

TEMA 3:

Fundamentos para la construcción de código a partir del algoritmo

TEMA 3:

Arreglos bidimensionales estáticos numéricos

Objetivo

- El alumno construirá programas utilizando el lenguaje de programación C a través de un análisis y modelado algorítmico previo.

Variables vs. ArregloBidi

```
int uno=2,dos=3,tres=5,cuatro=7;
```

	Memoria física	
uno	2	100
cuatro	7	201
dos	3	302
tres	5	1000
	OCUPADA POR EL SISTEMA OPERATIVO	2000

variables

Direcciones de memoria

```
int matrix[2][2]={{2,3},{5,7}};
```

	Memoria física	
matrix[0][0]	2	1
matrix[0][1]	3	2
matrix[1][0]	5	3
matrix[1][1]	7	4
		1000
	OCUPADA POR EL SISTEMA OPERATIVO	2000

arreglo bidimensional

Direcciones de memoria

Arreglo bidimensional estático

```
int matriz[2][2]= {1,2,3,4};
```

Posiciones
de los
elementos

0,0	0,1
1,0	1,1

Datos que
guardan cada
elemento de
la matriz

1	2
3	4

```
int matriz[3][3]= {{1,2,3},{4,5,6},{7,8,9}};
```

Posiciones
de los
elementos

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

Datos que
guardan cada
elemento de
la matriz

1	2	3
4	5	6
7	8	9

Arreglo bidimensional estático

- Son cuadritos o espacios de memoria RAM consecutivos del mismo tipo de dato al igual que un arreglo unidimensional.
- Todos los cuadritos tienen el mismo nombre.
- Cada cuadrito se identifica por 2 índices de posición en el arreglo, uno indica el renglón y otro la columna.
- Los índices van de 0 hasta un valor anterior a la cantidad reservada para renglones y columnas respectivamente.
- Se declaran indicando su tipo de dato, nombre y tamaño que se cree necesitar de renglones entre corchetes y columnas también entre corchetes que puede ser la misma o diferente.

Arreglo bidimensional estático

- El sistema operativo reserva el arreglo desde que se ejecuta el programa y la memoria es liberada cuando se finaliza.
- No puede cambiarse su dimensión en tiempo de ejecución.
- El arreglo no se opera al mismo tiempo, debe hacerse un cuadrito a la vez aunque sea bidimensional.
- Se emplean 2 ciclos anidados para recorrer cada elemento del arreglo.
- NO puede declararse sin un tamaño ni con una variable sin un valor inicial pero si definida previamente.
- Si empleamos un cuadrito que no reservamos el sistema operativo puede sobrescribirlo o indicar violación de segmento.

Cómo usar un arreglo bidimensional?

- Declaración de un arreglo:

```
int m[3][3];    //3 renglones por 3 columnas  
int mat[10][10]; //10 renglones por 10 columnas
```

```
#define MAX 50  //define MAX con valor de 50  
int arregloBidi[MAX][MAX]; //50x50 por MAX
```

- Impresión de un cuadrado en específico del arreglo:

```
printf("Elemento 0,0: %d", arregloBidi[0][0]);
```

- Lectura de un cuadrado en específico del arreglo:

```
scanf("%d", &arregloBidi[0][0]);
```


Cómo usar un arreglo bidimensional?

- El primer elemento siempre es el índice cero:

```
int m[2][2]; //reserva 4 espacios de 2x2
```

```
m[0][0]= 1; //primer elemento es 0,0 y recibe valor
```

```
m[0][1]= 2; //segundo elemento es 0,1 y recibe valor
```

- Hay arreglos de los diferentes tipos de datos:

```
int arreBidi[3][2]; //reserva 6 espacios enteros
```

```
float matriz[10][10]; //reserva 100 espacios reales
```

```
unsigned char nombre[10][50]; //arreglo de cadenas
```

```
double matrix[200][200]; //reserva 40000 espacios
```

Arreglos bidimensionales inicializados

- Los arreglos bidimensionales estáticos numéricos pueden inicializarse al igual que los unidimensionales colocando los valores entre llaves y separados por comas en orden de renglones.

```
int mat[2][2]= {1,3,5,7};
```

- Es recomendable ser más específicos para evitar advertencias del compilador indicando los valores de cada reglón agrupados a su vez también con llaves.

```
int m[3][2]= {{1,3},{5,7},{9,11}};
```

Arreglo bidimensional en DFD, PSeInt y C

- En **DFD** se declara con el nombre de la variable y **entre paréntesis la cantidad de renglones, columnas** a reservar.
- En **PSeInt** se declara con el nombre de la variable y su tipo de dato, después la palabra reservada **Dimension** variable y **entre corchetes la cantidad de renglones, columnas** a reservar.
- En **lenguaje C** se declara con el tipo, nombre de la variable, **entre corchetes la cantidad de renglones y entre corchetes la cantidad de columnas** a reservar e inicializar si se desea.

