14. ¿Qué técnicas utilizaría para almacenar caracteres en los compiladores de que usted dispone?
15. Encuentre tablas que den los códigos de los caracteres en EBCDIC y ASCII.
16. Dé la representación de su nombre y dirección con el código EBCDIC.
17. Dé la representación de su nombre y dirección con el código ASCII.
18. Represente su número de identificación del seguro social o algún otro número, en códigos EBCDIC, ASCII y decimal empacado.
19. Dé al menos tres posibles formas de almacenamiento de las siguientes tres cadenas: 'MAGPIE', 'PIE' y 'MAGENTA'.
20. ¿Cuáles son las ventajas y desventajas básicas que deben considerarse cuando se almacenan cadenas traslapadas?
21. ¿Cuál son las ventajas y desventajas que deben considerarse cuando se almacenan cadenas empacadas.
22. ¿Cuáles son las ventajas y desventajas que deben considerarse al usar: a) un indicador de longitud para identificar el final de la cadena; b) un apuntador para identificar el final de la cadena; y c) una marca de fin de cadena para identificar el final de la cadena?
23. Explique en qué difiere un lenguaje de programación orientado a objetos de un lenguaje de programación convencional en cuanto a las operaciones sobre las estructuras de datos.
24. ¿Cuál es el conjunto de caracteres, más completo, que se usa en el compilador de su lenguaje de programación, ya sea Pascal o algún otro (COBOL, FORTRAN, PL/1, . . .)?
25. Investigue cómo debe ser el formato de entrada de cadenas para su compilador Pascal.
26. Escriba un programa para leer y escribir cinco cadenas de caracteres, cada una de longitud de 10.
27. Modifique su programa para leer y escribir *N* cadenas de caracteres, cada una de longitud indeterminada. ¿Cuáles son los requisitos de entrada para su programa?
28. Escriba un programa que utilice los operadores de longitud, concatenación y subcadena.
29. Agregue los operadores de inserción y supresión en el programa del ejercicio 28.

capítulo dos

arreglos

En este capítulo se discuten las estructuras de datos conocidas como arreglos, los cuales son bloques básicos para la construcción de estructuras de datos más complejas. Casi todas las estructuras complejas se pueden representar indirectamente mediante arreglos. Primero definiremos los arreglos simples y multidimensionales y, después veremos las facilidades para declarar arreglos en COBOL y Pascal. Gran parte del capítulo será dedicado al tema de cómo almacenar arreglos uni y multidimensionales. Finalmente, se concluye el capítulo con un análisis de las técnicas para representar arreglos dispersos y triangulares. Encontraremos que los arreglos son muy utilizados en la estructuración de datos en la memoria de trabajo de un programa, así como en la estructuración de datos en archivos más permanentes y en bases de datos.

ARREGLOS UNIDIMENSIONALES

Un *arreglo* es un conjunto finito ordenado de elementos homogéneos. La propiedad de ordenación significa que es posible identificar el primero, segundo, tercero, . . . , y el *enésimo* elemento del arreglo. Los elementos del arreglo son homogéneos porque todos son del mismo tipo de dato. Un arreglo puede ser un conjunto de elementos de tipo cadena en tanto que otro puede ser de tipo entero. Los elementos de un arreglo pueden ser, a su vez, otro arreglo. A los arreglos por lo general se les llama *tablas*.

La forma más simple de un arreglo es el *arreglo unidimensional,* conocido como *vector*. Un arreglo unidimensional llamado VICTOR, el cual consiste de *N* elementos, se puede representar como en la figura 2-1.

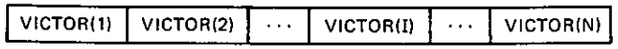

Figura 2-1 Ejemplo de un arreglo unidimensional.

Subíndices

Los *subíndices o índices* de un elemento designan su posición en el arreglo; en la notación que se emplea aquí, un elemento en particular se define con el nombre del arreglo, seguido por el subíndice del elemento entre paréntesis, es decir, VICTOR(I). Otras posibilidades para designar al elemento del arreglo VICTOR con subíndice I, son: VICTOR[I], VICTOR_I, VICTOR^I.

Nótese que sólo el arreglo en su totalidad tiene un nombre. Para hacer referencia a los elementos de un arreglo es necesario utilizar un subíndice, es decir, su posición relativa en el arreglo. Esta convención permite al programador escribir algoritmos que mediante cifras recorran los elementos de un arreglo, en lugar de usar un nombre distinto para cada elemento.

Definición

De manera más formal, el arreglo unidimensional A, con elementos del tipo de estructura de datos T, subindizados desde L hasta U, es

$A(L{:}U) = \{A(I)\}$
para $I = L, L+1, \ldots, U-1, U$
donde cada elemento $A(I)$ es una estructura de datos de tipo T.

La notación $A(L{:}U)$ indica que los subíndices de A están comprendidos en un rango de L a U.

El número de elementos en un arreglo se llama *rango*. El rango de un arreglo $A(L{:}U)$ es $U - L + 1$, en tanto que el del arreglo $B(1{:}N)$ es N.

Ejemplos

Como ejemplo, un arreglo unidimensional contiene las mediciones de temperatura de una ciudad, registrada cada hora, durante un periodo de 24 horas. Estas mediciones se ordenan por su hora de observación correspondiente. Todos los elementos son de un mismo tipo, pues cada uno corresponde a una medición de temperatura.

El valor mínimo permisible para el subíndice del arreglo se llama *límite inferior*; el valor máximo permisible se llama *límite superior*. En nuestra definición formal de un arreglo, L es el límite inferior y U es el límite superior por el rango del subíndice de A. En el ejemplo mediciones de temperatura, un límite inferior natural es 1 y el superior es 24. Al arreglo lo podemos llamar TEMPERATURA, y en TEMPERATURA(I) registrar las mediciones para la I-ésima hora, donde $1 \leq I \leq 24$.

Otro ejemplo de arreglo unidimensional es una tabla de promedio de ingresos estatales, si llamamos al arreglo como PROMEDIO-INGRESOS. El elemento PROMEDIO-INGRESOS(I) contiene el promedio de ingresos del I-ésimo del estado, donde los estados son ordenados secuencialmente. Así, PROMEDIO-INGRESOS(1) es el ingreso promedio del estado de Aguascalientes y PROMEDIO-INGRESOS(32) corresponde al ingreso promedio del estado de Zacatecas.

