6. ARREGLOS Y MATRICES

6.1 ARREGLOS

De manera informal, un **arreglo** es una lista finita e indexada de elementos de un mismo tipo. Una lista se dice indexada si para cada elemento de la lista existe un único número natural que permite hacer referencia a él. A tal número natural se le llama el **índice o subíndice** del elemento que referencia.

De manera formal un arreglo de tamaño n de elementos de tipo A, es un elemento del espacio n-dimensional del conjunto A, es decir, x es un arreglo de tamaño n del tipo A si y solo si $x \in A^n$

Ejemplos.

- [3, -6, 1, 24, 8] es un arreglo de tamaño cinco y tipo entero.
- [4.2, 0.0, -33.76, 19.7e-2] es un arreglo de tamaño cuatro y de tipo real.
- ['a', 'z', '@'] es un arreglo de tamaño tres de tipo carácter.
- [5] es un arreglo de tamaño uno de tipo entero.
- [2.7, 0.6] es un arreglo de tamaño dos de reales.

Para referirse a cada elemento de un arreglo se utiliza su índice. El índice es asignado generalmente por la posición que ocupa el elemento en el arreglo. En este curso se asignarán los índices a los elementos de un arreglo partiendo de cero y llegando hasta el índice n-1, donde n es el tamaño del arreglo.

Ejemplos.

- El elemento uno (1) del arreglo [3, -6, 1, 24, 8] es –6.
- El elemento cero (0) del arreglo [4.2, 0.0, -33.76, 19.7e-2] es 4.2.
- El elemento dos (2) del arreglo ['a', 'z', '@'] es '@'.
- El elemento cero (0) del arreglo [5] es 5.
- El elemento uno (1) del arreglo [2.7, 0.6] es 0.6.

El índice puede ser obtenido mediante cualquier expresión numérica que de cómo resultado un número entero. Cuando se usa una expresión para obtener un índice se debe garantizar que el resultado de la expresión esté entre los límites permitidos de los índices para el arreglo.
Ejemplos. Si se tiene el arreglo [4.2, 0.0, -33.76, 19.7e-2], la variable i tiene el valor de uno (1) y la variable j tiene el valor de cuatro (4), entonces:

- El elemento (i+j/2) del arreglo es 19.7e-2. Al evaluar la expresión da como resultado 3.
- El elemento (j-3*i) del arreglo es 0.0. Al evaluar la expresión da como resultado 1.
- El elemento (j*i) del arreglo no existe pues al evaluar la expresión da como resultado 4 el cual no es un índice permitido (recuerde que los índices en este seudo-código van de cero hasta uno menos que el tamaño del arreglo; en este ejemplo, los índices van de 0 hasta 3).
- El elemento (2*i-j) del arreglo no existe. Al evaluar la expresión da como resultado –2.

- El elemento (j-2*i+1) del arreglo es 19.7e-2.

Como los espacios n-dimensionales en si mismos son conjuntos, es natural preguntarse si se pueden definir variables del tipo arreglo. La respuesta a esta pregunta es si. Para esto es importante determinar de manera precisa el tamaño del arreglo, el cual no se puede modificar después de ser definido, es decir, no se puede cambiar el tamaño de un arreglo con alguna instrucción.

La siguiente es la forma como se declaran variables del tipo arreglo en el seudo-código usado en este curso:

<nombre> : **arreglo**[n] de <tipo>

Donde, <nombre> es el nombre de la variable, <tipo> es el tipo de datos del arreglo y n es el tamaño del arreglo (debe ser un literal entero o una constante entera).

Ejemplos.

- Punto_2D : **arreglo**[2] de real
- Velocidades : **arreglo**[100] de real
- Texto : **arreglo**[MAX_TEXTO] de caracter
- Edades : **arreglo**[MAX_PERSONAS] de entero
- Banderas : **arreglo**[10] de booleano.

6.1.1 Forma de Accesar un Elemento de un Arreglo

Para accesar un elemento de un arreglo se utiliza un índice.
Para obtener o modificar uno de los datos almacenados en una variable del tipo arreglo se utiliza la siguiente notación de subíndice:

<nombre>[<indice>]

donde, <nombre> es el nombre de la variable de tipo arreglo e <indice> es el índice que referencia al elemento que se quiere acceder.