Arreglo bidimensional en DFD, PSeInt y C

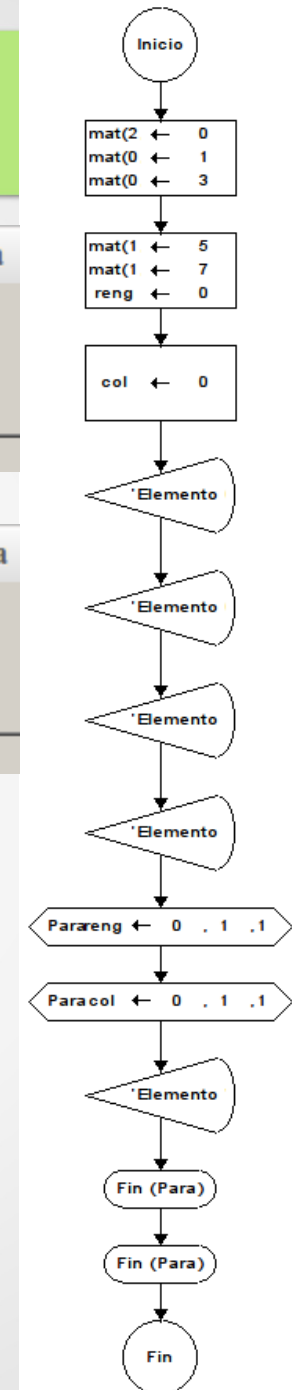
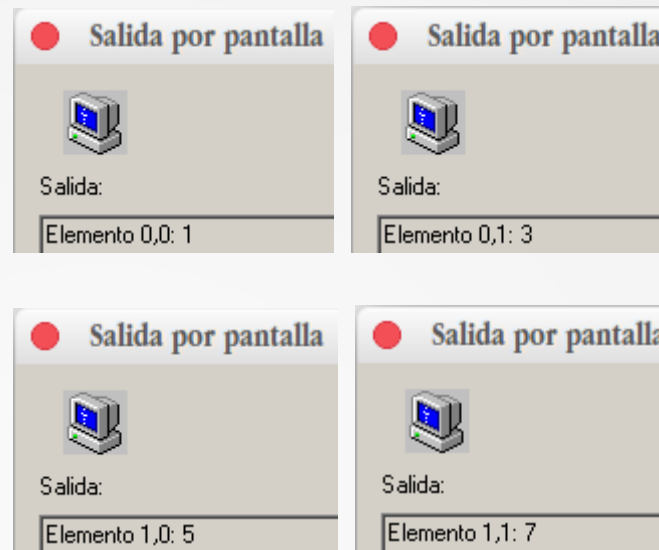
- Puede imprimirse por separado indicando el índice del renglón y el índice de la columna.
- Puede recorrerse fácilmente empleando un **2 CICLOS Para ANIDADOS** o **2 CICLOS for ANIDADOS**
- Un ciclo recorre renglones y se queda en espera en cada uno mientras el otro ciclo recorre sus columnas.
- Se requieren 2 contadores, uno para cada ciclo.
- Aunque sean 2 índices continúa siendo una variable.

Arreglo bidimensional de 2x2 en DFD, PSeInt y C

- Los contadores **reng y col** se declaran como **enteros** o **enteros cortos** o **short** e inicializan de acuerdo a la herramienta y ciclo empleado.
- Los contadores **reng y col** **comienzan a contar en 0** (cero) porque el primer elemento del arreglo es el índice 0.
- Los contadores **reng y col** **se detienen en el índice 1** respectivamente porque el arreglo fue solicitado de 4 elementos (2x2).
- Los contadores **reng y col** **incrementan en 1** respectivamente porque los índices son enteros consecutivos.