En algunas aplicaciones es más natural usar un límite inferior que no sea igual a 1. Por ejemplo, puede ser más conveniente almacenar las temperaturas de un experimento que se registran cada 10 segundos desde 0, en un arreglo con límite inferior igual a 0. Aún más, el límite inferior puede ser negativo. Este es el caso más frecuente en arreglos que almacenan datos que se exhiben gráficamente, tal como en un eje de puntos en el renglón de -100 a +100.

ARREGLOS MULTIDIMENSIONALES

Un arreglo de dos dimensiones, es un arreglo en el cual cada elemento es otro arreglo. Un arreglo llamado B, el cual consiste de M elementos, cada uno de los cuales es un arreglo de N elementos se puede representar como una tabla de M por N, como se muestra en la figura 2-2.

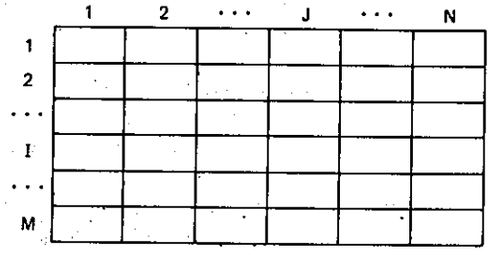

Figura 2-2 Ejemplo de un arreglo de M por N.

Subíndices

Es necesario especificar dos subíndices para identificar un elemento individual en un arreglo bidimensional. Por convención, el primer subíndice hace referencia al *renglón*

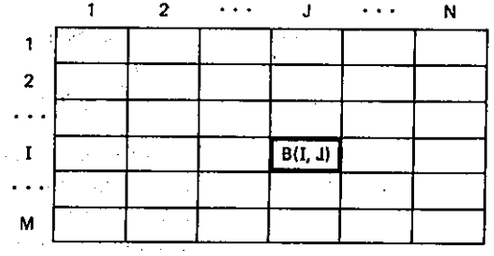

Figura 2-3 Elemento B(I,J) en un arreglo bidimensional.

y el segundo subíndice se refiere a la *columna*. Esto es $B(I,J)$ es elemento de B que está en el I-ésimo renglón y J-ésima columna del arreglo B, tal como se muestra en la figura 2-3.

Definiciones

Más formalmente, el arreglo bidimensional B con elementos individuales de tipo T tienen subíndices para el renglón que se extiende de 1 a M, y subíndices para la columna, desde 1 hasta N es

$B(1{:}M, 1{:}N) = \{B(I,J)\}$
para $I = 1, \ldots, M$ y $J = 1, \ldots, N$
donde cada $B(I,J)$ es una estructura de datos de tipo T.

Se dice que el arreglo B es de dimensiones de M por N. Existen N elementos en cada renglón, y M elementos en cada columna de B; así, el total de elementos en el arreglo B es de $M*N$.

En general, un arreglo bidimensional B, donde su primer subíndice tiene límite inferior L_1, y límite superior U_1, y su segundo subíndice tiene límites L_2 y U_2, inferior y superior, respectivamente, se definen como

$B(L_1{:}U_1, L_2{:}U_2) = \{B(I,J)\}$ para $L_1 \le I \le U_1$
y $L_2 \le J \le U_2$
donde cada $B(I,J)$ es una estructura de datos de tipo T.

El número de elementos en un renglón de B es $U_2 - L_2 + 1$ y el número de elementos de una columna de B es $U_1 - L_1 + 1$. Así, el total de elementos en el arreglo B es $(U_2 - L_2 + 1)*(U_1 - L_1 + 1)$.

Ejemplos

Los arreglos bidimensionales son muy comunes. Por ejemplo, la Cámara de Comercio de la ciudad puede registrar el número de horas con cielo despejado durante los 365 días del año. Estos días pueden ser almacenados en un arreglo con dimensiones de 365 por 24. Llamemos al arreglo W, los valores de W se pueden determinar por

$W(I,J)$ = 1 si está despejado durante la J-ésima hora del I-ésimo día del año.
0 en cualquier otro caso, es decir, si neva, llueve, llovizna, etc.,
durante la J-ésima hora en el I-ésimo día del año.

Todos los elementos del arreglo son de la misma clase; cada uno es un booleano.

Otro ejemplo de arreglos bidimensionales es aquel que contiene datos sobre las calificaciones de los exámenes de estudiantes de cuarto semestre. Este arreglo llamado BOLETA, puede tener dimensiones de M por 4, donde M es el número de estudiantes. Entonces BOLETA (I,J) es la calificación obtenida por el I-ésimo estudiante en su

J-ésimo examen. El valor de I puede estar entre 1 y M inclusive; J puede estar entre 1 y 4 inclusive. Cada elemento del arreglo es una calificación; todos los elementos pueden ser enteros o de punto flotante con dos decimales después del punto, o bien pueden ser caracteres, todo depende de la convención adoptada.

Sección transversal

Una *sección transversal* de un arreglo bidimensional se obtiene al mantener a uno de sus subíndices constantes mientras se hace variar a otro en el rango de valores para este último. La notación frecuentemente usada para denotar una sección transversal es un asterisco (*) para el subíndice que puede tomar cualquier valor del rango definido. Por ejemplo,

$B(*,4)$

se refiere a la cuarta columna del arreglo B presentado al inicio de la sección, es decir:

$B(*,4) = \{B(1,4), B(2,4), B(3,4), \ldots, B(M,4)\}$

De manera semejante,

$B(I,*)$

es el I-ésimo renglón del arreglo de B;

$W(47,*)$

es el renglón de W que contiene los valores registrados en el 47o. día del año;

BOLETA(*,3)

es la columna de BOLETA, que contiene las calificaciones de los exámenes de todos los estudiantes del tercer semestre. Una sección transversal de un arreglo bidimensional es esencialmente una rebanada tomada del arreglo.

Transpuesta

La *transpuesta* de un arreglo bidimensional se obtiene invirtiendo las posiciones del subíndice. La transpuesta de un arreglo de M por N es un arreglo de N por M. La transpuesta de un arreglo B es comúnmente denotada por B^T. La transpuesta de un arreglo B se muestra en la figura 2-4.

Por definición,

$B(I,J) = B^T(J,I)$.

elementos del arreglo. Considere primero la declaración de un arreglo unidimensional llamado TEMPERATURA, con subíndices en el rango de 1 a 24. Suponga que cada elemento del arreglo es un entero, con rango de 0 a 99.

