UTN - FRRO

Algoritmos
y
Estructuras de Datos

* Arreglos

Prof. CLAUDIA DANIA
Magister en Docencia Universitaria
Analista Universitario de Sistemas
Licenciada en Sistemas de Información

ARREGLOS: Tipo de Datos Estructurados

Hasta el momento se han estudiado los datos de tipo simple cuya característica común es que cada variable representa a un solo dato individual.
La diferencia con los datos estructurados es que un único identificador puede representar a múltiples datos individuales, siendo referenciados en forma separada cada uno de éstos por medio de índices.

ARREGLOS UNIDIMENSIONALES

Puede ser considerado como una lista de datos, todos del mismo tipo, identificados en forma global mediante un nombre de no más de 8 caracteres formado por letras y/o números.
Cada elemento (cada dato) se lo referencia con el nombre del arreglo seguido de un índice (también llamado subíndice) encerrado entre corchetes.

VAR nombre del array: ARRAY [tipo de indice] OF tipo del arreglo;

El tipo de índice puede ser de tipo simple: entero, char, booleano, enumerativo o subrango, **nunca real.**
El tipo de array puede ser cualquiera: entrero, char, booleano, real.
Recordar que no se pueden mezclar datos de distintos tipos en un mismo arreglo.

Se puede declarar de dos maneras:

```
VAR columna: ARRAY [1 .. 100] OF integer;

        ó

TYPE indice = 1 . . 100;
VAR columna: ARRAY [indice] OF integer:
```

Los elementos del arreglo pueden ser utilizados en estructuras de control, comparación, sentencias de asignación, sentencias de lectura y escritura, etc., identificándolos con el valor del índice correspondiente, el cual puede ser una constante, una variable o una expresión.

Ejemplo con índice tipo enumerativo:

```
TYPE color = (rojo , blanco , azul , amarillo , verde);
VAR  indice: color;
     valores: ARRAY [color] OF integer;
     .....
BEGIN
     FOR indice:='rojo' TO 'verde' DO
          WRITELN ( valores[indice] );
     ....
END.
```

ORDENAMIENTO en Unidimensionales

Ordenar un arreglo unidimensional de menor a mayor (Falso Burbuja):

```
PROCEDURE orden1,
(* intercambiar de menor a mayor los elementos de un array unidimensional *)
BEGIN
     FOR i := 1 TO n-1 DO
          FOR k:= i + 1 TO n DO
               IF vec [i] > vec [k] THEN
                    BEGIN
                         aux:= vec [i];
                         vec [i] := vec [k];
                         vec [k] := aux
                    END
END;
```

ARREGLOS MULTIDIMENSIONALES

Valen las mismas consideraciones de los arreglos unidemensionales, en cuanto a los tipos de datos y tipos de índices.
Dentro de los arreglos multidimensionales daremos ejemplos de los bidimensionales compuestos por filas y columnas.
La primera dimensión (es decir el primer índice) se refiere al número de fila y la segunda dimensión (segundo índice) al número de columna. En el caso de tres dimensiones, se lo considera como si fuese un conjunto de tablas encuadernadas, donde la tercer dimensión indicaría la página del cuaderno.

VAR nombre del array: ARRAY (tipo indice1, tipo indice2,...,tipo indiceN) OF tipo del arreglo;

Se lo puede declarar como:

```
VAR tabla : ARRAY [0 .. 100 , 1 .. 50] OF real;
```

ó

```
TYPE indice1 = 0 .. 100;
     indice2 = 1 .. 50;
VAR tabla : ARRAY [indice1 , indice2] OF real;
```

ORDENAMIENTO en Bidimensionales

Ordenar un arreglo bidimensional de menor a mayor, por una columna en particular (Falso Burbuja):

```
PROCEDURE orden2;
CONST    fila , col = 25;
         z = 3;
VAR  i , r , k , aux : INTEGER;
     mat : ARRAY [fila,col] of INTEGER;
BEGIN
     FOR i := 1 TO fila-1 DO
          FOR r := i + 1 TO fila DO
               IF mat [i,z] > mat [r,z] THEN
                    FOR k := 1 TO col DO
                         BEGIN
                              aux:= MAT [i,k];
                              MAT [i,k] := MAT [r,k];
                              MAT [r,k] := aux
                         END
END;
```