Ejemplos. Si x es un arreglo de tamaño cuatro (4) de reales, que tiene los siguientes elementos almacenados [4.2, 0.0, -33.76, 19.7e-2], si la variable i tiene el valor de uno (1) y la variable j tiene el valor de cuatro (4), entonces:

- x[i+j/2] retornaría el elemento tres del arreglo, es decir el valor 19.7e-2.
- x[j-3*i] retornaría el elemento uno del arreglo, es decir el valor 0.0.
- La asignación x[j-2*i+1] ←2.2222 modificaría el arreglo x y lo dejaría como [4.2, 0.0, -33.76, 2.2222].
- Si no se hace la asignación del ejemplo 3, la asignación x[0] ←10.0 dejaría el arreglo x como [10.0, 0.0, -33.76, 19.7e-2]. Si se hace la asignación del ejemplo anterior, el arreglo x quedaría como [10.0, 0.0, -33.76, 2.2222].
- x[2] retornaría el valor –33.76.

Cuando un arreglo es de enteros, un valor almacenado en el mismo se puede utilizar como índice de un arreglo o como parte de una expresión numérica.

Ejemplos. Si x es un arreglo de tamaño cuatro (4) de enteros, que tiene los siguientes elementos almacenados [4, 0, -33, 19], si la variable i tiene el valor de uno (1), la variable j tiene el valor de cuatro (4) y k es una variable del tipo entero, entonces:

- La asignación k ←x[i+j/2]+3 dejaría en la variable k el valor 22.
- x[x[i]+2*i] retornaría el elemento dos del arreglo, es decir el valor -33.
- La asignación x[x[0]-3*i] ←2 modificaría el arreglo x en la posición uno y lo dejaría como [4, 2, -33, 19].

6.1.2 Ejemplos Prácticos

Los arreglos son útiles cuando se requiere mantener y realizar operaciones sobre una colección determinada de datos de un mismo tipo, por ejemplo:

- Cuando se realizan estudios estadísticos, para calcular la moda, varianza, media, mediana, etc., de una colección de n datos, donde n es un valor conocido.
- En problemas de física vectorial, donde los vectores del espacio n-dimensional son representados como arreglos de tamaño n de tipo real.
- En aproximaciones de funciones por medio de polinomios de grado n, para realizar extrapolaciones e interpolaciones. En general, para realizar álgebra de polinomios.
- Para representar circuitos eléctricos, estructuras químicas, máquinas, etc.
- Para manipulación de mensajes que se presentan al usuario.
- Encriptamiento y seguridad de datos.

Ejemplo. Se requiere un programa que calcule el promedio de una colección de 20 reales que el usuario ingresará.

Aquí esta el pseudo código para este programa, se deja la especificación del mismo al lector.

```
Inicio
/* Las siguientes seis lineas de codigo leen el arreglo de reales */
para (i :=0 hasta 19) hacer
   escribir ("ingrese el dato:")
   escribir (i)
   escribir (cambio_linea)
   leer (A[i])
Fin_para
/* Las siguientes cuatro lineas de codigo calculan la suma de los
reales */
suma :=0
para (i :=0 hasta 19) hacer
   suma :=suma + A[i]
Fin_para
/* la siguiente linea calcula el promedio */
promedio :=suma / 20.0
/* las siguientes dos lineas imprimen el resultado */
escribir ("El promedio es:")
escribir (promedio)
Fin
```

6.1.3 Ordenamientos

En algunos casos el tipo de un arreglo tiene un orden total asociado a él. Por ejemplo, los números enteros están ordenados totalmente y los caracteres están ordenados totalmente por el ASCII asociado a cada uno. Un **ordenamiento** sobre un arreglo es una operación que dado un arreglo retorna el mismo arreglo con los elementos ordenados en forma ascendente o descendente.