Arreglo bidimensional en DFD

- Se recorre empleando 2 ciclos **Para** anidados
- Los contadores **reng** y **col** se declaran como enteros e inicializan
- **reng** y **col** comienzan en 0 (cero)
- **reng** y **col** se detienen en el índice 1
- **reng** y **col** incrementan en 1



PSeInt y ejecución

```
1  Algoritmo arregloBidiNum
2      Definir reng, col Como Enteros
3      Definir arregloBidi como Entero
4      Dimension arregloBidi[2,2]
5
6      arregloBidi[0,0]= 1
7      arregloBidi[0,1]= 3
8      arregloBidi[1,0]= 5
9      arregloBidi[1,1]= 7
10
11      Escribir 'Este algoritmo emplea un arregloBidi entero de 2x2'
12
13      Escribir 'Elemento 0,0: ',arregloBidi[0,0]
14      Escribir 'Elemento 0,1: ',arregloBidi[0,1]
15      Escribir 'Elemento 1,0: ',arregloBidi[1,0]
16      Escribir 'Elemento 1,1: ',arregloBidi[1,1]
17
18      Escribir ''
19      Escribir 'El arregloBidi con ciclo es: '
20      Para reng<-0 Hasta 1 Con Paso 1 Hacer
21          Para col<-0 Hasta 1 Con Paso 1 Hacer
22              Escribir 'Elemento ',reng,', ',col,': ',arregloBidi[reng,col]
23          Fin Para
24      Fin Para
25
26  FinAlgoritmo
```

● ● ● PSeInt - Ejecutando proceso ARREGLOBIDINUM

```
*** Ejecución Iniciada. ***
Este algoritmo emplea un arregloBidi entero de 2x2
Elemento 0,0: 1
Elemento 0,1: 3
Elemento 1,0: 5
Elemento 1,1: 7

El arregloBidi con ciclo es:
Elemento 0, 0: 1
Elemento 0, 1: 3
Elemento 1, 0: 5
Elemento 1, 1: 7
*** Ejecución Finalizada. ***
```


Arreglo bidimensional en C y ejecución

- Se recorre empleando 2 ciclos **for** anidados
- Los contadores **reng y col** se declaran como **short** o enteros cortos
- **reng y col** comienzan en 0 (cero)
- **reng y col** se detienen en el índice 1
- **reng y col** incrementan en 1

```
1  /*programa que muestra arreglo bidi por separado y con ciclos for
2  * hecho por Huicho*/
3
4  #include <stdio.h> //para printf y getchar
5
6  int main(int argc, char* argv[])
7  {
8      //declaración de arreglo bidimensional de 2x2 con valores iniciales
9      int arregloBidi[2][2]= {{1,3},{5,7}};
10     short reng, col; //contadores para recorrer elementos del arregloBidi
11
12     printf("Este programa emplea un arreglo bidimensional de 2x2 \n");
13
14     printf("\n Elemento 0,0: %d", arregloBidi[0][0]);
15     printf("\n Elemento 0,1: %d", arregloBidi[0][1]);
16     printf("\n Elemento 1,0: %d", arregloBidi[1][0]);
17     printf("\n Elemento 1,1: %d", arregloBidi[1][1]);
18
19     printf("\n\n El arreglo con ciclos es: \n");
20
21     //el ciclo debe recorrer los índices 0 y 1 de renglones
22     for (reng=0; reng<=1; reng++)
23     {
24         //el ciclo debe recorrer los índices 0 y 1 de columnas
25         for(col=0; col<=1; col++)
26         {
27             printf("\n Elemento %hd, %hd: %d", reng,col,arregloBidi[reng][col]);
28         }
29     }
30
31     getchar();
32     return 0;
33 }
```



```
/bin/bash
/bin/bash 80x24
Este programa emplea un arreglo bidimensional de 2x2

Elemento 0,0: 1
Elemento 0,1: 3
Elemento 1,0: 5
Elemento 1,1: 7

El arreglo con ciclos es:

Elemento 0, 0: 1
Elemento 0, 1: 3
Elemento 1, 0: 5
Elemento 1, 1: 7
```

Hecho por Huicho :)

Cómo leer un arreglo bidimensional?

