------------------

begin

Trafico[4]	:= Rojo;
...	

End.

TIPO DE DATO DE UN ARRAY

TYPE

Letra	= 'A'..'Z';
ArrayReal	= Array[1..15] of real;
NumeroDeArray	= Array[0..100] of 1..1999;
Linea	= Aray[1..80] of Char;

VAR

Coordenadas	: ArrayReal;
Numero	: NumeroArray
LineaTexto	: Linea;

ARRAY UNIDIMENSIONAL

Un array de una dimensión (vector o lista) es un tipo de datos estructurado compuesto de un número de elementos finitos, tamaño fijo y elementos homogéneos.

Finito indica que hay un último elemento, *tamaño fijo* significa que el tamaño del array debe ser conocido en tiempo de compilación, *homogéneo* significa que todos son del mismo tipo.

ARRAYS COMO PARÁMETROS

Los arrays pueden ser utilizados con parámetros en funciones y procedimientos, el valor de una función no pueden ser ARRAY.

NOTA: Se debe declarar algún parámetro formal del mismo tipo que el array que constituye el parámetro actual en la Sección TYPE

OPERACIONES CON ARRAY

1. Lectura de un vector

```
For i:= 1 to 100 do  
  ReadLn(Notas[i]);
```

- Índice Enumerado:

```
For Dia:= Lunes to Domingo do  
  Read(Cuenta[Dia]);
```

2.- Escritura de un Vector

```
For i:= 1 to Numero do  
  writeLn(Notas[i]:3);
```

3.- Copia de Vectores / Asignación

```
Type  
  ListaReal = array[1..5] of Real;  
Var  
  Alfa,Beta: ListaReal;  
For i:= 1 to 5 do  
  Beta[i]:= Alfa[i]; {Beta = Alfa}
```

4.- Búsqueda

- Lineal (Secuencial)
- Binaria

5.- Ordenamiento

- Inserción, Selección, Burbuja o Intercambio, Shell, QuickSort (Ordenación rápida), Ordenación por Fusión o Mezcla

6.- Inserción

7.- Eliminación

Program Vector_edades; {Teoria5_1.pas}

```
{El siguiente programa captura 20 edades
y las muestra en forma ascendente por medio
de arrays}
Uses Crt;
Const
  MaxPersonas = 10;
Var
  edades : array [1..MaxPersonas] of byte;
  i,j,paso : byte;
begin
  ClrScr;
  {lectura de array}
  for i:=1 to MaxPersonas do
  begin
    gotoxy(10,5);
    ClrEol;
    Write('Edad de la ',i,' persona : ');
    ReadLn(edades[i]);
  end;
  {ordenación}
  for i:=1 to MaxPersonas-1 do
  begin
    for j:=i+1 to MaxPersonas do
    begin
      if edades[i]>edades[j] then
      begin
        paso :=edades[i];
        edades[i]:=edades[j];
        edades[j]:=paso
      end
    end;
    WriteLn(edades[i]) {escritura del array}
  end;
  Readkey
end.
```

Parámetros por valor.

```
Program Identidad; {Teoria5_2.pas}
{Programa que verifica si dos vectores son iguales}
Uses Crt;
Const
  Limite = 5;
Type
  Vector = Array[1..Limite] of integer;
Var
  M,N : Vector;
  Res_Identidad : Boolean;

Procedure Llenar_Vector(Var Z:Vector);
VAR i:Byte;
Begin
  Randomize;
  for i:= 1 to Limite do
    Z[i]:=Random(41)-20; {Rango entre [-20..20]}
  End;

Procedure Imprimir(Z:Vector;Dim:byte);
var i :byte;
begin
  for i:= 1 to Dim do
    write(Z[i]:7);
  writeln
End;
```

```
Function Identidad ( A,B : Vector) : Boolean;
Var
  I: Byte ;
Begin
  i:=1;
  While (I<Limite) and (A[I] = B[I]) do
    Inc (I);
  Identidad := A[I] = B[I]
End;

{P.P}
Begin
  ClrScr;
  Llenar_Vector(M);Imprimir(M,Limite);
  Llenar_Vector(N);Imprimir(N,Limite);
  Res_Identidad:= Identidad(M,N);
  If Res_Identidad then
    Write('Los vectores son iguales')
  Else
    Write('Los vectores no son iguales');
  readkey
End.
```