En COBOL, para referirse a la *I*-ésima temperatura como TEMPERATURA(*I*), el programador tiene que introducir un nombre de agrupación para los elementos del arreglo. Por lo que resulta conveniente agregarle el nombre TABLA al nombre del arreglo de aplicación. Así, la DIVISION DE DATOS del programa debe incluir:

```
01 TABLA-TEMPERATURA.
   02 TEMPERATURA    OCCURS 24 TIMES
                     PICTURE 99
```

En Pascal no es necesaria la variable de agrupamiento artificial. Los arreglos se declaran con el enunciado var:

```
var temperatura: array [1..24] of integer.
```

Pascal permite límites inferiores para el subíndice, menores a 1, por ejemplo:

```
var puntos-gráfica: array [-100 .. 100] of integer.
```

En COBOL, los arreglos sólo permiten subíndices mayores a 0. La cláusula OCCURS indica el límite superior del subíndice.

Para referirse al *I*-ésimo elemento del arreglo de arriba, COBOL usa la notación: TEMPERATURA(*I*), en tanto que Pascal usa corchetes en lugar de paréntesis: Temperatura[*i*].

Arreglos bidimensionales

La declaración de un arreglo que contenga las puntuaciones de 40 estudiantes en 4 exámenes puede ser como sigue:

En COBOL se introducen dos nombres de agrupamiento:

```
01 TABLA-CALIFICACION.
   02 ESTUDIANTES OCCURS 40 TIMES.
      03 CALIFICACION OCCURS 4 TIMES
                      PICTURE 99V9.
```

En Pascal,

```
var calificación: array [1..40, 1..4] of real.
```

En COBOL, la calificación para el *I*-ésimo estudiante en el *J*-ésimo examen es CALIFICACION (*I,J*). Pascal utiliza corchetes en lugar de paréntesis: calificación [*i,j*]. Los subíndices deben estar dentro del rango válido.

Más dimensiones

El número máximo de dimensiones que se pueden declarar en COBOL es de tres. Por ejemplo:

```
01 TABLA COLEGIO.
   02 CLASE OCCURS 6 TIMES.
      03 SEXO OCCURS 2 TIMES.
         04 COLEGIO OCCURS 10 TIMES
                    PICTURE 9(5).
```

Si intentamos declarar más de 3 dimensiones, el compilador lo detecta.

En Pascal los arreglos tridimensionales se pueden definir por:

```
var colegio: array [1..6,1..2,1..10] of integer.
```

En algunos lenguajes, como FORTRAN y COBOL se asigna espacio de memoria para los arreglos en el momento de la compilación. Así los rangos deben especificarse explícitamente cuando se define un arreglo. En otros lenguajes, como PL/I, el espacio puede asignarse en forma dinámica en el momento de la ejecución.

Operaciones sobre los elementos del arreglo

COBOL y Pascal (y otros lenguajes que permiten declarar arreglos) tienen facilidades para la manipulación de elementos individuales de un arreglo. El conjunto de operaciones legales se define por las estructuras de datos de esos elementos. Por ejemplo, las operaciones de subcadena y concatenación se pueden ejecutar sobre los elementos de un arreglo de tipo cadena. Los cálculos aritméticos pueden ejecutarse sobre elementos individuales de un arreglo de enteros o de punto flotante. Por ejemplo, en COBOL:

```
COMPUTE PAGO-TOTAL (I) = PAGO-POR-HORA (I) * HORAS TRABAJADAS (I)
```

donde cada una de las tres variables es un elemento del arreglo.

Operaciones sobre arreglos

Algunos lenguajes de programación también tienen operaciones sobre arreglos. Por ejemplo, si *A* fue declarada como un arreglo en PL/I, entonces *A* = *A* + 2 sumaría 2 a cada uno de los elementos del arreglo. Si *A* y *B* fueron declarados como arreglos con las mismas dimensiones, entonces en PL/I, *A* = *A* * *B* multiplicaría cada elemento de *A* por su correspondiente elemento de *B* (es decir, por el elemento con el mismo subíndice), guardando el resultado en *A*. Este producto de arreglos *no* es la multiplicación matricial (como la que se define en álgebra).

En PL/I, las operaciones pueden ejecutarse sobre secciones transversales del arreglo.

Por ejemplo,

```
CALIFICACION(20,*) = 0;
```

inicializa cada elemento del arreglo CALIFICACION en el renglón 20 con valor cero.

```
VECTOR(*) = ARREGLO1(I,*) * ARREGLO2(*,J);
```

multiplica los elementos del *I*-ésimo renglón del ARREGLO1 por los elementos de la *J*-ésima columna del ARREGLO2. Los arreglos se deben declarar de tal manera que exista compatibilidad en el rango de valores para los subíndices. Por ejemplo, si el segundo subíndice del ARREGLO1 tiene un rango de 0 a 25, entonces el primer subíndice del ARREGLO2 también debe estar en un rango de 0 a 25, lo mismo que el subíndice de VECTOR. La multiplicación de arriba tiene los mismos efectos que el siguiente ciclo:

```
DO K = 0 TO 25;
   VECTOR(K) = ARREGLO1(I,K) * ARREGLO2(K,J);
END;
```

FORMAS DE ALMACENAMIENTO: ARREGLOS UNIDIMENSIONALES

Como en las otras estructuras de datos, hay varias formas de representar arreglos en memoria. Estos esquemas de representación se pueden evaluar con base en cuatro características: 1) La simplicidad de acceso a los elementos, 2) facilidad de recorrer a través de varios caminos, 3) eficiencia de almacenamiento, y 4) facilidad de crecer. Sin embargo, no es posible optimizar simultáneamente estas cuatro características.

Límite inferior: uno

Inicialmente consideremos la forma de almacenar arreglos unidimensionales y nombremos un arreglo como EMP-NO con límite inferior igual a uno y límite superior igual a *N*. Una forma de almacenar este arreglo es que el orden físico de los elementos sea el mismo que el orden lógico de los elementos. El almacenamiento para el elemento EMP-NO(I + 1) será adyacente al almacenamiento del elemento EMP-NO(I), para $I = 1, \ldots, N-1$.