- Bloque de código para leer incluyendo ciclos, petición y lectura.
- El tope de cada ciclo es 1 por ser declarado 2 renglones por 2 columnas o pueden ser una o dos variables con la cantidad dada por el usuario.

```
float mat[2][2]; //los 4 datos son reales
short reng, col; //los contadores son enteros cortos

for(reng=0; reng<=1; reng++)
{
    for(col=0; col<=1; col++)
    {
        printf("Ingrese elemento[%hd][%hd]: ", reng, col);
        scanf("%f", &mat[reng][col]);
    }
}
```

Cómo imprimir un arreglo bidimensional?

- Bloque de código para imprimir incluyendo ciclos.
- El tope de cada ciclo es 1 por ser declarado de 2 renglones por 2 columnas o pueden ser una o dos variables con la cantidad dada por el usuario.

```
float mat[2][2]; //los 4 datos son reales
short reng, col; //los contadores son enteros cortos

for(reng=0; reng<=1; reng++)
{
    for(col=0; col<=1; col++)
    {
        printf("\n\t Elemento[%hd][%hd]: %f",reng,col,mat[reng][col]);
    }
}
```

Cómo imprimir un arreglo bidimensional en forma de matriz?

- Bloque de código para imprimir incluyendo ciclos.
- El tope de cada ciclo es 1 por ser declarado de 2 renglones por 2 columnas o pueden ser una o dos variables con la cantidad dada por el usuario.

```
float mat[2][2]; //los 4 datos son reales
short reng, col; //los contadores son enteros cortos

for(reng=0; reng<=1; reng++)
{
    for(col=0; col<=1; col++)
    {
        printf("%5g", mat[reng][col]);
    }
    printf("\n");
}
```

Suma de matrices

```
1  /*programa que muestra suma de 2 matrices de 2x2 con ciclos for
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5
6  int main(int argc, char* argv[])
7  {
8      float a[2][2], b[2][2], c[2][2];
9      short reng, col;
10
11      printf("Programa que suma 2 matrices de 2x2 \n\n");
12
13      printf("Primer matriz a: \n");
14      for(reng=0; reng<=1; reng++) //ciclo para renglones
15      {
16          for(col=0; col<=1; col++) //ciclo para columnas
17          {
18              printf("Ingrese el valor de a[%hd][%hd]: ", reng, col);
19              scanf("%f", &a[reng][col]);
20          }
21      }
22
23      printf("\nSegunda matriz b: \n");
24      for(reng=0; reng<=1; reng++) //ciclo para renglones
25      {
26          for(col=0; col<=1; col++) //ciclo para columnas
27          {
28              printf("Ingrese el valor de b[%hd][%hd]: ", reng, col);
29              scanf("%f", &b[reng][col]);
30          }
31      }
32
33      for(reng=0; reng<=1; reng++) //ciclo para renglones
34      {
35          for(col=0; col<=1; col++) //ciclo para columnas
36          {
37              c[reng][col] = a[reng][col] + b[reng][col];
38              printf("\n c[%hd][%hd]= %f + %f = %f", reng,col,
39                  a[reng][col],b[reng][col],c[reng][col]);
40          }
41      }
```

```
42
43      printf("\n\n Primer matriz a: \n");
44      for(reng=0; reng<=1; reng++) //ciclo para renglones
45      {
46          for(col=0; col<=1; col++) //ciclo para columnas
47          {
48              printf("%5g", a[reng][col]);
49          }
50          printf("\n");
51      }
52
53      printf("\n\n Segunda matriz b: \n");
54      for(reng=0; reng<=1; reng++) //ciclo para renglones
55      {
56          for(col=0; col<=1; col++) //ciclo para columnas
57          {
58              printf("%5g", b[reng][col]);
59          }
60          printf("\n");
61      }
62
63      printf("\n\n La matriz c con la suma: \n");
64      for(reng=0; reng<=1; reng++) //ciclo para renglones
65      {
66          for(col=0; col<=1; col++) //ciclo para columnas
67          {
68              printf("%5g", c[reng][col]);
69          }
70          printf("\n");
71      }
72
73      getchar();
74      getchar();
75      return 0;
76  }
```

Suma de matrices

<code>a[0][0]= 1</code>	<code>a[0][1]= 2</code>
-------------------------	-------------------------

<code>a[1][0]= 3</code>	<code>a[1][1]= 4</code>
-------------------------	-------------------------



<code>b[0][0]= 1</code>	<code>b[0][1]= 2</code>
-------------------------	-------------------------

<code>b[1][0]= 3</code>	<code>b[1][1]= 4</code>
-------------------------	-------------------------