Parámetros por referencia.

```
Program Suma_Vectores; { Teoria5_3.pas}
{El siguiente programa suma dos vectores }
Const
  N          = 12;
Type
  Indice     = 1..N
Type
  Vector     = Array[Indice] of Integer;
Var
  A,B,S      : Vector;
```

```
Procedure Leer_Vector(Var X:Vector);
Var J: 1..N;
Begin
  For j:= 1 to N do
    Read(X[j])
End;

Procedure Escribir_Vector(X: Vector);
Var j: 1..N;
Begin
  For j:= 1 to N do
    Writeln(X[j])
End;

Procedure Sumar_Vectores(X,Y: Vector; var Z:Vector);
{Nota: se debe declarar algún parámetro del mismo tipo que
el array que constituye el parámetro actual}

Var: i: 1 ..N;
Begin
  For i:= 1 to N do
    Z[i] := X[i] + Y[i];
End;

{P.P}
Begin
  Writeln('Para cada vector introduzca', N:1, ' elementos: ');
  Writeln('Introduzca datos del primer vector: ');
  Leer_Vector (A);
  Writeln('Introduzca datos del segundo vector: ');
  Leer_Vector (B);
  Sumar_Vectores(A,B,S);
  Escribir_Vector(S);
  Writeln('Presione tecla para continuar');
  Readkey
End
```

ARRAYS PARALELOS

Dos o más arrays que utilizan el mismo subíndice para referirse a términos homólogos se llaman arrays paralelos.

Basados en el programa anterior se tienen las edades de 'x' personas, para saber a que persona se refiere dicha edad se puede usar otro arreglo en forma paralela y asociarle los nombres de manera simultánea con las edades.

Nombres[1]	Fidel	Edades[1]	82
Nombres[2]	Ely	Edades[2]	28
	.		.
	.		.
	.		.
Nombres[10]	Juan	Edades[10]	9

```
Program Paralelo_edades;
{El siguiente programa captura 10 edades y nombres por medio de
arrays paralelos y los muestra ordenados en forma ascendente}
Uses Crt;
Const
  MaxPersonas = 10;
Var
  edades      :array [1..MaxPersonas] of byte;
  nombres     :array [1..MaxPersonas] of string [10];
  aux_nom     :string[10];
  i,j,aux_edad :byte;

begin
  ClrScr;
  {lectura de arrays paralelos de manera simultánea}
  for i:=1 to MaxPersonas do
    begin
      gotoxy(10,5); ClrEol;
      Write(i,'- Nombre : ','Edad : ');
      gotoxy(23,5);ReadLn(nombres[i]) ;
      gotoxy(48,5);ReadLn(edades[i])
    end;

  {ordenación}
  for i:=1 to MaxPersonas-1 do
    begin
      for j:=i+1 to MaxPersonas do
        begin
          if edades[i]>edades[j] then
            begin
              aux_edad :=edades[i];      edades[i] :=edades[j];
              edades[j] :=aux_edad;      aux_nom :=nombres[i];
              nombres[i]:=nombres[j];    nombres[j]:=aux_nom
            end
          end;
        WriteLn(nombres[i]:10,' ',edades[i]:3)
        {escritura de los arrays paralelos}
      end;
      Readkey
    end.
End.
```

ARRAY BIDIMENSIONAL

Un array bidimensional (tabla o matriz) es un array con dos índices, al igual que los vectores que deben ser ordinales o tipo subrango.

		Columnas		
		1	2	3
Filas	1	A[1,1]	A[1,2]	A[1,3]
	2			
	3			
	4	A[4,1]	A[4,2]	A[4,3]

Para localizar o almacenar un valor en el array se deben especificar dos posiciones (dos subíndices), uno para la fila y otro para la columna.

FORMATO DE ARRAYS BIDIMENSIONALES

identificador = array [índice1, índice 2] of tipo de elemento

MANIPULACIÓN DE TABLA

Recorrido por fila

For Fila:= 1 to 3 do

For Columna:= 1 to 4 do

Readln(A[filas,columna]);

Recorrido por Columnas

For Columna:= 1 to 4 do

For Fila:= 1 to 3 do

Readln(A[columna,filas]);