Para calcular la dirección de inicio (es decir, la localidad) del elemento EMP-NO (I), es necesario conocer:

1. La dirección de inicio del espacio reservado para almacenar el arreglo.
2. El tamaño de cada elemento en el arreglo.

Denotamos a B como la dirección de inicio del arreglo, también conocida como *localidad base*. Supongamos que cada elemento del arreglo ocupa S bytes. Entonces la localidad del *I*-ésimo elemento del arreglo es:

$$B + (I - 1) * S \qquad \text{(ecuación 2-1)}$$

ya que $I - 1$ elementos, cada uno de tamaño S, físicamente preceden al *I*-ésimo elemento. El compilador necesita ser capaz de determinar las localidades de los elementos; dichas localidades también son útiles para un programador que efectúe un vaciado de memoria para depuración.

Generalización del límite inferior

Ahora, extendamos la ecuación 2-1 para encontrar el *I*-ésimo elemento de un arreglo, el cual tiene subíndices con límite inferior diferente de uno. Es decir, considere el arreglo declarado como *Z1*(4 : 10). Tomando un caso específico, la dirección de inicio de *Z1*(6) es:

$$B + (6 - 4) * S$$

ya que 6 - 4 (es decir, 2) elementos preceden a *Z1*(6). Para un arreglo declarado *Z2* (- 2:2), la localidad de *Z2*(1) es

$$B + (1 - (- 2)) * S$$

ya que (1 - (- 2)) (es decir, 3), elementos (*Z2*(- 2), *Z2*(- 1), y *Z2*(0)) preceden a *Z2*(1).

En general, el elemento ARREGLO(I) del arreglo definido como ARREGLO(L:U) está en la localidad

$$B + (I - L) * S \qquad \text{(ecuación 2-2)}$$

Esta fórmula es válida para cualquier límite inferior L ya sea positivo, negativo o cero.

FORMAS DE ALMACENAMIENTO: ARREGLOS MULTIDIMENSIONALES

Orden por renglón

Debido a que la memoria de la computadora es lineal, se debe linealizar un arreglo multidimensional para su almacenamiento. Una alternativa para la linealización es almacenar primero el primer renglón del arreglo, después el segundo renglón del arreglo, después el tercero, y así sucesivamente. Por ejemplo, el arreglo definido por *RAZON* (1:4,1:6), el cual lógicamente aparece como se muestra en la figura 2-6, físicamente aparece en *orden por renglón* como en la figura 2-7.

Este es el esquema de almacenamiento usado para arreglos declarados en la mayoría de las implantaciones de COBOL, Pascal, C y PL/1.

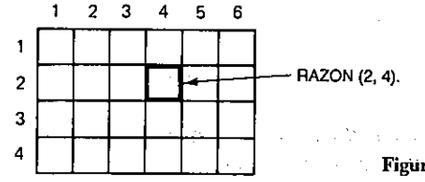

Figura 2-6 Ejemplo de un arreglo bidimensional.

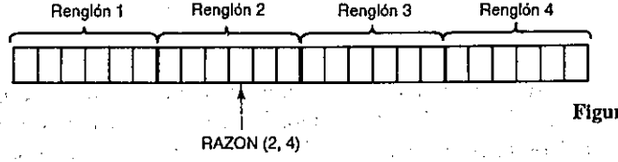

Figura 2-7 Arreglo de la figura 2-6, linealizado por renglón mayor.

Supongamos que B es la dirección base del arreglo y que cada elemento del arreglo es de tamaño S, la dirección inicial del elemento *RAZON(I,J)* es

$$B + (I - 1)*6*S + (J - 1)*S$$

debido que existen $I - 1$ renglones, cada uno de longitud $6 * S$, los cuales preceden al renglón en el que está el elemento RAZON(I,J) y existen $J - 1$ elementos, cada uno de longitud S, los cuales preceden al elemento RAZON(I,J) en el renglón I. El elemento RAZON(2,4) se localiza en $B + 9*S$.

Generalizando, el elemento ARREGLO(I,J) del arreglo definido por ARREGLO(L_1:U_1,L_2:U_2) está en la localidad:

$$B + (I - L_1)*(U_2 - L_2 + 1)*S + (J - L_2)*S \qquad \text{(ecuación 2-3)}$$

ya que, hay $I - L_1$ renglones, cada uno de longitud $(U_2 - L_2 + 1)*S$, los cuales preceden al renglón en el que está el elemento ARREGLO(I,J), y hay $J - L_2$ elementos, cada uno de longitud S, los cuales preceden al elemento ARREGLO(I,J) en el renglón I.

Ilustrando esto, el arreglo definido por Z(- 2:2, 4:6) puede representarse lógicamente como se muestra en la figura 2-8.

Este arreglo aparece físicamente ordenado por renglón como se muestra en la figura 2-9. Hay dos renglones (renglón -2 y renglón -1), cada uno de longitud $3 * S$, que preceden al renglón 0. En el renglón 0, hay 2 (es decir, 6 - 4) elementos, cada uno de longitud S, que preceden al elemento $Z(0,6)$. Esto es, la localidad de inicio de $Z(0,6)$ es

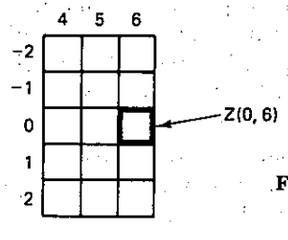

Figura 2-8 Ejemplo de un arreglo bidimensional.

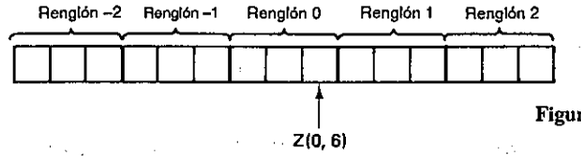

Figura 2-9 Arreglo de la figura 2-8, linealizado por renglón

$$B + (0 - (-2))*(6 - 4 + 1)*S + (6 - 4)*S,$$

la cual es $B + 8*S$.

