

Matrices

Sitio: [Agencia de Aprendizaje a lo largo de la Vida](#)
Curso: Tecnicas de Programación 1° G
Libro: Matrices

Imprimido por: Sebastian Puche
Día: domingo, 13 de octubre de 2024, 17:12

Tabla de contenidos

1. Introducción

2. Arreglos ordenamiento

2.1. Método de ordenamiento por máximos sucesivos

3. Arreglos bidimensionales (de dos dimensiones) o matrices

3.1. Inicialización de matrices

4. Carga de datos en matrices

4.1. Carga total de los elementos de una matriz

4.2. Carga parcial, o bien aleatoria, de los elementos de una matriz

5. Recorrido de matrices

5.1. Recorrido por filas

5.2. Recorrido por columnas

6. Posicionamiento indirecto utilizando vectores y matrices

1. Introducción

En esta semana continuaremos incorporando conocimientos sobre la forma en que podemos almacenar [datos para su procesamiento](#). Hasta el momento hemos usado [variables de tipo simple](#) (entero, carácter, etc.). Una [variable de tipo simple](#) consiste de una sola “caja” de memoria y sólo puede contener un valor, es decir, existe una relación de uno a uno entre la [variable y el número de elementos](#) (valores) que es capaz de almacenar, por ejemplo, un contador. Además, trabajamos con [arreglos](#) que nos permiten almacenar una colección de elementos del mismo tipo bajo un nombre único y acceder fácilmente a cada elemento individual.

Veremos cómo realizar un ordenamiento de la información y poder mediante esta técnica tener los valores del vector de manera ordenada bajo un determinado criterio.

2. Arreglos ordenamiento

Uno de los procedimientos más comunes y útiles en el procesamiento de datos es la [clasificación u ordenamiento](#) de los mismos. Se considera ordenar al proceso de reorganizar un conjunto dado de objetos en una secuencia determinada. Cuando se analiza un [método de ordenamiento](#), hay que determinar cuántas comparaciones e intercambios se realizan para el caso más favorable, para el caso medio y para el caso más desfavorable.

[La colocación en orden de una lista de valores se llama ordenamiento](#). Por ejemplo, se podría disponer una lista de valores numéricos en orden ascendente o descendente o bien una lista de nombres en orden alfabético.




Existen varios métodos para ordenamiento, explicaremos uno de ellos a continuación.

2.1. Método de ordenamiento por máximos sucesivos

Es el método más sencillo de comprender, lamentablemente no es el más efectivo porque realiza muchos recorridos del arreglo y muchas comparaciones y esto se traduce en mucho tiempo de procesamiento.

Para resumir, si el arreglo que tenemos que ordenar es pequeño, este puede ser un buen método, pero si el arreglo es muy grande, es recomendable utilizar otro método.

Para nuestro ejemplo vamos a ordenar un [arreglo de números enteros en orden descendente](#). Esto es de mayor a menor, en la primera posición quedará el mayor valor y en la última el más pequeño.



La idea de este método es hacer una primera recorrida por el arreglo buscando el mayor elemento y dejándolo en la primera posición; como siguiente paso recorrer el arreglo desde la segunda posición buscando el mayor y colocándolo en la segunda posición y así sucesivamente.

Analizamos la estrategia [sobre un arreglo de cinco elementos enteros](#) (por comodidad lo presentamos en forma vertical en vez de horizontal, con lo cual la celda cero nos queda en la parte superior del gráfico y la celda cuatro en la inferior).

Por ejemplo:

Arreglo inicial

Celda	Valor original
0	23
1	19
2	45
3	15
4	31

Primera pasada

Se busca el [valor máximo dentro del arreglo](#) desde la posición cero hasta la posición cuatro. En este caso está en la posición 2, con lo cual se intercambia con la posición cero y de esta manera ya tenemos el mayor en la posición cero del arreglo.