EJEMPLO ARRAY BIDIMENSIONAL

Supóngase que se desea almacenar las calificaciones de 5 alumnos obtenidas en 3 exámenes y mostrar en orden ascendente sus promedios respectivamente. En este caso se usará un array bidimensional (tabla o matriz) de 5 filas y 4 columnas en la cual se almacenará las calificaciones de 3 exámenes en 3 columnas y la cuarta columna se utilizará para almacenar su promedio respectivo, además de un array unidimensional (vector) donde en forma paralela se almacenarán los nombres de los alumnos de la siguiente forma :

		Exam 1	Exam 2	Exam3	Promedio
Alumno[1]	Fidel	10	10	10	10
Alumno[2]	Ely	20	20	20	20
Alumno[3]	Luis	15	15	15	15
Alumno[4]	Juan	05	05	05	05

```
Program Matriz_Vector;
{El siguiente programa captura las calificaciones de 5 alumnos en 3
exámenes, y despliega en pantalla los promedios ordenados en forma
descendente }
Uses Crt;
Const
  MaxAlumno = 5;
  MaxExamen = 4;{Columna 4 almacena el promedio}
Var
  Alumno   :array[1..MaxAlumno]      of string[10];
  examen   :array[1..MaxAlumno,1..MaxExamen] of real;
  aux_examen :array[1..MaxExamen]    of real;
{reserva 20 posiciones de memoria de datos reales :
5 filas por 4 columnas}
  promedio :real;
  aux_alumno :string [10];
  i,j,col,ren :byte;

begin
  ClrScr;
  {lectura de arrays paralelos de manera simultánea}
  gotoxy(5,5);Write('Nombre');
  gotoxy(20,5);Write('Examen1 Examen2 Examen3 Promedio');
  col:=5;ren:=6;
  for i:=1 to MaxAlumno do
    begin
      gotoxy(col,ren);
      ReadLn(alumno[i]); {lectura de vector}
      col:=22;promedio:=0;
      for j:=1 to MaxExamen-1 do
        begin
          gotoxy(col,ren);
          ReadLn(examen[i,j]); {lectura de matriz}
          promedio:=promedio+examen[i,j];
          col:=col+10
        end;
      examen[i,j+1]:=promedio/3;
      gotoxy(col,ren);Write(promedio/3:3:2);
      inc(ren);
      col:=5
    end;
```

```
{ordenación}
for i:=1 to MaxAlumno-1 do
  for j:=i+1 to MaxAlumno do
    begin
      if examen[i,MaxExamen]<examen[j,MaxExamen] then
        begin
          {intercambio de nombres en vector}
          aux_alumno:=alumno[i];
          alumno[i] :=alumno[j];
          alumno[j] :=aux_alumno;
          {intercambio de calificaciones en matriz}
          move(examen[i],aux_examen,SizeOf(aux_examen));
          move(examen[j],examen[i],SizeOf(aux_examen));
          move(aux_examen,examen[j],SizeOf(aux_examen))
        end
      end;
    {recorrido de matriz y vector}
    gotoxy(25,14);Write('Datos ordenados');
    gotoxy(5,16);Write('Nombre');
    gotoxy(20,16);Write('Examen1 Examen2 Examen3 Promedio');
    col:=5;ren:=17;
    for i:=1 to MaxAlumno do
      begin
        gotoxy(col,ren);
        Write(alumno[i]);
        col:=22;
        for j:=1 to MaxExamen do
          begin
            gotoxy(col,ren);
            Write(examen[i,j]:3:2);
            col:=col+10
          end;
        col:=5;
        inc(ren)
      end;
    readkey
  end.
```

**{TAREA: TRAER ESTE EJEMPLO CON UNA UNIDAD
PARA VALIDACIÓN Y SUBPROGRAMAS}**

INSERCIÓN DE ELEMENTOS EN VECTORES

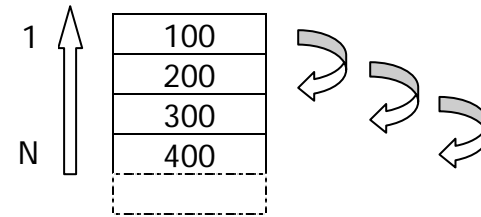
Requiere de tres acciones:

- Desplazar los elementos para liberar el espacio para el valor a insertar
- Asignar el valor en la posición
- Modificar la dimensión del arreglo

Casos de Inserción:

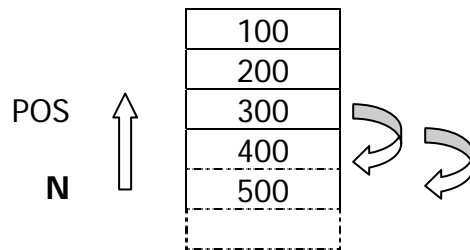
- Al inicio del arreglo
- Entre elementos del arreglo
- Al final del arreglo

INSERCIÓN AL INICIO DEL ARREGLO (POS = 1)



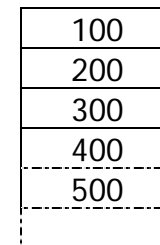
```
For i:= N downto 1 do
    Vector[i+1] := Vector[i];
Vector[1] := Valor
N      := Inc(N);
```

INSERCIÓN ENTRE ELEMENTOS (1 > POS < N)



```
For i:= N downto POS do
    Vector[i+1] := Vector[i];
Vector[POS] := Valor
N      := Inc(N);
```

INSERCIÓN AL FINAL DEL ARREGLO (POS = N+1)



```
Vector[POS] := Valor
N      := Inc(N);
```

ELIMINACIÓN DE ELEMENTOS EN VECTORES

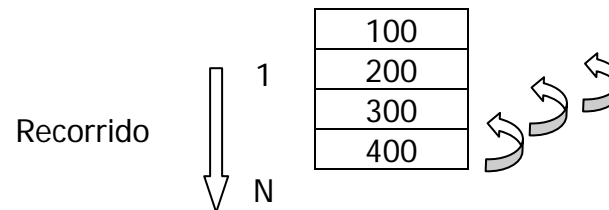
Requiere de tres acciones:

- Desplazar los elementos para eliminar el valor
- Modificar la dimensión del arreglo

Casos de Eliminación:

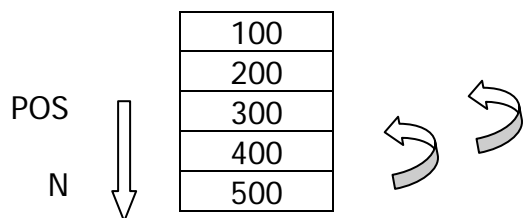
- El primer elemento del arreglo (POS=1)
- Entre elementos del arreglo
- Al final del arreglo (POS = N)

ELIMINACIÓN AL INICIO DEL ARREGLO (POS = 1)



```
For i:= 2 to N-1 do  
    Vector[i] := Vector[i+1];  
N      := Dec(N);
```

ELIMINACIÓN ENTRE ELEMENTOS (1 > POS < N)



```
For i:= POS to N-1 do  
    Vector[i] := Vector[i+1];  
N      := Dec(N);
```

ELIMINACIÓN AL FINAL DEL ARREGLO (POS = N+1)

100
200
300
400
500

```
N      := Dec(N);
```

MANEJO DE MATRICES

Diagonal Principal ($i = j$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

For i:= 1 to N do
A[i,i]:= 0;

MANEJO DE MATRICES

Diagonal Secundaria ($i+j = N+1$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

For i:= 1 to N do
Begin
j:= N - i + 1;
A[i,j]:= 0;
End;

For i:= 1 to N do
if (i+j = N+1) then
A[i,i]:= 0;

MANEJO DE MATRICES

Diagonal Principal Superior ($i < j$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

for i:= 1 to N-1 do
for j:= i+1 to N do
A1[i,j]:= 0;

for i:= 1 to N do
for j:= 1 to N do
if (i<j) Then
A1[i,j]:= 0;

MANEJO DE MATRICES

Diagonal Secundaria Superior ($i+j < N+1$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

for i:= 1 to N-1 do
for j:= 1 to N-i do
A1[i,j]:= 0;

for i:= 1 to N do
for j:= 1 to N do
if (i+j < N + 1) then
A1[i,j]:= 0;

MANEJO DE MATRICES

Diagonal Principal Inferior ($i > j$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

```
for i:= 2 to N do
  For j:=1 to i-1 do
    A1[i,j]:= 0;
```

```
for i:= 1 to Dim do
  for j:= 1 to Dim do
    if (i>j) Then
      A1[i,j]:= 0;
```

MANEJO DE MATRICES

Diagonal Secundaria Inferior ($i+j > N+1$)