Con un arreglo N-dimensional, el orden por renglón varía los subíndices en orden de derecha a izquierda. Por ejemplo para el arreglo:

$$A(L_1:U_1,L_2:U_2, \ldots, L_N:U_N)$$

Los elementos se almacenan en el siguiente orden:

$A(L_1,L_2,\ldots,L_N)$
$A(L_1,L_2,\ldots,L_N + 1)$
...
$A(L_1,L_2,\ldots,U_N)$
$A(L_1,L_2,\ldots,L_{N-1} + 1,L_N)$
...
$A(L_1,L_2,\ldots,L_{N-1} + 1,U_N)$
...
$A(L_1,L_2,\ldots,U_{N-1},U_N)$
...
$A(U_1,L_2,\ldots,L_{N-1},L_N)$
$A(U_1,L_2,\ldots,L_{N-1},L_N + 1)$
...
$A(U_1,L_2,\ldots,U_{N-1},U_N)$
...
$A(U_1,U_2,\ldots,U_N)$

Orden por columna

Otra alternativa de linealización de un arreglo bidimensional es almacenar los elementos en *orden por columna* es decir, almacenar primero la primer columna, después la segunda columna, después la tercera, y así sucesivamente. El arreglo PORCENTAJE, ya mencionado, aparece físicamente ordenado por columna en la figura 2-10.

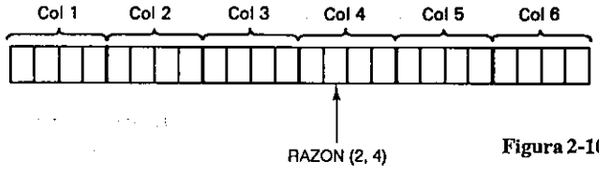

Figura 2-10 Arreglo de la figura 2-6, linealizado por columna.

Este esquema de almacenamiento se utiliza para arreglos declarados en varias implantaciones de FORTRAN.

Suponiendo que *B* es una dirección base del arreglo y que cada elemento del arreglo es de tamaño *S*, la dirección de inicio del elemento RAZON(*I,J*) es

$$B + (J - 1)*4*S + (I - 1)*S$$

ya que hay $J - 1$ columnas, cada una de longitud $4*S$, las cuales preceden a la columna del elemento RAZON(*I,J*), y hay $I - 1$ elementos, cada uno de longitud *S*, los cuales preceden a RAZON(*I,J*) en la columna *J*. El elemento RAZON(2,4) es localizado en $B + 13*S$.

Generalizando, el elemento ARREGLO(*I,J*) del arreglo definido como ARREGLO (L_1:U_1,L_2:U_2), ordenado por columna se localiza en:

$$B + (J - L_2)*(U_1 - L_1 + 1)*S + (I - L_1)*S. \qquad \text{(ecuación 2-4)}$$

Compare esta ecuación con la 2-3.

Con un arreglo *N* dimensional, el orden por columna varía a los subíndices de izquierda a derecha. Por ejemplo, para el arreglo:

$$A(L_1:U_1,L_2:U_2,\ldots,L_N:U_N)$$

los elementos se almacenan en el siguiente orden:

$A(L_1,L_2,\ldots,L_N)$
$A(L_1 + 1,L_2,\ldots,L_N)$
...
$A(U_1,L_2,\ldots,L_N)$
$A(L_1,L_2 + 1,\ldots,L_N)$
$A(L_1 + 1,L_2 + 1,\ldots,L_N)$
...
$A(U_1,L_2 + 1,\ldots,L_N)$
...
$A(L_1,L_2,\ldots,L_N + 1)$
$A(L_1 + 1,L_2,\ldots,L_{N+1})$
...
$A(U_1,L_2,\ldots,L_{N+1})$

...
$A(L_1,L_2,\ldots,U_N)$
...
$A(L_1,U_2,\ldots,U_N)$
...
$A(U_1,U_2,\ldots,U_N)$

Selección de una técnica de linealización

Para determinar cuando es más ventajoso almacenar un arreglo en orden por columna o en orden por renglón es necesario conocer en qué orden se referenciarán los elementos del arreglo. De hecho, los lenguajes de programación disponibles no dan oportunidad al programador de elegir la técnica de almacenamiento de arreglos. Es más, especialmente en un ambiente de memoria virtual, el programador debe hacer un esfuerzo por ajustar el patrón de referencias de los elementos del arreglo al patrón de almacenamiento. Por ejemplo, considere un procedimiento para calcular el valor promedio de los elementos de un arreglo llamado *A* de 50 por 225. Con un almacenamiento por columna se debe usar el siguiente patrón de referencias:

En COBOL:

```
COMPUTE TOTAL = 0.
PERFORM SUMA VARYING J FROM 1 BY 1 UNTIL J > 225
                    AFTER I FROM 1 BY 1 UNTIL I > 50.
```

donde

```
SUMA.
    COMPUTE TOTAL = TOTAL + A (I,J)
```

Este algoritmo suma todos los elementos de una columna antes de continuar con la siguiente. Lo mismo en lenguaje Pascal es:

```
total: = 0;
for j:= 1 to 225 do
    for i: = 1 to 50 do
        total: = total + a[i,j];
```

Por otra parte, con almacenamiento por renglón, el cual es más común en COBOL y Pascal, es preferible el siguiente patrón de referencia. El algoritmo suma todos los elementos de un renglón antes de continuar con el siguiente.

En COBOL:

```
COMPUTE TOTAL = 0.
PERFORM SUMA VARYING I FROM 1 BY 1 UNTIL I > 50
                    AFTER J FORM 1 BY 1 UNTIL J > 255.
```

donde

En Pascal:

```
SUMA.
    TOTAL = TOTAL + A(I,J).
```

```
total: = 0;
for i: = 1 to 50 do
   for j: = 1 to 255 do
   total: = total + a[i,j];
```

Cuando sea necesario pasar arreglos entre rutinas de FORTRAN y rutinas escritas en otros lenguajes (como COBOL, PL/I, Pascal), es de suma importancia que el programador esté enterado de las técnicas de linealización utilizadas.

ARREGLOS TRIANGULARES

En la sección anterior discutimos la linealización de arreglos multidimensionales, en esta sección consideraremos algunos aspectos de la linealización de tipos especiales de arreglo: los *arreglos triangulares.*

Definiciones

Un arreglo triangular puede ser uno de los dos arreglos *superior* o *inferior*, mostrados en las figuras 2-11 y 2-12, respectivamente. Donde todos los elementos de abajo (o arriba) de la diagonal son cero. Se dice que un arreglo es estrictamente triangular superior (o inferior) si los elementos de la diagonal, también, son cero.

$$\begin{bmatrix} X & X & X & X & X & X \\ 0 & X & X & X & X & X \\ 0 & 0 & X & X & X & X \\ 0 & 0 & 0 & X & X & X \\ 0 & 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & 0 & 0 & X \end{bmatrix}$$

Figura 2-11 Arreglo triangular superior.

$$\begin{bmatrix} X & 0 & 0 & 0 & 0 & 0 \\ X & X & 0 & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 \\ X & X & X & X & 0 & 0 \\ X & X & X & X & X & 0 \\ X & X & X & X & X & X \end{bmatrix}$$

Figura 2-12 Arreglo triangular inferior.