- ***Ejemplos de ordenamientos de arreglos:***

Ejemplo (Tipo)	Arreglo	ascendentemente	descendentemente
1 (Entero)	[1, 4, 2, -3, 0, 8]	[-3, 0, 1, 2, 4, 8]	[8, 4, 2, 1, 0, -3]
2 (Caracter)	['3', 'A', 'a', ' ', '#']	[' ', '#', '3', 'A', 'a']	['a', 'A','3', '#', ' ']
3 (Real)	[3.1, 2.4, -12, 3.1, 0.5, 4.8]	[-12, 0.5, 2.4, 3.1, 3.1, 4.8]	[4.8, 3.1, 3.1, 2.4, 0.5, -12]

Para ordenar un arreglo se han desarrollado diversos algoritmos, entre los cuales están burbuja, selección y quick-sort. El siguiente es el algoritmo de ordenamiento ascendente por selección:

```
Inicio
para (i =0 hasta n-2) hacer
   para (j :=i+1 hasta n-1) hacer
      si (A[i] > A[j]) entonces
      /* Las siguientes 3 lineas de programa son muy famosas y  son conocidas
como el intercambio de variables o swapping */
         auxiliar :=A[i]
         A[i] :=A[j]
         A[j] :=auxiliar
      si_no
      fin_si
   fin_para
fin_para
Fin
```

6.1.4 Búsquedas

Una **operación de búsqueda** sobre un arreglo, es una operación que permite encontrar un elemento en un arreglo que cumple una cierta propiedad, por ejemplo el primer par, el elemento máximo, mínimo, la mediana, etc.

Una de las primeras operaciones de búsqueda sobre un arreglo que se puede desarrollar es encontrar el primer elemento del mismo que cumpla con una cierta propiedad, por ejemplo el primer par, el primer número primo, el primer número negativo, etc. De manera muy general (y poco eficiente), un algoritmo para este tipo de búsqueda tiene la forma:

```
Inicio
continuar :=verdadero
i :=0
mientras (i<n &continuar ) hacer
/*  aunque aquí aparece como una asignación simple esta línea de programa puede
involucrar varias operaciones más. Es lo que en la metodología de programación se
llama división.  */
   cumple_propiedad :=propiedad( A[i] )
 /* si el elemento cumple la propiedad, se deja de iterar el ciclo, el elemento buscado
es A[i].  si no se cumple, se prueba el siguiente elemento del arreglo */
   si cumple_propiedad entonces
      continuar :=falso
   si_no
      i :=i + 1
   fin_si
fin_mientras
/* si se encontro un elemento que cumple con la propiedad la variable continuar se
pone a falso y la variable contadora i indica cual es el elemento, si no se encuentra, la
variable continuar esta en verdadero */
si (continuar = falso) entonces
   escribir ("El elemento buscado es:")
   escribir (A[i])
si_no
```

```
   escribir (“No existe un elemento que cumpla con la propiedad”)
fin_si
Fin
```

Ejemplo. Desarrollar un algoritmo que permita encontrar el primer número par en un arreglo de número enteros.

```
Inicio
continuar :=verdadero
i :=0
mientras (i<n &continuar ) hacer
   cumple_propiedad :=(A[i] modulo 2 = 0)
   si (cumple_propiedad) entonces
      continuar :=falso
   sino
      i :=i + 1
   fin_si
fin_mientras
si i < n entonces
   escribir (“El elemento buscado es:”)
   escribir (A[i])
sino
   escribir (“No existe un elemento que cumpla con la propiedad”)
fin_si
Fin
```

Ejemplo. Desarrollar un algoritmo que permita encontrar el primer número primo en un arreglo de número enteros.

```
Inicio
continuar :=verdadero
i :=0
mientras (i<n &continuar ) hacer
 /*  las siguientes cinco lineas verifican si el elemento A[i] cumple la propiedad
de ser numero primo */
    k :=2
   mientras (k<A[i] &A[i] modulo k <>0 ) hacer
      K:=k + 1
   fin_mientras
   cumple_propiedad :=k = A[i]
   si (cumple_propiedad) entonces
      continuar :=falso
   sino
      i :=i + 1
   fin_si
fin_mientras
si (i < n) entonces
   escribir (“El elemento buscado es:”)
   escribir (A[i])
si_no
   escribir (“No existe un elemento que cumpla con la propiedad”)
fin_si
Fin
```

Este algoritmo se puede adaptar para encontrar no solo el primer elemento que cumple una propiedad sino el m-ésimo elemento que cumple con la propiedad.

Aunque toda búsqueda se puede realizar con el algoritmo anterior, este resulta poco eficiente en muchos casos, por ejemplo para calcular el máximo o mínimo de un arreglo. El anterior algoritmo se puede modificar de acuerdo al problema concreto a resolver.