A11	A12	A13	A14	A15	A16
A21	A22	A23	A24	A25	A26
A31	A32	A33	A34	A35	A36
A41	A42	A43	A44	A45	A46
A51	A52	A53	A54	A55	A56
A61	A62	A63	A64	A65	A66

```
for i:= 2 to N do
  for j:= N-i+2 to N do
    A1[i,j]:= 0;
```

```
for i:= 1 to N do
  for j:= 1 to N do
    if (i+j > N +1) then
      A1[i,j]:= 0;
```

MANEJO DE MATRICES

Producto de Matrices

```
for i:= 1 to N do
  For j:= 1 to N do
    begin
      prod[i,j] := 0;
      For k:= 1 to N do
        prod[i,j]:= prod[i,j] + a[i,k]*b[k,j];
      end;
```

MATRIZ SIMÉTRICA

Una matriz A se dice que es simétrica si $A(i,j) = A(j,i)$ para todo i,j dentro de los límites de las matriz. (MXM)

MATRIZ ASIMÉTRICA

$$A(i,j) = - A(j,i)$$

MATRIZ TRASPUESTA

Una matriz M se dice que es traspuesta (MT) si $M(i,j) = MT(j,i)$ para todo i,j dentro de los límites de las matriz. (MXN)

QUIZ DE VECTORES

1.- Realice la corrida en frío en el siguiente programa y optimice el código (variables parámetros..)

<pre> Program Ins_Name; Uses Crt; Const MaxLista = 10; Type Cad20 = string[20]; Vector = array [1..MaxLista] of Cad20; Var Lista :Vector; n,p,i : byte; e : cad20; error : boolean; Procedure Ins_Lista(var List: vector; Var N: byte; var error: Boolean; p:Byte; e:cad20); Var i:byte; Begin error := (n=MaxLista) or (p>n+1); If not error then begin for i:= n downto p do List[i+1] := List [i]; List[p] := e; n := n+1 End End; End; Begin Repeat Write('Número de elementos: '); Readln(n); until n<=MaxLista; writeln('Introduzca los elementos'); for i:= 1 to n do Readln(Lista[i]); writeln; write('Indique la posicion donde desea insertar : '); readln(p); write('Escriba la cadena a insertar: '); readln(e); Ins_Lista(Lista,N,error,p,e); If not error then for i:= 1 to n do writeln(Lista[i]) else Begin writeln('Error'); for i:= 1 to n do writeln(Lista[i]) End End; End; End. </pre>	<pre> Program Eli_Random; Uses Crt; Const MaxLista = 10; Type Vector = array [1..MaxLista] of Integer; Var Lista :Vector; n,p,i : byte; error : boolean; Procedure Eli_Lista(var List: vector; Var N: byte; var error: Boolean; p:Byte); Var i:byte; Begin error := (n=0) or (p>n); If not error then begin for i:= p to n-1 do List[i] := List [i+1]; n := n-1 End End; End; Begin Randomize; Repeat Write('Numero de elementos[1-10]: '); Readln(n); until n<=MaxLista; for i:= 1 to n do Lista[i]:=Random(20)+1; writeln; for i:= 1 to n do writeln(i:2,'Ø : ',lista[i]:8); writeln; write('Indique la posicion donde desea borrar : '); readln(p); Eli_Lista(Lista,N,error,p); If not error then for i:= 1 to n do writeln(Lista[i]:8) else Begin writeln('Error'); for i:= 1 to n do writeln(Lista[i]:8) End; End; writeln; readkey End. </pre>
---	--

QUIZ DE MATRICES

1.- Realice la corrida en frío en el siguiente programa.

```

Program Operaciones_Mat;
Uses
  Crt;
Const
  MaxDim = 5;
Type
  tabla = array[1..MaxDim,1..MaxDim] of integer;
Var
  a,b,prod      : tabla;
  i,j,DimA,DimB_f,DimB_c : byte;
  res           : integer;
  opcion       : char;

(*****LEER*****)
Procedure leer(var t:tabla; var Dim:byte);
var
  i,j,Dim_C : byte;

begin
  ClrScr;
  gotoxy(20,10);write('Dimensiones del vector:');
  gotoxy(10,12);
  write('Ingrese dimensión fila(1-5):');
  readln(Dim);
  gotoxy(10,14);
  ;write('Ingrese dimensión columna(1-5):');
  repeat
    readln(Dim_c) ;
  until (Dim=Dim_c);
  gotoxy(20,18);write('Lectura del vector:');
  for i:= 1 to Dim do
    begin
      for j:= 1 to Dim do
        t[i,j]:= random(9) +1;
      end;
    end;
end;
(*****PRODUCTO*****)
Procedure matrizProducto(a,b:tabla; Dim:byte; var
prod:tabla);
var
  k:byte;
begin
  for i:= 1 to Dim do
    for j:= 1 to Dim do
      begin
        prod[i,j] := 0;
        for k:= 1 to Dim do
          prod[i,j]:= prod[i,j] + a[i,k]*b[k,j];
        end;
      end;
    end;
end;