Celda	Valor original	Cambios realizados	Comentarios
0	23	45	Se intercambia con el 45 que es el mayor.
1	19		
2	45	23	
3	15		
4	31		

Segunda pasada

Se busca el [valor máximo dentro del arreglo desde la posición uno hasta la posición 4](#). En este caso, está en la posición 3, con lo cual se intercambia con la posición uno y de esta manera ya tenemos el segundo valor mayor en la posición uno del arreglo.

Celda	Valor original	Luego primer pasada	Cambios realizados	Comentarios
0	23	45		
1	19	19	31	Se intercambia el 19 con el 31 que es el mayor.
2	45	23		
3	15	15		
4	31	31	19	

Tercera pasada

Se busca [el valor máximo dentro del arreglo desde la posición dos hasta la posición 4](#). En este caso está en la posición dos, con lo cual se intercambia con el mismo valor sin notar modificaciones, hasta acá ya tenemos el tercer valor mayor en la segunda posición del arreglo.

Celda	Valor original	Luego primer pasada	Luego segunda pasada	Cambios realizados	Comentarios
0	23	45	45		
1	19	19	31		
2	45	23	23	23	Observar que en esta pasada, el mayor valor está en la 2da posición, con lo cual lo intercambia con el mismo valor.
3	15	15	15		
4	31	31	19		

Cuarta pasada

Se busca el [valor máximo dentro del arreglo desde la posición tres hasta la posición cuatro](#). En este caso, está en la posición cuatro, con lo cual se intercambia con la posición tres y de esta manera ya tenemos el arreglo ordenado ya que estamos seguros que en la última posición tenemos el valor más pequeño.

Celda	Valor Original	Luego primer pasada	Luego segunda pasada	Luego tercera pasada	Cambios realizados	Comentarios
0	23	45	45	45		
1	19	19	31	31		
2	45	23	23	23		
3	15	15	15	15	19	Esta es la última pasada ya que estamos seguros que en la última celda tenemos el menor valor.
4	31	31	19	19	15	

Arreglo resultante

Celda	Valor original
0	45
1	31
2	23
3	19
4	15

Generamos un vector con valores aleatorios y luego realizamos el ordenamiento:

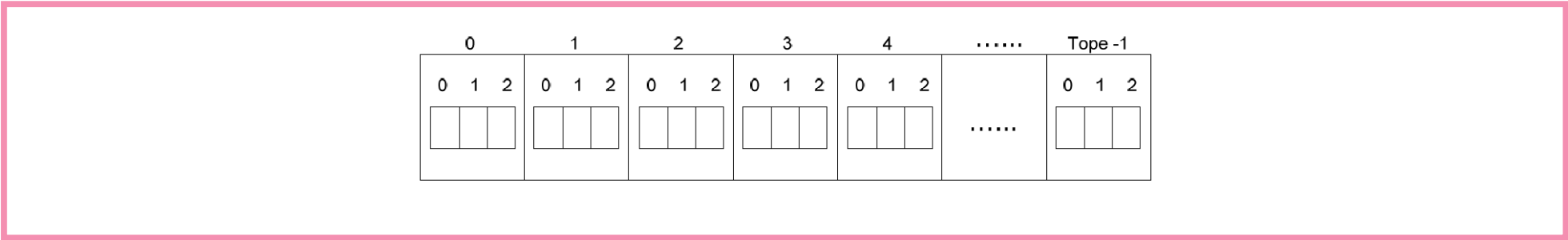
```
1  Algoritmo ordenamiento
2      definir vec, i, j, posicion,max Como Entero
3      dimension vec(5)
4      // asignamos valores aleatorios a cada posición del arreglo
5      Para i = 0 hasta 4 Hacer
6          ..... vec[i]=Aleatorio(1,200)
7      Finpara
8
9      // Proceso de ordenamiento
10     Para i = 0 hasta 3 Hacer
11         ..... max = vec[i]
12         Para j = i+1 hasta 4 Hacer
13             ..... si vec[j] > max
14                 ..... posicion = j
15                 ..... max = vec[j]
16             FinSi
17         FinPara
18         Si max ≠ vec[i]
19             ..... vec[posicion] = vec[i]
20             ..... vec[i] = max
21         FinSi
22     FinPara
23
24 FinAlgoritmo
```

Si decimos que tope es la cantidad de elementos del arreglo, el ciclo For irá desde cero hasta $\text{TOPE}-1$, pero con lo visto anteriormente, este método realiza un recorrido menos que la cantidad de elementos del arreglo intercambiando los valores ya que el último siempre será el menor, con lo cual el índice va desde cero hasta $\text{TOPE}-2$.