OPERACIONES con ARREGLOS

Unidimensionales

```
PROCEDURE carga (VAR vec: ARRAY[1..N] OF REAL);
VAR I: INTEGER;
BEGIN
     FOR i := 1 TO N DO
          READ ( vec[i])
END;
```

```
PROCEDURE exhibir (VAR vec: ARRAY[1..N] OF REAL);
VAR i: INTEGER;
BEGIN
     FOR i := 1 TO N DO
          WRITE ( vec[i] )
END;
```

Bidimensionales

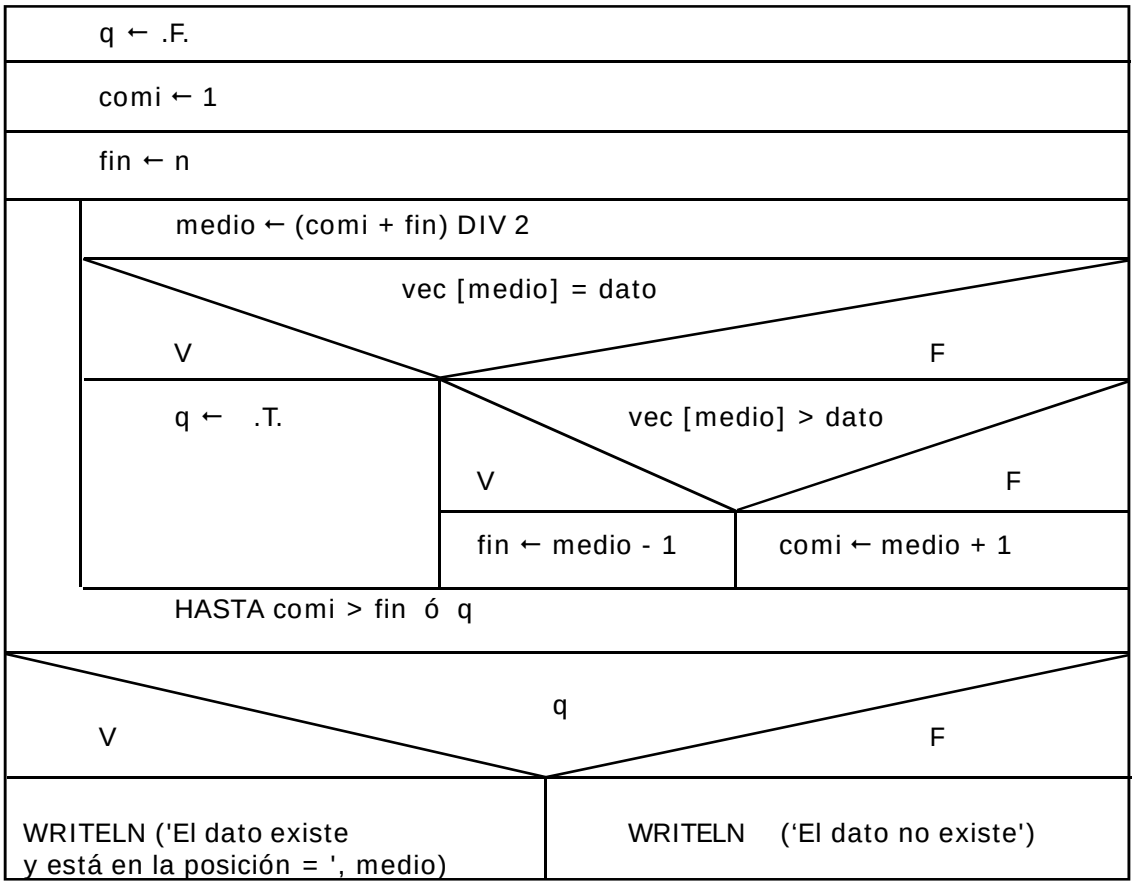
```
PROCEDURE carga (VAR mat: ARRAY[1..fila, 1..col] OF REAL);
VAR i , j : INTEGER;
BEGIN
     FOR i:= 1 TO fila DO
          FOR j := 1 TO col DO
               READ ( mat [i,j] )
END;
```

```
PROCEDURE exhibir (VAR mat: ARRAY[I..fila, 1..col] OF REAL);
VAR i , j : INTEGER;
BEGIN
     FOR i := 1 TO fila DO
          BEGIN
               FOR j := 1 TO col DO
                    WRITE ( mat [i , j] );
               WRITELN
          END
END;
```