```


A

B

=

C

```

For i:= 1 to FA
  For j:= 1 to CB
    .....
  For k:= 1 to CA

```

FA=FC ;CB=CC

```

(*****PRINT*****
Procedure Imprimir(A1:tabla;Dim:byte);
var
  i,j :byte;
begin
  for i:= 1 to Dim do
    begin
      for j:=1 to Dim do
        write(A1[i,j]:7);
      writeln;
    end;
end;

(*****DIAG PP*****
Function DiagPP(A1:tabla;Dim:byte): integer;
{ i = j R}
var
  i,j :Byte; suma:integer;
begin
  suma:= 0;
  for i:= 1 to Dim do
    suma:= suma + A1[i,i];
  DiagPP:= suma;
end;

(*****DIAG INF PP*****
Function DiagInfPP(A1:tabla;Dim:byte): integer;
{ i>j R}
var
  i,j :Byte; suma:integer;
begin
  suma:= 0;
{
  for i:= 2 to Dim do
    for j:= 1 to i-1 do
      suma:= suma + A1[i,j];
    }
  for i:= 1 to Dim do
    for j:= 1 to Dim do
      if (i>j) Then
        suma:= suma + A1[i,j];
    }
  DiagInfPP:= suma;
end;

(*****DIAG SUP PP*****
Function DiagSupPP(A1:tabla;Dim:byte):integer;
{ i < j R}
var
  i,j :byte;suma: integer;
begin
  suma:= 0;
{
  for i:= 1 to Dim-1 do
    for j:= i+1 to Dim do
      suma:= suma + A1[i,j];
    }
  for i:= 1 to Dim do
    for j:= 1 to Dim do
      if (i<j) Then
        suma:= suma + A1[i,j];
    }
  DiagSupPP:= suma;
end;

```

QUIZ DE MATRICES

1.- Realice la corrida en frío en el siguiente programa.

```
(*****DIAG SEC*****)
Function DiagSec(A1:tabla;Dim:byte): integer;
{ i+j = DIM +1 R}
var
i,j :Byte; suma:integer;
begin
suma:= 0;
for i:= 1 to Dim do
begin
j:= Dim -i +1; {if (i+j = Dim+1) then}
suma:= suma + A1[i,j];
end;
DiagSec:= suma;
end;

(*****DIAG SUP SEC*****)
Function DiagSupS(A1:tabla;Dim:byte): integer;
{ i+j < DIM +1 R}
var
i,j :Byte; suma:integer;
begin
suma:= 0;
{ for i:= 1 to Dim-1 do
for j:= 1 to Dim -i do
suma:= suma + A1[i,j];
for i:= 1 to Dim do
for j:= 1 to Dim do
if (i+j < Dim +1) then
suma:= suma + A1[i,j];
DiagSupS:= suma;
end;
}

(*****DIAG INF SEC*****)
Function DiagInfS(A1:tabla;Dim:byte): integer;
{ i+j > DIM +1 R}
var
i,j :Byte; suma:integer;
begin
suma:= 0;
{for i:= 2 to Dim do
for j:= Dim-i+2 to Dim do
suma:= suma + A1[i,j];}
for i:= 1 to Dim do
for j:= 1 to Dim do
if (i+j > Dim +1) then
suma:= suma + A1[i,j];
DiagInfS:= suma;
end;

begin
leer(A,DimA);
ClrScr;
writeln('Vector A:');
Imprimir (A,DimA);
writeln;
res:= DiagPP(A,dimA);
writeln('La suma de la diagonal Principal es ',res);
res:= DiagSupPP(A,DimA);
writeln('La suma por encima ',res);
res:= DiagInfPP(A,DimA);
writeln('La suma por debajo ',res);
res:= DiagSec(A,dimA);
writeln;
writeln('La suma de la diagonal Secundaria es ',res);
res:= DiagSupS(A,DimA);
writeln('La suma por encima ',res);
res:= DiagInfS(A,DimA);
writeln('La suma por debajo ',res);
gotoxy(20,20);
write('Presione tecla para continuar');
readkey;
Leer(B,DimA);
matrizProducto(A,B,DimA,prod);
Imprimir(Prod,DimA);
gotoxy(20,20);
write('Presione tecla para continuar');
readkey;
end.
```