3. Arreglos bidimensionales (de dos dimensiones) o matrices

Como mencionamos anteriormente, un arreglo puede ser de un tipo de datos simple como [int](#), [char](#), [float](#) y también que puede ser de un arreglo, es decir que cada celda del arreglo contenga un arreglo.

Esquematzamos un arreglo que tiene [TOPE elementos](#) (índice cero hasta TOPE-1) y cada elemento es un arreglo de tres posiciones.



¿Para qué podemos utilizar una estructura con estas características?

Supongamos que tenemos un arreglo con las notas finales de tres materias de los seis alumnos/as de un curso. Pero nos piden discriminar para cada alumno/a (que lo almacenamos en el arreglo que va desde el índice 0 hasta TOPE -1), cuál fue la nota final en cada una de las 3 materias.

La declaración de este arreglo se realiza de la siguiente manera (vamos a suponer que los datos a almacenar son números reales, es decir, admiten decimales):

```
Definir TOPE_F como Entero

TOPE_F = 6 //constante para la cantidad de alumnos

Definir TOPE_C como Real

TOPE_C = 3 //constante para la cantidad de materias

//Definimos la variable

Definir alumnosPorMateria como Entero

//Dimensionamos la variable

Dimension alumnosPorMateria[TOPE_F , TOPE_C]
```

Cómo acceder a los datos en arreglos de arreglos

Nota: para nuestra explicación tomamos como alumno cero al primer alumno que se encuentra en el arreglo para facilitar la comprensión.

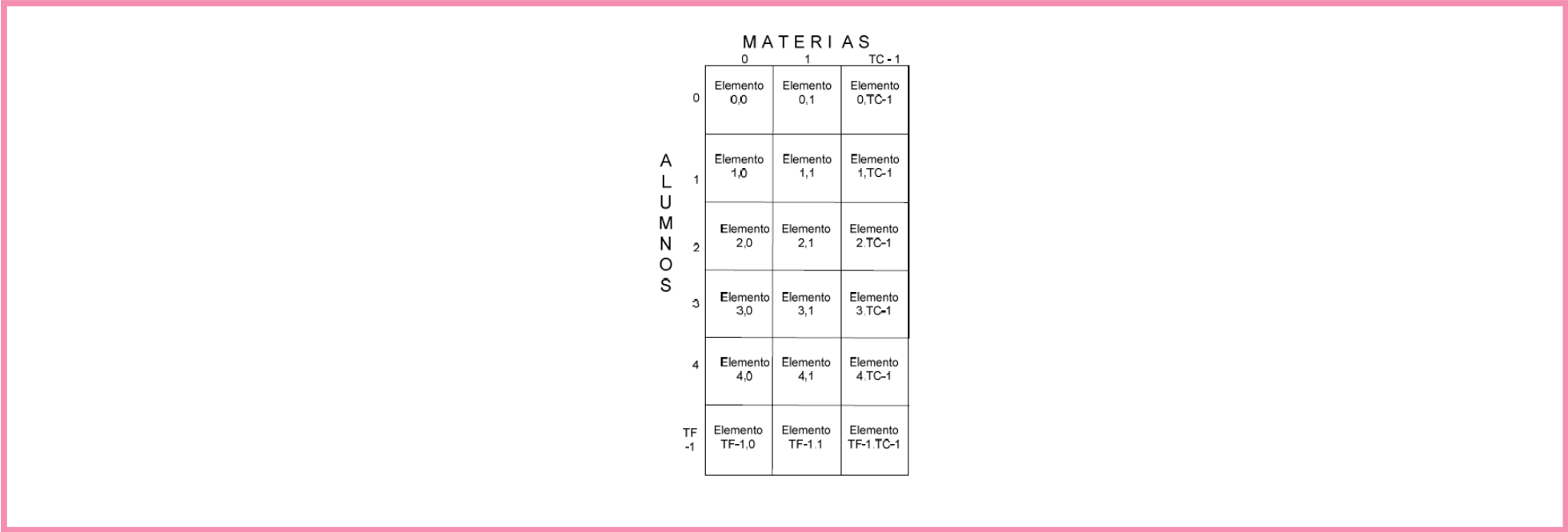
Supongamos que queremos mostrar para el cuarto alumno (celda 3), la nota de la primera materia (celda 0).

```
Escribir alumnosPorMateria[3,0]
```