BUSQUEDA DICOTOMICA en Unidimensionales

```
PROCEDURE buscar;
(* el arreglo se llama VEC, FIN es igual a la longitud del mismo,*)
(* DATO es una variable cualquiera cuyo contenido puede ser *)
(* igual ó no al contenido de algunas de las celdas *)
VAR   comi, fin, medio: INTEGER;
      q: BOOLEAN;
BEGIN
      q := .F.;
      comi := 1;
      fin := n;
      REPEAT
            medio := (comi + fin) DIV 2;
            IF vec [medio] = dato THEN q := .T.
                                  ELSE
                                        IF vec [medio] > dato THEN fin := medio - 1
                                                              ELSE comi := medio + 1
      UNTIL com > fin .OR. q
      IF q   THEN WRITELN ( 'El Dato existe y está en la posicion = ', medio)
             ELSE WRITELN ('El dato no existe')
END;
```

BUSCA

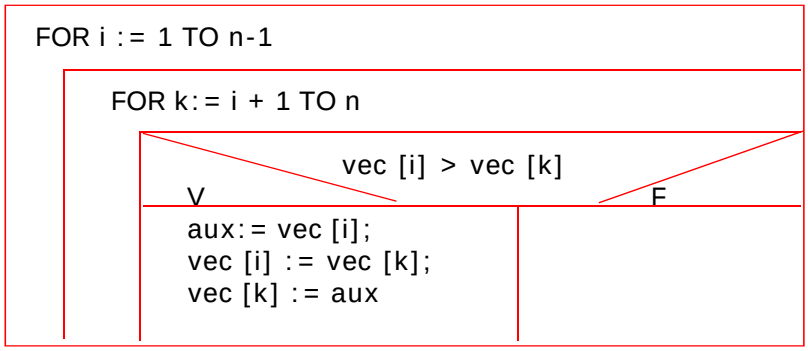


MERGE en Unidimensionales

```
program mezcla  (input, output);
uses crt;
const n=9;
type  w= array[1..n]of integer;
var a,b,c:w;
    i,j,aux,ca,cb:integer;

procedure mezclar;
begin
 ca:=1;
 cb:=1;
 i:=1;
 repeat
      if b[cb]< a[ca] then
                   begin
                      c[i]:= b[cb];
                      cb:=cb+1;
                    end
                else
                     begin
                      c[i]:= a[ca];
                      ca:=ca+1;
                     end;
     i:=i+1
 until (ca > 4)or(cb > 5);

     if ca > 4 then
                  repeat
                    c[i]:=b[cb];
                    cb:=cb+1;
                    i:=i+1;
                  until cb > 5

            else
                  repeat
                     c[i]:=a[ca];
                     ca:=ca+1;
                     i:=i+1;
                  until ca>4;
end;
```

```
procedure carga(var z:w ; lim:integer);
begin
   write('Ingrese los Numeros');
   for i := 1 to lim do readln(z[i]);
end;

procedure orden(var z:w ; lim:integer);
begin
  for i:= 1 to lim-1 do
   for j:= i+1 to lim do
        if z[i] > z[j] then begin
                              aux:=z[i];
                              z[i]:=z[j];
                              z[j]:=aux;
                            end;
end;

begin
    carga(a,4);
    carga(b,5);
    orden(a,4);
    orden(b,5);
    mezclar;
    for i:= 1 to n do writeln(c[i])  ;
    readkey();
end.
```

ORDENAMIENTO en Unidimensionales



ORDENAMIENTO en Bidimensionales