Ejemplo. Desarrollar un algoritmo que calcule el máximo de un arreglo A de tamaño n de reales.

```
Inicio
x :=A[0]
i :=1
mientras (i<n ) hacer
   si (x < A[i]) entonces
      x :=A[i]
   sino
   fin_si
   i :=i + 1
fin_mientras
escribir (“El elemento buscado es:”)
escribir (A[i])
Fin
```

En algunos problemas, si los elementos del arreglo están ordenados las búsquedas se pueden optimizar.

Ejemplo. Desarrollar un algoritmo que permita encontrar el elemento mínimo en un arreglo que esta ordenado ascendentemente.

```
escribir (“El elemento buscado es:”)
escribir (A[0])
```

Ejemplo. Desarrollar un algoritmo que permita encontrar la mediana de un arreglo que esta ordenado ascendentemente.

```
escribir ("El elemento buscado es:")
escribir (A[n/2])
```

6.2 Cadenas de caracteres

Una **cadena de caracteres** o simplemente cadena, es un arreglo de caracteres, el cual siempre debe contar entre sus elementos a un caracter especial llamado fin de cadena, el cual determina la longitud y el texto de la cadena. En este curso se notará al carácter fin de cadena como ‘\0’.

Ejemplos.

- ['w', 'q', '9', '?', '\0', 'j'] es una cadena de caracteres de tamaño seis (6).
- [' ', '\0', '*', 'u', '\0', 'K', '='] es una cadena de caracteres de tamaño siete (7).
- ['\0', 'b', ')', '\0', ')', '\0'] es una cadena de caracteres de tamaño seis (6).
- ['#', '\0'] es una cadena de caracteres de tamaño dos (2).
- ['_', '0', '1', '3', '\0'] es una cadena de caracteres de tamaño cinco (5).

La cantidad de caracteres que aparecen antes del primer fin de cadena determina la longitud de una cadena. Es de aclarar que el tamaño y la longitud de una cadena de caracteres no son iguales, la relación que existe entre las dos es que la longitud es siempre menor que el tamaño de la cadena. Los caracteres que aparecen antes del fin de cadena determinan el texto de la misma. Generalmente el texto de una cadena es presentado entre comillas dobles (").

Ejemplos.

- ['w', 'q', '9', '?', '\0', 'j'] es una cadena de tamaño seis (6), longitud cuatro (4) y texto "wq9?".
- [' ', '\0', '*', 'u', '\0', 'K', '='] es una cadena de tamaño siete (7), longitud uno (1) y texto " ".
- ['\0', 'b', ')', '\0', ')', '\0'] es una cadena de tamaño seis (6), longitud cero (0) y texto ""(**la cadena vacía**).
- ['#', '\0'] es una cadena de tamaño dos (2), longitud uno (1) y texto "#".
- ['_', '0', '1', '3', '\0'] es una cadena de tamaño cinco (5), longitud cuatro (4) y texto "_013".

En muchas ocasiones solo se presenta el texto de la cadena, pues es la parte más importante de la cadena, además, a partir de este se puede inferir la longitud de la misma.

Una variable de tipo cadena de caracteres se puede declarar de una de las dos formas siguientes:

<variable> : **arreglo**[n] de **caracter**

donde, <variable>es el nombre de la variable de tipo cadena que se esta declarando y n es el tamaño de la cadena de caracteres.

Ejemplos.

- Texto : **arreglo**[20] de **caracter**
- Mensaje : **arreglo**[50] de **caracter**
- Saludo : **arreglo**[70] de **carácter**

Como las cadenas de carácter son uno de los datos más utilizados en programación, en muchos lenguajes se han implementado funciones para las operaciones más comunes sobre cadenas. La siguiente es una lista corta de estas operaciones:

Operación	Descripción
longitudCadena(<cadena1>)	retorna la longitud de la cadena

copiarCadena(<cadena1>, <cadena2>)	crea una copia de la cadena <cadena2> en la cadena <cadena1>
concatenarCadena(<cadena1>, <cadena2>)	Retorna la concatenación de <cadena1> con <cadena2> en <cadena1>.
compararCadenas(<cadena1>, <cadena2>)	retorna menos uno (-1) si <cadena1> es menor que <cadena2>, cero (0) si son iguales y uno (1) en otro caso.*