Estructuras como las que acabamos de presentar, [se denominan arreglos de dos dimensiones o arreglos bidimensionales o bien matrices](#).

Una matriz se utiliza [cuando queremos representar una tabla de doble entrada](#). A la primera dimensión la denominamos filas (en nuestro ejemplo serían los alumnos) y a la segunda dimensión la llamaremos columnas (materias).

Si tenemos una matriz de [TOPE_F filas por TOPE_C columnas](#) (constantes enteras definidas anteriormente), se puede esquematizar de la siguiente manera:



En dónde la primera dimensión (los/as alumnos/as) será representada por las filas y la segunda dimensión (las materias) será representada por las columnas.

Una representación con números será:

		MATERIAS		
		0	1	TC-1
A L U M N O S	0	8.75	4.25	6.00
	1	10.00	9.50	8.75
	2	6.00	3.50	10.00
	3	5.25	6.50	7.00
	4	8.75	4.25	9.00
	TF-1	2.00	10.00	7.50

Observemos que, por ejemplo, la nota del alumno 4 en la materia 1 es: 4.25

3.1. Inicialización de matrices

Por ejemplo, supongamos que quisiéramos inicializar los elementos de la matriz anteriormente definida con valor cero, habrá que:

1. Por cada fila, recorrer todas sus columnas e ir poniendo el valor.

O bien:

2. Por cada columna, recorrer todas sus filas y asignarle el valor elegido.

Método a

```

1  Algoritmo InicializarMatrizPorFila
2      Definir matriz,f,c,tope_f,tope_c Como Entero
3      Escribir "Ingrese la cantidad de filas de la matriz"
4      Leer tope_f
5      Escribir "Ingrese la cantidad de columnas de la matriz"
6      Leer tope_c
7      dimension matriz(tope_f,tope_c)
8      Para f = 0 Hasta tope_f-1 Hacer
9          Para c = 0 Hasta tope_c-1 Hacer
10             matriz[f,c] = 0
11         FinPara
12     FinPara
13 FinAlgoritmo

```

Método b

```

1  Algoritmo InicializarMatrizPorColumna
2      Definir matriz,f,c,tope_f,tope_c Como Entero
3      Escribir "Ingrese la cantidad de filas de la matriz"
4      Leer tope_f
5      Escribir "Ingrese la cantidad de columnas de la matriz"
6      Leer tope_c
7      dimension matriz(tope_f,tope_c)
8      Para c = 0 Hasta tope_c-1 Hacer
9          Para f = 0 Hasta tope_f-1 Hacer
10             matriz[f,c] = 0
11         FinPara
12     FinPara
13 FinAlgoritmo

```



Importante: observar que para el método b) lo que se hizo fue invertir los recorridos de los ciclos For.

Nunca se deben invertir los índices de la matriz.

Porque, volviendo a nuestro ejemplo, tenemos definida una matriz de 6 filas por 3 columnas, con lo cual, si invertimos los índices en lugar de los ciclos, accederemos, por ejemplo al elemento de la fila 2 columna 3 la cual es inexistente.