En un arreglo triangular inferior con N renglones, el máximo número de elementos, diferentes de cero en el I-ésimo renglón, es I. El total de elementos diferentes de cero es a lo más:

$$\sum_{I=1}^{N} I = \frac{N(N+1)}{2}$$

Esta expresión también es verdadera para un arreglo triangular superior con N renglones. Para una N grande, sería deseable no tener almacenados todos los valores iguales a cero del arreglo.

Linealización

Un enfoque para este problema es linealizar el arreglo y almacenar sólo los elementos diferentes de cero. Supongamos que se almacena un arreglo triangular superior T por renglones en un arreglo unidimensional, llamado S, con subíndices limitados entre 1 y

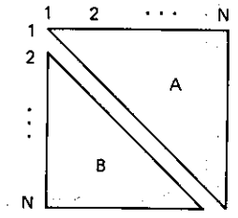

Figura 2-13 Arreglos triangulares superior e inferior que comparten un espacio de N por N.

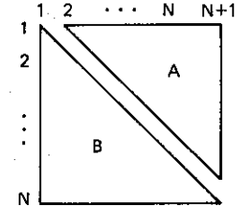

Figura 2-14 Arreglos triangulares superior e inferior que comparten un espacio de N por N + 1.

$N(N+1)/2$. El elemento $T(1,1)$ es almacenado en el elemento $S(1)$, el elemento $T(1,2)$ es almacenado en $S(2)$, y así sucesivamente hasta el elemento $T(1, N)$ el cual se almacena en el elemento $S(N)$. Entonces, el elemento $T(2,2)$ se almacena en $S(N+1)$, ya que $T(2,1)$ es cero. El elemento $T(N,N)$ se almacena en $S(N(N+1)/2)$. En general, ¿dónde se almacena $T(I,J)$ en S? Observe que $I \leq J$ para todo elemento en la parte diferente de cero de T.

Espacio compartido

Algunas veces un programa requiere el uso de más de un arreglo triangular. Si dos de estos arreglos son esencialmente de las mismas dimensiones, entonces es una tarea directa el almacenarlos, de tal forma que se economice espacio. Suponga que un arreglo A es un arreglo triangular superior de N por N y que B es un arreglo triangular inferior de $N - 1$ por $N - 1$. Entonces A y B se pueden almacenar juntas en un arreglo C de N por N como se muestra en la figura 2-13. Los elementos $C(I,J)$ donde $I \leq J$ son elementos de A, y los elementos $C(I,J)$ donde $I > J$ son elementos de B. De hecho los elementos $A(I,J)$ se almacenan como $C(I,J)$ para $I \leq J$ y los elementos $B(I,J)$ se almacenan como $C(I+1,J)$ para $I \geq J$.

Sea A un arreglo triangular superior, sea B un arreglo triangular inferior, ambos de N por N. Entonces, el arreglo C puede ser de N por $N+1$ para contener ambos arreglos A y B, tal como se muestra en la figura 2-14. Aquí, los elementos $A(I,J)$ se almacenan como $C(I,J+1)$ para $I \leq J$ y, $B(I,J)$ se almacena como $C(I,J)$ para $I \geq J$.

Considere ahora el caso en el cual dos arreglos triangulares, digamos A y AA (ambos de N por N), tienen que compartir el mismo espacio. Un método sería transponer uno de los arreglos triangulares, digamos AA, de tal forma que en lugar de ser un arreglo triangular superior, aparezca como un arreglo triangular inferior. Esto es, el arreglo AA se transpone para formar el arreglo AA^T, y los elementos $AA(I,J)$ se convierte en $A^T(J,I)$. Entonces A y AA^T pueden almacenarse juntos en un arreglo C con dimensiones N por $(N+1)$, tal y como se muestra en el párrafo anterior. El elemento $A(I,J)$ se almacena como $C(I,J+1)$ y el elemento $AA(I,J)$ es almacenado como $C(J,I)$.

Cambio de acomodo

Un problema con estas técnicas de linealización aparece cuando elementos de la parte inferior de un arreglo triangular superior se vuelven diferentes de cero, lo que convierte

0	0	0	0	1	0	0	2	0	0
0	1	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	4	0	0	0	0	0	0
0	0	0	0	0	0	0	2	0	0
0	0	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0	0

Figura 2-15 Ejemplo de un arreglo disperso.

al arreglo en un arreglo no triangular. Si se comparte el espacio de un arreglo cuadrado con un arreglo triangular inferior, entonces, habrá problemas. Una solución es no permitir que un arreglo triangular superior sea actualizado si eso causara que el arreglo no fuera triangular superior. Otro método, es reconocer el cambio en las características del arreglo transfiriéndolo a su propio espacio.

ARREGLOS DISPERSOS

Definición

Otro tipo especial de arreglos los cuales aparecen con frecuencia en aplicaciones, son los *arreglos dispersos*. Hoy en día es difícil establecer una demarcación entre estos arreglos y los que no lo son; informalmente, a un arreglo se le llama disperso si tiene, al parecer, una alta densidad de elementos cero. Por ejemplo, el arreglo de la figura 2-15, en la cual se muestran 8 elementos diferentes de cero de 80 que sí lo son, es disperso, esto es el 90% de ceros.

Linealización

Los arreglos dispersos multidimensionales se pueden linealizar a través de las técnicas ya presentadas en este capítulo; sin embargo, éstas podrían ocasionar el desperdicio de mucho espacio. Consideremos dos representaciones alternativas para almacenar explícitamente sólo elementos diferentes de cero.

Representación vectorial

Cada elemento diferente de cero en un arreglo disperso bidimensional se puede representar como una terna con el formato (subíndice del renglón, subíndice de la columna, valor). Entonces estas ternas pueden ordenarse incrementando el subíndice del renglón y el subíndice de la columna y se pueden almacenar como un vector. Por ejemplo, usando este método el arreglo disperso de 8 por 10 que se muestra en la figura 2-15 se representa por el vector V que se muestra en la figura 2-16.