Para la comparación de la cadena <cadena1> con la cadena <cadena2> usa el orden lexicográfico, esto es, primero compara el primer elemento del texto de cada cadena usando el orden asociado a los caracteres. Si el primer elemento de <cadena1> es menor que el primer elemento de <cadena2>, entonces <cadena1> es menor que <cadena2>, si es mayor, entonces <cadena1> es mayor que <cadena2>. Pero si son iguales, se revisan los siguientes caracteres del texto de cada cadena y se hace el mismo proceso que para el primer carácter. Se repite el mismo proceso si los caracteres son iguales hasta que alguno de los textos de las cadenas se termine. En tal caso, el que primero se termine se considera menor, pero si los dos terminan igual se consideran iguales. El siguiente es el algoritmo de comparación usando el orden lexicográfico.

```
Inicio
i := 0
mientras(cadena1[i] <> '\0' &cadena2[i] <> '\0' &cadena1[i] <> cadena2[i])
hacer
   i := i + 1
fin_mientras
si ( cadena1[i] = '\0' ) entonces
   si ( cadena2[i] = '\0' ) entonces
      x := 0
   sino
      x := -1
   fin_si
sio
   si ( cadena2[i] = '\0' ) entonces
      x := 1
   sino
      si ( cadena1[i] < cadena2[i] ) entonces
         x := -1
      sino
         x := 1
      fin_si
   fin_si
fin_si
Fin
```

Ejemplos. Sean cad1 =“Texto uno (1)”, cad2 =“Otro texto 2-”, cad3 =“Adios !...” y cad4 = “Texto dos (2)” cadenas de caracter.

1 concatenarCadena(cad1, cad2) deja a cad1 como “Texto uno

(1)Otro texto 2-".
2 concatenarCadena (cad2, cad1) deja a cad2 como "Otro texto 2-Texto uno (1)".
3 copiarCadena(cad2, cad1) deja a cad2 como "Texto uno (1)".
4 compararCadenas(cad4, cad1) retorna uno (1) pues cad4 es mayor según el orden lexicográfico que cad1.
5 compararCadenas(cad1, "Texto uno (1)") retorna cero (0) pues el texto de cad1 es igual al texto enviado.
6 copiarCadena(cad2, "texto uno (1)") deja a cad2como "texto uno (1)".
7 concatenarCadena(cad1, "Hola mundo..") deja a cad1 como "Texto uno (1) Hola mundo..".

6.3 MATRICES

Pusto que el espacio m-dimensional de un conjunto A, notado A^m, es a su vez un conjunto, es posible definir el espacio n-dimensional de A^m, es decir, se puede tener un arreglo de tamaño n de arreglos de tamaño m de un conjunto A.

Ejemplos.

1 [[1, 2], [0, 3], [8, -3]] es un arreglo de tamaño tres (3) de arreglos de tamaño dos (2) de enteros.
2 [[0.3, 1.2, -1.2], [0.0, -2.7, 5.1], [9.4, 3.2, 1e-5]] es un arreglo de tamaño tres (3) de arreglos de tamaño tres (3) de reales.
3 [['r', '$', 'q'], ['p', '@', 'i']] es un arreglo de tamaño dos (2) de arreglos de tamaño tres (3) de caracteres.

Una **matriz** es un arreglo de tamaño n de arreglos de tamaño mde un tipo dado. Al tamaño nse le conoce como **número de filas** y al tamaño m como **número de columnas**. Estos nombres son debidos a la presentación de una matriz como una tabla, donde la fila i-esima de la tabla corresponde con el arreglo i-esimo de la matriz.

Ejemplos.

1 El arreglo [[1, 2], [0, 3], [8, -3]] se presenta en forma de tabla como sigue:

1	2
0	3
8	-3

2 El arreglo [[‘r’, ‘$’, ‘q’], [‘p’, ‘@’, ‘i’]] se presenta en forma tabular como sigue

‘r’	‘$’	‘q’
‘p’	‘@’	‘i’

3 El arreglo [[0.3, 1.2, -1.2], [0.0, -2.7, 5.1], [9.4, 3.2, 1e-5]] se presenta en forma de tabla así:

0.3	1.2	-1.2
0.0	-2.7	5.1
9.4	3.2	1e-5

En el pseudo código empleado en este curso existen dos formas diferentes de declarar una matriz, como un arreglo de arreglos o como una matriz:

<nombre> : arreglo [n] de arreglo [m] de <tipo>
<nombre> : arreglo [n][m] de <tipo>

donde, <nombre>:es el nombre de la variable que se esta definiendo. <tipo>: es el tipo de elementos almacenados en la matriz. n: es el número de filas de la matriz (debe ser una constante o un literal entero). m: es el número de columnas de la matriz (debe ser una constante o un literal entero).