4. Carga de datos en matrices



En el próximo capítulo conoceremos los dos tipos de carga de datos en matrices.

4.1. Carga total de los elementos de una matriz

Retomando nuestro ejemplo de alumnos y materias, si quisiéramos cargar los valores en cada elemento de la matriz lo deberíamos hacer de la siguiente manera:

```
1  Algoritmo CargarMatriz
2      Definir matriz, tope_f, tope_c, f, c Como Entero
3      //supongamos 20 alumnos y 4 materias por alumno
4      Dimension matriz(20,4)
5
6      // asignamos los límites de filas y columnas
7      tope_f = 20
8      tope_c = 4
9
10     //cargamos la matriz
11     Para f=0 Hasta tope_f-1 Hacer
12         Para c = 0 Hasta tope_c-1 Hacer
13             Escribir "Ingresa la nota para el alumno " ,f+1," materia " ,c+1
14             Leer matriz[f,c]
15         FinPara
16     FinPara
17
18 FinAlgoritmo
```

4.2. Carga parcial, o bien aleatoria, de los elementos de una matriz

Supongamos que el usuario no necesita [ingresar todos los datos en la matriz](#), o bien lo necesita realizar en forma aleatoria, es decir, le daremos la posibilidad de elegir a qué alumno/a (en la variable F) y materia (en la variable C) ingresará la nota. Recordemos que se debe validar que el usuario no pueda ingresar un valor más allá de los “**TOPES**” de los [arreglos](#). La finalización del ingreso de datos se produce ingresando -1 para el alumno. Esquematizando lo anteriormente mencionado:

```

1  Algoritmo CargarMatrizNoCompleta
2    Definir matriz, tope_f, tope_c, f, c Como Entero
3    //supongamos 20 alumnos y 4 materias por alumno
4    Dimension matriz(20,4)
5
6    // asignamos los límites de filas y columnas
7    tope_f = 20
8    tope_c = 4
9
10   //inicializamos la matriz en 0
11   Para f = 0 hasta 19
12   |   Para c = 0 hasta 3
13   |   |   matriz[f,c] = 0
14   |   FinPara
15   FinPara
16
17   //cargamos la matriz
18   Escribir "Ingrese el número de alumno:"
19   Leer f
20   Mientras f ≠ -1
21   |   Mientras f > tope_f o f < 1
22   |   |   Escribir "Ingrese el número de alumno:"
23   |   |   Leer f
24   |   FinMientras
25   |
26   |   Repetir
27   |   |   Escribir "Ingrese el número de materia:"
28   |   |   Leer c
29   |   |   Hasta que c > 0 y c ≤ tope_c
30   |   |   Escribir "Ingresa la nota para el alumno " ,f," materia ",c
31   |   |   Leer matriz[f-1,c-1]
32   |   |   Escribir "Ingrese el número de alumno:"
33   |   |   Leer f
34   |   FinMientras
35   FinMientras
36
37  FinAlgoritmo

```

5. Recorrido de matrices



A continuación conoceremos dos formas de recorrer matrices.

5.1. Recorrido por Filas

Supongamos que queremos mostrar todo el contenido de la matriz, por ejemplo: para cada alumno/a, cuáles fueron sus calificaciones en cada una de las materias.

```
1  Algoritmo RecorrerMatriz
2      Definir matriz, tope_f, tope_c, f, c Como Entero
3      //supongamos 20 alumnos y 4 materias por alumno
4      Dimension matriz(20,4)
5
6      // asignamos los límites de filas y columnas
7      tope_f = 20
8      tope_c = 4
9      .....
10     //recorremos la matriz
11     Para f = 0 hasta tope_f - 1 Hacer
12         Escribir "Las notas del alumno ", f + 1, " son:"
13         Para c = 0 hasta tope_c - 1
14             Escribir "Para la materia ", c + 1, ": " matriz[f,c]
15         FinPara
16         Escribir " "
17     FinPara
18
19 FinAlgoritmo
```