Esta forma de almacenamiento usa más espacio para representar cualquier elemento diferente de cero pero evita la representación de los elementos cero.

Si el arreglo disperso fuese unidimensional, cada elemento diferente de cero se

	renglón	columna	valor
V(1):	1,	5,	1
V(2):	1,	8,	2
V(3):	2,	2,	1
V(4):	3,	1,	1
V(5):	5,	4,	4
V(6):	6,	8,	2
V(7):	8,	1,	2
V(8):	8,	2,	1

Figura 2-16 Representación vectorial del arreglo disperso de la figura 2-15.

representaría por una pareja. En general, cada elemento diferente de cero de un arreglo N-dimensional se representa con $N + 1$ valores.

Las operaciones sobre la representación vectorial de un arreglo disperso de dos dígitos deben utilizar la información sobre renglón y columna para determinar dónde se encuentra cada valor. Un inconveniente que se presenta al usar esta forma de representación es el manejar la suma de los elementos diferentes de cero del arreglo o cambiar valores no cero a valores cero. El problema no está en el cambio del valor, sino en mantener el orden de los elementos diferentes de cero en la representación vectorial. Por ejemplo, si el arreglo de arriba fuera actualizado de tal forma que el elemento con subíndices (1,8) cambiara su valor a cero, entonces los elementos V(3) a V(8) del vector deberían recorrerse hasta convertirse en los elementos V(2) a V(7). De manera similar, si los elementos con subíndices (4,6) fueran actualizados con el valor 9, entonces los elementos V(5) a V(8) necesitarían recorrerse hasta convertirse en los elementos V(6) a V(9) y V(5) sería la terna (4,6,9).

Representación de listas ligadas

Otra representación para arreglos dispersos es el uso de listas ligadas. Diferiremos la consideración de esa importante alternativa hasta el capítulo 6, donde se analizan las listas ligadas.

RESUMEN

Un *arreglo* es un conjunto finito de elementos homogéneos ordenados. Los elementos del arreglo son referenciados según sus posiciones, las cuales se especifican por subíndices. La composición de arreglos en los cuales cada elemento es un arreglo, da como resultado un arreglo multidimensional. El capítulo presentó la declaración de arreglos en programas en COBOL y Pascal.

Los arreglos unidimensionales pueden almacenarse en una forma lineal. Para calcular la dirección de inicio de un elemento en particular, es necesario conocer la localidad base del arreglo y del tamaño de cada elemento precedente en el arreglo. Los arreglos multidimensionales pueden linealizarse de acuerdo a la forma de almacenamiento que se elija. Las técnicas más comunes de linealización para arreglos bidimensionales, son el almacenamiento por renglón mayor (COBOL, Pascal, C, y PL/I) o por columna mayor (FORTRAN).

El capítulo termina con el análisis de dos tipos especiales de arreglos: arreglos triangulares y arreglos dispersos. Para estos tipos de arreglos, el uso de las técnicas de linealización convencionales por lo general ocasiona excesivos requerimientos de espacio. Las técnicas de linealización adecuadas a las características de los arreglos triangulares y dispersos comprenden el almacenamiento de dos arreglos triangulares unidos en un solo arreglo cuadrado y la representación exclusiva de los elementos diferentes de cero de arreglos dispersos.

Las técnicas de direccionamiento en arreglos se analizan con detalle en el capítulo 13, en la sección sobre técnicas de cálculo de dirección para archivos relativos.

TERMINOLOGIA

Arreglo
Arreglo disperso
Arreglo estrictamente triangular
Arreglo triangular
Arreglo triangular inferior
Arreglo triangular superior
Indice
Límite inferior
Límite superior
Localidad base
Orden por columna
Orden por renglón
Rango
Sección transversal
Subíndice
Tabla
Transpuesta
Vector

REFERENCIAS SUGERIDAS

DeMillo, R. A., S. C. Eistenstat, and R. J. Lipton. "Preserving average proximity in arrays," *Comm. ACM* 21(3): 228–231, March 1978.

Greenberg, H. J. and R. P. O'Neill. "Representing super-sparse matrices with perturbed values," *Comm. ACM* 24(7): 451–456, July 1981.

Hellerman, H. "Addressing multidimensional arrays," *Comm. ACM* 5(4): 205–207, April 1962.

MacVeigh, D. T. "Effect of data representation on cost of sparse matrix operations," *Acta Informatica* 7(4): 361–394, 1977.

Pooch, U. W., and A. Nieder. "A survey of indexing techniques for sparse matrices," *ACM Computing Surveys* 5(2): 109–133, June 1973.

Rosenberg, A. L. and L. J. Stockmeyer. "Hashing schemes for extendible arrays," *Jour. ACM* 24(2): 199–221, April, 1977.

Rosenberg, A. L., and L. J. Stockmeyer. "Storage schemes for boundedly extendible arrays," *Acta Informatica* 7(3): 289–303, 1977.

Rosenberg, A. L. "Allocating storage for extendible arrays," *Jour. ACM* 21(4): 652–670, Oct. 1974.

Standish, T. A. "Arrays," Chapter 8 in *Data Structure Techniques*, Reading, Mass.: Addison-Wesley Publishing Co., pp. 348–376, 1980.

Tarjan, R. E. and A. C. Yao. "Storing a sparse table," *Comm. ACM* 22(11): 606–611, Nov. 1979.