Ejemplos.

1. M: arreglo[4][5] de entero, define una variable con nombre M de tipo entero, de cuatro (4) filas y cinco (5) columnas.
2. Tabla: arreglo [3][2] de caracter, define una variable con nombre Tabla de tipo caracter, de tres filas y dos columnas.
3. Cuadro: arreglo[4] de arreglo[3] de real, define una matriz con nombre Cuadro de tipo real, de cuatro filas por tres columnas.
4. Transición: arreglo [10][10] de caracter, define una variable de tipo caracter, de diez filas por diez columnas.

Para acceder cada uno de los elementos de la matriz se utiliza el doble subíndice, donde el primer subíndice indica la fila del elemento y el segundo indica la columna. Al igual que en arreglos el índice para fila varía entre cero (0) y el número de filas menos uno (n-1), y el índice para columna varía entre cero (0) y el número de columnas menos uno (m-1).

La notación en el pseudo código utilizado en este curso es la siguiente:

<nombre>[<fila>][<columna>]

donde, <nombre>: es el nombre de la variable de tipo matriz. <fila>: es una expresión numérica (entera) que indica la fila del elemento a acceder. <columna>: es una expresión numérica (entera) que indica la columna del elemento a acceder.

Ejemplos. Sean i y j variables enteras con valores 1 y 2 respectivamente y Tabla la siguiente matriz de enteros,

3	-2	4	5
2	4	-8	0
0	10	13	1

1. Tabla[2][1], retorna el elemento que está en la fila dos y columna uno, es decir 10 (recuerde que se indexa desde cero).
2. Tabla[0][0], retorna el elemento que está en la fila cero y columna cero, es decir 3.
3. Tabla[0][3], retorna el elemento que está en la fila cero y columna tres, es decir 5.
4. Tabla[2][0], retorna el elemento que está en la fila dos y columna cero, es decir 0.
5. Tabla[1][2], retorna el elemento que está en la fila uno y columna dos, es decir -8.
6. Tabla[j-i][j+i], retorna el elemento que está en la fila uno y columna tres, es decir 0.
7. Tabla[Tabla[j-i][j+i]][j-1], retorna el elemento que está en la fila cero y columna uno, es decir -2.
8. La asignación Tabla[j][j+i] ←20, cambia el elemento que está en la fila dos y columna tres por el valor 20, es decir, Tabla queda como:

3	-2	4	5
2	4	-8	0
0	10	13	20

9. Sin tener en cuenta la asignación del ejemplo ocho, la asignación: Tabla[j-2][j*i] ←Tabla[j-i][j+i] + 3 cambia el elemento que está en la fila cero y columna dos por el valor 3, es decir, Tabla queda como:

3	-2	3	5
2	4	-8	0
0	10	13	1

Se puede apreciar en los ejemplos siete y nueve que cuando una matriz es de enteros, un valor almacenado en la matriz se puede utilizar como índice o como parte de una expresión numérica.

Las matrices son útiles cuando se tiene información en forma tabular que debe ser mantenida y procesada. Ejemplos de información que se presenta en forma tabular son: los sistemas de ecuaciones lineales (donde la información proporcionada son los coeficientes de cada ecuación lineal), problemas de optimización lineal, cuadros estadísticos, entre otros.

Ejemplo. Una empresa tiene una tabla con las unidades vendidas por cada mes de un año, de los cuatro productos que vende. La tabla tiene la siguiente forma:

Producto \ Mes	1	2	3	4	5	6	7	8	9	10	11	12
Producto 1												
Producto 2												
Producto 3												
Producto 4												

donde la fila indica el producto y la columna el mes del año. El elemento en la fila i y columna j indica el número de unidades vendidas del producto i+1, en el mes j+1.

Realizar un programa que lea las unidades vendidas de cada producto por mes en un año y calcule la cantidad de unidades vendidas de cada producto en un año, la cantidad de unidades vendidas de los cuatro productos por cada mes, el producto con menos ventas en un año y el mes en que más se vendió cada producto.