5.2. Recorrido por columnas

Teniendo [almacenada en la matriz](#) las notas de los/as alumnos/as, debemos calcular para cada materia y mostrar cuál fue la nota más elevada y el alumno que la obtuvo (suponer único máximo).

```

1  Algoritmo RecorrerMatrizColumna
2      Definir pos, max, matriz, tope_f, tope_c, f, c Como Entero
3      //supongamos 20 alumnos y 4 materias por alumno
4      Dimension matriz(20,4)
5
6      // asignamos los límites de filas y columnas
7      tope_f = 5
8      tope_c = 4
9
10     //recorremos la matriz y buscamos el máximo por columna
11     Para c = 0 hasta tope_c - 1 Hacer
12         max = matriz[0,c]
13         Para f = 0 hasta tope_f - 1
14             Si matriz[f,c] > max Entonces
15                 max = matriz[f,c]
16                 pos = f
17             FinSi
18         FinPara
19         Escribir "Para la materia ", c + 1, " la nota más alta en un ", max, " y la obtuvo el alumno ", pos + 1
20         Escribir " "
21     FinPara
22
23 FinAlgoritmo

```



Observá que en este último ejemplo necesitamos [recorrer la matriz por columnas](#), es decir, para cada una de las columnas [ir analizando todas sus filas](#).

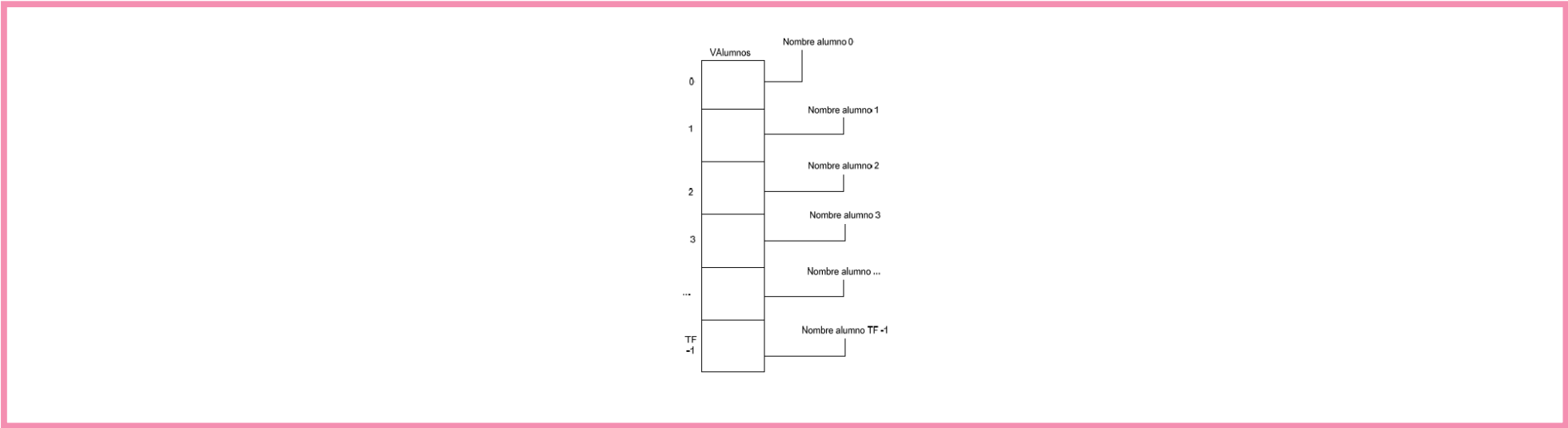
Lo que difiere con el ejemplo anterior es [que invertimos los ciclos](#), es decir, ahora el [ciclo externo es el de las columnas y el interno el de las filas](#), pero los [índices, al hacer referencia a la matriz, son siempre iguales](#), primero las filas y luego las columnas.