Matriz transpuesta

1	2	3
4	5	6

A

1	4
2	5
3	6

B

```

FB:=CA;
CB:=FA;
For i:= 1 to FA do
For j:= 1 to CA do
B[j,i] := A[i,j];

```

TALLER DE ARREGLOS

Dado un arreglo A de n elementos enteros se quiere:

1.- Generar una secuencia de enteros que contenga las posiciones de aquellos elementos del arreglo que sean múltiplos de un entero m dado.

2.- Leer un número y el lugar donde se cambiará dicho número por otro elemento del vector. Imprimir el arreglo, antes y después del cambio.

3.- Se quiere crear otro arreglo, tal que cada uno de sus elementos sea la suma de los elementos opuestos en el arreglo dado.

Ejemplo: Arreglo dado A = [9,5,3,10,2,8,1]
Arreglo resultante B = [10,13,5,10]

4.- Generar tres arreglos que contenga los elementos negativos, cero y positivos del arreglo.

5.- Generar otro arreglo que contenga las posiciones de los elementos del arreglo dado que sean iguales a un valor x dado.

Ejemplo: Arreglo dado A = [4,6,8,2,6,9,6,1]
x = 6
Arreglo resultante B = [2,5,7]

6.- Almacenar los elementos mayores y menores que la media, almacenarlos en vectores diferentes.

7.- Obtener otro arreglo tal que sus elementos sean la diferencia de los elementos sucesivos del arreglo dado.

Ejemplo: Arreglo dado A = [4,6,8,2,6,9,5,1]
Arreglo resultante B = [-2,-2,6,-4,-3,4,4]

8.- Obtener dos arreglos tal que sus elementos sean los números pares y números impares del arreglo.

9 - Generar una secuencia de enteros que contenga aquellos elementos, cuyo valor sea múltiplo de un entero m dado.

0.- Eliminar del arreglo los elementos repetidos, sin dejar posiciones vacías en el mismo.

Ejemplo: Arreglo dado A = [4,6,8,2,6,9,6,1]
Arreglo resultante B = [4,6,8,2,9,1]

TALLER DE MATRICES

Dado una matriz A de NxM (Dimensión: dato ingresado por el usuario) , cuyos elementos son enteros y generados aleatoriamente, se quiere:

1.- Dada una matriz B (aleatoria) ,obtener una matriz S la cual es el resultado de $A + B$. Para ésta operación se debe verificar que las matrices tengan la misma dimensión.

2.- Obtener la posición de todo elementos de la matriz que sea múltiplo de un entero h dado(dado por el usuario).

3.- Dado un arreglo T con p elementos (ingresado por el usuario), indicar la posición de todos los elementos de T que aparezcan en la matriz (sólo la primera aparición); se debe indicar también la posición en la matriz.

4.- Modifique el algoritmo anterior de manera de obtener la posición de todas las apariciones de los elementos de T en la matriz.

5.- Dados i y j , intercambiar la fila i con la fila j.

6.- Modifique la solución anterior, para intercambiar la fila i, con la columna j.

7.- Calcular la suma de sus elementos mediante una función

8.- Obtener una matriz P la cual es el resultado de $A * A$ (investigar la fórmula de multiplicación de matrices).

9.- Obtener la matriz A^t , donde:

- A es la matriz dada.
- A^t es la matriz transpuesta de A, es decir, aquella matriz cuyos elementos son los mismos de A, pero la columna i-ésima de A^t es la fila i-ésima de A.

Ejemplo: si $A = \begin{pmatrix} 1 & 2 & 3 \\ 9 & 3 & 5 \\ 7 & 4 & 8 \end{pmatrix}$, entonces $A^t = \begin{pmatrix} 1 & 9 & 7 \\ 2 & 3 & 4 \\ 3 & 5 & 8 \end{pmatrix}$

0.- Si $N = M$ (Es decir, la dimensión de A es NxN) Calcular la sumatoria de los elementos de cada diagonal de A. Almacenar los resultados en un arreglo. ¿Cuál debe ser la dimensión del arreglo).