DIALOGO:

Objetos conocidos: Una tabla de números de cuatro (4) filas por doce (12) columnas que indica la cantidad de unidades vendidas de un producto por cada mes de un año.

Objetos desconocidos: Una lista de números de tamaño cuatro (4) con la cantidad de unidades vendidas de cada producto, otra lista de números de tamaño doce (12) con las unidades vendidas en cada mes, un número que indica el producto más vendido y una lista de números de tamaño cuatro (4) que indica en que mes se vendió más cada producto.

Condiciones: La cantidad de unidades vendidas en un año de un producto es la suma de las unidades vendidas del producto en cada mes. La cantidad de unidades vendidas en un mes es la suma de las unidades vendidas de cada producto en ese mes. El producto que más se vendió es el producto del que se hayan vendido más unidades en un año. El mes en que más se vendió un producto es el mes en el que más unidades del producto se hayan vendido. El elemento en la fila i y columna j indica el número de unidades vendidas del producto i+1, en el mes j+1.

ESPECIFICACIÓN: Se deja al lector.

ALGORITMO:

```
Inicio
/* Las variables usadas en el algoritmo*/
i : entero
j : entero
min : entero /* indice del producto menos vendido */
M : arreglo[4][12] de entero  /* la tabla de ventas */
T : arreglo[4] de entero /* los totales por producto */
V : arreglo[12] de entero /* los totales por mes */
Max : arreglo[4] de entero /* meses de mas ventas por producto */
/* las siguientes diez lineas leen la tabla de ventas */
para(i :=0 hasta 11) hacer
 para( j :=0 hasta 3) hacer
  escribir ("Ingrese las ventas del producto ")
  escribir (j+1)
  escribir (" Para el mes ")
  escribir (i+1)
  escribir (cambio_linea)
  leer (M[j][i])
 fin_para
fin_para
/*las siguientes seis lineas calculan los totales de ventas por producto */
para (i :=0 hasta 3) hacer
T[i] :=0
 para (j :=0 hasta 11) hacer
  T[i] :=T[i] + M[i][j]
fin_para
fin_para
/*  las siguientes seis lineas calculan los totales de ventas por mes */
para ( i :=0 hasta 11) hacer
 V[i] :=0
 para (j :=0 hasta 3) hacer
  V[i] :=V[i] + M[j][i]
fin_para
fin_para
/*  las siguientes catorce lineas (incluyendo comentarios) calculan el mes de mas
ventas por producto */
para (i :=0 hasta 3) hacer
 /* se supone que el de mas ventas fue el primero */
 Max[i] :=0
 /* se compara con los demas y si hay uno mejor se cambia */
 para ( j :=1 hasta 11) hacer
  /*  se compara el mes j-esimo con el que se lleva como mejorsi el mes j-esimo es
mejor se cambia por j el que se lleva  */
  si (M[i][j] < M[i][Max[i]]) entonces
   Max[i] :=j
  fin_si
 fin_para
fin_para
/*  las siguientes catorce lineas (incluyendo comentarios) calculan el producto
menos vendido en el año */
/* se supone que el menos vendido es el producto cero */
min :=0
para ( i :=1 hasta 3) hacer
 /* si hay uno menor se cambia */
 si (T[i] < T[min]) entonces
  min :=i
```

```
 fin_si
fin_para
/* se imprimen los resultados */
/* ventas totales por producto */
para (i := 0 hasta 3) hacer
 escribir ("Unidades vendidas del producto ")
 escribir (i+1)
 escribir (T[i])
 escribir (cambioLinea)
fin_para
/* ventas totales por mes */
para(i := 0 hasta 11) hacer
 escribir ("Unidades vendidas en el mes ")
 escribir (i+1)
 escribir (V[i])
 escribir (cambioLinea)
fin_para
/* mes en el que mas se vendio cada producto */
para(i := 0 hasta 3)hacer
 escribir ("El mes en el que mas se vendio el producto ")
 escribir (i+1)
 escribir (" fue el mes ")
 escribir (Max[i])
 escribir (cambioLinea)
fin_para
/* producto menos vendido */
escribir ("El producto menos vendido fue ")
escribir (min +1)
Fin
```