6. Posicionamiento indirecto utilizando vectores y matrices

Supongamos que tenemos un vector con los nombre de los alumnos y una matriz donde almacenamos las notas de cada uno de esos alumnos.

Y la idea es que el usuario ingrese el nombre de un alumno (que está [almacenado en una estructura independiente de la matriz](#)) y luego poder acceder a [la matriz cuando sepamos el código](#)

. La estructura adecuada para almacenar los nombres, [es un arreglo o vector unidimensional](#) en dónde cada celda contiene el nombre del alumno. Con lo cual, nuestro gráfico queda de la siguiente manera:

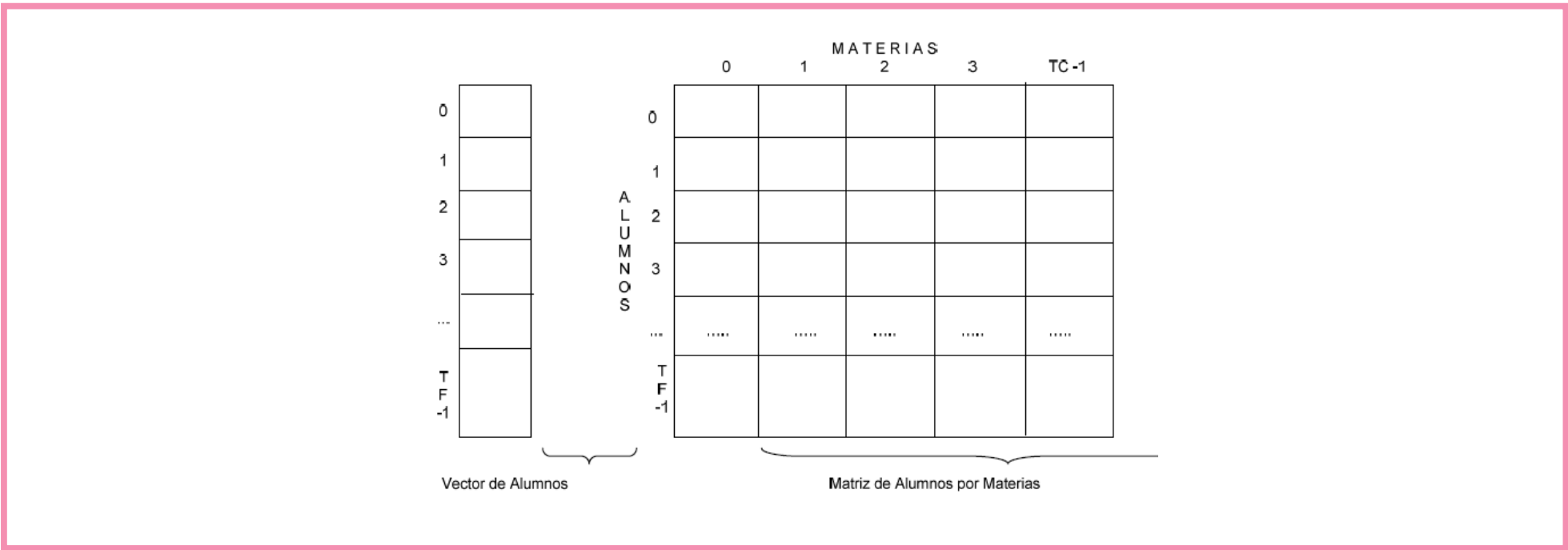


Pero, ¿cómo hacer para mostrar una nota si el/la usuario/a nos ingresa el nombre del alumno?, ¿cómo relacionar el vector de nombres con la matriz de notas?



Recordemos que los índices en los [arreglos](#) son del tipo entero, y el/la usuario/a está ingresando una palabra, con lo cual, necesitamos tener los nombres de los alumnos cargados en una estructura (vector) para poder realizar una búsqueda y obtener la posición del alumno en el arreglo para luego utilizarla como índice en la matriz.