EJERCICIOS DE REPASO

1. En un arreglo con columnas numeradas del 4 al 13 y renglones de 6 a 12, ¿cuál es el número máximo de elementos que se pueden almacenar?
2. ¿Qué significan los términos "orden por renglón" y "orden por columna"?
3. ¿Cuántos elementos hay en el arreglo ESPACIO(*A*:*B*,*C*:*D*)?
4. ¿Cuántos elementos pueden contener arreglos con dimensiones A(1:*N*) y B(- *N*:0,3)?
5. El arreglo PRUEBA(1 : 10,1 : 5) se almacena en memoria por columnas. Si la localidad base es $B = 0$ y el tamaño del elemento es $S = 1$, ¿cuál es la dirección del elemento PRUEBA(7, 2)?
6. Dado ARREGLO(10:30, 10:100), con localidad base = 50 y tamaño de elemento = 8 bits:
   a) ¿Cuántos elementos hay en el arreglo?
   b) ¿Cuál es la localidad de inicio de ARREGLO(21,75) si ARREGLO se almacena por renglón?
7. Dado el arreglo A(50:100, 50:75), ¿cuál es la localidad de inicio de A(62,56)?
8. Escriba una fórmula para encontrar la dirección del *J*-ésimo elemento en el *i*-ésimo renglón del arreglo IVAN(*B*:*A*,*D*:*C*), donde cada elemento en el arreglo ocupa *n* palabras de memoria. Asuma una localidad de inicio igual a 0.
9. Considere el arreglo Q(*A*:*B*,*C*:*D*,*E*:*F*), la localidad base = *X*, longitud de elemento = *L*;
   a) Encuentre el número total de elementos del arreglo.
   b) Encuentre la localidad de inicio de Q(*I*,*J*,*K*). ¿Qué suposición tuvo que adoptar acerca de cómo se linealiza y almacena el arreglo?
10. ¿Cuál es el número máximo de dimensiones disponible para arreglos en COBOL?
11. Dado el siguiente arreglo en COBOL:

```
01 TABLA.
   03 RENGLONES OCCURS 50 TIMES.
      05 COLUMNAS OCCURS 4 TIMES.
         07 PROFUNDIDAD-DE TABLA OCCURS 6 TIMES PIC XX.
```

¿Cuántos elementos contiene la tabla?

12. Ilustre y describa un método para representar arreglos dispersos.
13. Escriba un algoritmo para transponer un arreglo.
14. Escriba un algoritmo para almacenar el siguiente arreglo disperso como un vector de tres componentes (subíndice del renglón, subíndice de la columna, valor)

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 9 & 0 & 1 \\ 0 & 16 & 0 & 0 & 0 & 0 \end{bmatrix}$$

15. Escriba un algoritmo para calcular el producto de dos matrices dispersas, cada una representada por un arreglo.

16. Muestre cómo pueden almacenarse juntos dos arreglos triangulares.

17. ¿Qué ventajas y desventajas deben considerarse cuando se almacenan dos arreglos triangulares en un espacio cuadrado?

18. Muestre que un arreglo triangular superior con N renglones no tiene más de $N(N + 1)/2$ elementos diferentes de cero.

19. Considere el almacenamiento por renglones de un arreglo triangular inferior de N por N, llamado T, en un arreglo unidimensional S, con subíndices de 1 a $N(N + 1)/2$. ¿Cuál es el subíndice en S para el elemento $T(I,J)$? (¡Es difícil!)

20. Un tipo de arreglo disperso que ocurre con cierta frecuencia es un arreglo "tridiagonal", donde todos los elementos, excepto aquellos sobre la diagonal principal y las diagonales inmediatamente arriba y abajo, son cero. Por ejemplo, un arreglo tridiagonal de 5 por 5, tiene la estructura:

$$\begin{bmatrix} a_{11} & a_{12} & 0 & 0 & 0 \\ a_{21} & a_{22} & a_{23} & 0 & 0 \\ 0 & a_{32} & a_{33} & a_{34} & 0 \\ 0 & 0 & a_{43} & a_{44} & a_{45} \\ 0 & 0 & 0 & a_{54} & a_{55} \end{bmatrix}$$

Considere el almacenamiento de un arreglo tridiagonal en un arreglo unidimensional B(I), con a_{11} en B(1), a_{12} en B(2); y así sucesivamente. ¿Cuál es el subíndice, en B, del elemento a_{ij}?

21. Escriba un programa para recorrer un laberinto. El laberinto es una tabla bidimensional, de 12 por 12 con valores de 1, que indican las posibles rutas y 0 que indican las paredes. La figura P2-21 es un ejemplo de laberinto.

Habrá un solo elemento en la primera columna con el valor de 1 el cual será el punto de inicio. Habrá, también sólo un único valor 1 en la última columna el cual indicará el punto final. El laberinto se leerá con un renglón por línea. Los valores del renglón estarán en las 12 columnas de la línea.

Haga su propio laberinto para probar y después use el ejemplo de arriba.

0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0
0	1	0	0	0	0	1	1	1	0	0	0
0	1	0	0	0	0	1	0	1	0	1	0
1	1	0	0	1	1	1	0	1	1	1	1
0	1	1	1	0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	0	0	0	0	0
0	0	0	1	0	1	1	1	1	0	0	0
0	1	1	1	1	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	0	0	0	1	0
0	1	1	1	1	1	1	1	1	1	1	0

Figura P2-21 Ejemplo de un laberinto.

La salida debe consistir de la ruta (excluyendo las vueltas) especificada por los renglones y columnas de los elementos recorridos desde el inicio hasta el final del laberinto. No puede recorrer en diagonal.

22. Hay un tipo de dato conocido como cadena de longitud variable, el cual tiene un atributo de longitud que puede tomar diferentes valores a la vez. Considere un arreglo de cadenas de longitud variable, por ejemplo el arreglo MES(1:12), donde cada elemento es una cadena de longitud de 0 y 9 caracteres, cuyo valor es la longitud del nombre de cada mes. Una forma de almacenar este arreglo es reservar espacio para 10 caracteres para cada elemento, sin considerar qué tanto espacio se utiliza en realidad. ¿Cómo se puede almacenar el arreglo, de tal forma que el total de espacio reservado sea mínimo? ¿Su técnica funcionaría si se modificaran los valores de los elementos del arreglo?

23. ¿Por qué los algoritmos convencionales para encontrar elementos en un arreglo, requieren que todos sean del mismo tipo de estructura de datos?

24. Algunos lenguajes de programación, por ejemplo COBOL y FORTRAN dan por omisión el límite inferior 1 para arreglos. Otros, como C, dan 0 como límite inferior. Sugerimos que practique la programación que pueda ayudarle a erradicar las dificultades en subindización que se puedan encontrar en la convención de programas FORTRAN a C (o viceversa).

25. Algunos lenguajes de programación como FORTRAN, almacenan arreglos en orden por columna. Otros como C y COBOL almacenan arreglos en orden por renglón. Ilustre los problemas encontrados al pasar arreglos entre programas FORTRAN, C, o COBOL.

26. Considere el arreglo Q(- 1:1,1:2,1:3).

a) ¿Cuántos elementos hay en Q?

b) Liste la secuencia de elementos, si Q es almacenada en orden por renglón.

c) Liste la secuencia de elementos, si Q es almacenada en orden por columna.