Lo que haremos es lo que se conoce como [colocar un arreglo paralelo a la matriz](#). Para comprenderlo mejor, veamos el siguiente esquema:



El/la usuario/a ingresa el nombre de un alumno, se realiza la búsqueda dentro del vector de alumnos, si se encuentra el nombre buscado, entonces guardamos la posición y la utilizamos como índice de fila para acceder a la matriz.

A esto se lo [denomina posicionamiento indirecto, o también direccionamiento indirecto](#), porque para acceder a un elemento en una estructura necesitamos previamente realizar un proceso para conocer su ubicación.

En este ejemplo, para acceder a los datos de un alumno en la matriz de notas, debemos previamente buscar su nombre en el vector de alumnos para determinar a qué fila de la matriz acceder. Esta búsqueda la realizaremos en el ingreso y solamente dejamos de pedir datos al/la usuario/a cuando [la función Buscar devuelve](#) un número mayor que -1, es decir, que encontremos una posición dentro del arreglo que contenga el código ingresado.



Veamos cómo queda nuestro ejercicio:

```

1  Algoritmo RecorrerMatrizColumna
2      Definir pos, tope_f, tope_c, f, c Como Entero
3      Definir nombre,matriz como caracter
4      //supongamos 3 alumnos
5      Dimension matriz(3,2)
6
7      // asignamos los límites de filas y columnas
8      tope_f = 3
9      tope_c = 2
10
11     //Cargamos la matriz para el ejemplo (debería ya existir este arreglo)
12     matriz[0,0]="Juan"
13     matriz[1,0]="Sandra"
14     matriz[2,0]="Laura"
15     matriz[0,1]="5"
16     matriz[1,1]="7"
17     matriz[2,1]="9"
18
19     //Ingresamos el nombre del alumno del cual queremosps conocer la nota
20     Escribir "Ingrese el nombre del alumno: "
21     Leer nombre
22
23     //Buscamos la posición del alumno en el arreglo
24     pos = -1
25     Para f = 0 hasta tope_f - 1 Hacer
26     |     Si matriz[f,0] = nombre Entonces
27     |     |     pos = f
28     |     FinSi
29     FinPara
30
31     //Mostramos lo solicitado
32     Si pos = -1 Entonces
33     |     escribir "No se encuentra ese alumno en el arreglo"
34     SiNo
35     |     escribir "La nota de ", nombre," es ", matriz[pos,1]
36     FinSi
37
38     FinAlgoritmo

```

Podemos optimizar el algoritmo si nos detenemos al encontrar al alumno que buscamos.

```

1  Algoritmo RecorrerMatrizColumna
2      Definir pos, tope_f, tope_c, f, c Como Entero
3      Definir nombre,matriz como caracter
4      Definir existe como logico
5
6      //supongamos 3 alumnos
7      Dimension matriz(3,2)
8
9      // asignamos los límites de filas y columnas
10     tope_f = 3
11     tope_c = 2
12
13     //Cargamos la matriz para el ejemplo (debería ya existir este arreglo)
14     matriz[0,0]="Juan"
15     matriz[1,0]="Sandra"
16     matriz[2,0]="Laura"
17     matriz[0,1]="5"
18     matriz[1,1]="7"
19     matriz[2,1]="9"
20
21     //Ingresamos el nombre del alumno del cual queremosps conocer la nota
22     Escribir "Ingrese el nombre del alumno: "
23     Leer nombre
24
25     //Buscamos la posición del alumno en el arreglo hasta encontrarlo o hasta completar el arreglo
26     pos = -1
27     f = 0
28     existe = Falso
29     Mientras !existe y f < tope_f Hacer
30         Si matriz[f,0] = nombre Entonces
31             pos = f
32             existe = Verdadero
33         FinSi
34         f = f + 1
35     FinMientras
36
37     //Mostramos lo solicitado
38     Si pos = -1 Entonces
39         escribir "No se encuentra ese alumno en el arreglo"
40     SiNo
41         escribir "La nota de ", nombre," es ", matriz[pos,1]
42     FinSi
43
44     FinAlgoritmo

```