<code>c[0][0]= 2</code>	<code>c[0][1]= 4</code>
-------------------------	-------------------------

<code>c[1][0]= 6</code>	<code>c[1][1]= 8</code>
-------------------------	-------------------------

```
/bin/bash
/bin/bash 80x36
Programa que suma 2 matrices de 2x2

Primer matriz a:
Ingrese el valor de a[0][0]: 1
Ingrese el valor de a[0][1]: 2
Ingrese el valor de a[1][0]: 3
Ingrese el valor de a[1][1]: 4

Segunda matriz b:
Ingrese el valor de b[0][0]: 1
Ingrese el valor de b[0][1]: 2
Ingrese el valor de b[1][0]: 3
Ingrese el valor de b[1][1]: 4

c[0][0]= 1.000000 + 1.000000 = 2.000000
c[0][1]= 2.000000 + 2.000000 = 4.000000
c[1][0]= 3.000000 + 3.000000 = 6.000000
c[1][1]= 4.000000 + 4.000000 = 8.000000

Primer matriz a:
1 2
3 4

Segunda matriz b:
1 2
3 4

La matriz c con la suma:
2 4
6 8
```

Multiplicación de matrices de 2x2

- La multiplicación no es elemento a elemento como la suma.

```
matriz3[0][0]= matriz1[0][0]*matriz2[0][0] + matriz1[0][1]*matriz2[1][0];
```

```
matriz3[0][1]= matriz1[0][0]*matriz2[0][1] + matriz1[0][1]*matriz2[1][1];
```

```
matriz3[1][0]= matriz1[1][0]*matriz2[0][0] + matriz1[1][1]*matriz2[1][0];
```

```
matriz3[1][1]= matriz1[1][0]*matriz2[0][1] + matriz1[1][1]*matriz2[1][1];
```

- $19 = 1 * 5 + 2 * 7 \rightarrow 19 = 5 + 14$

matriz1

0,0	0,1
1,0	1,1

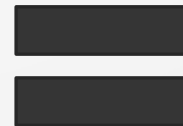
1	2
3	4



matriz2

0,0	0,1
1,0	1,1

5	6
7	8



matriz3

0,0	0,1
1,0	1,1

19	22
43	50

Preguntar al usuario el tamaño

- Un arreglo estático no puede reservar el tamaño exacto dado en tiempo de ejecución.
- Puede definirse en el código un tamaño suficientemente grande y preguntar al usuario el número de renglones y columnas que desea o la dimensión dentro del rango previamente reservado.

```
#define MAX 50 //define MAX con valor de 50  
int arregloBidi[MAX][MAX]; //50 reng x 50 col x MAX
```

- Mientras el programa esté en ejecución el sistema operativo no puede reasignar el espacio solicitado para el arreglo se encuentre o no ocupado en su totalidad.


```

1  /*programa que lee y muestra arreglo bidimensional de mxn con ciclos for
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5
6  #define MAX 100
7
8  int main(int argc, char* argv[])
9  {
10     float arregloBidi[MAX][MAX]; //arreglo bidimensional para reales
11     short reng, col; //contadores que recorren renglones y columnas
12     short renglones; //guarda la cantidad de renglones del usuario
13     short columnas; //guarda la cantidad de columnas del usuario
14
15     printf("Programa que lee e imprime arreglo bidimensional de mxn \n\n");
16     printf("Ingresa la cantidad de renglones: ");
17     scanf("%hd", &renglones);
18     printf("Ingresa la cantidad de columnas: ");
19     scanf("%hd", &columnas);
20
21     printf("\n Matriz a: \n\n");
22     for(reng=0; reng<=renglones-1; reng++) //ciclo para renglones
23     {
24         for(col=0; col<=columnas-1; col++) //ciclo para columnas
25         {
26             printf("Ingrese el valor del elemento[%hd][%hd]: ", reng, col);
27             scanf("%f", &arregloBidi[reng][col]);
28         }
29     }
30
31     printf("\n Matriz a: \n\n");
32     for(reng=0; reng<=renglones-1; reng++) //ciclo para renglones
33     {
34         for(col=0; col<=columnas-1; col++) //ciclo para columnas
35         {
36             printf("%5g", arregloBidi[reng][col]);
37         }
38         printf("\n");
39     }
40
41     getchar();
42     getchar();
43     return 0;
44 }

```

- El tope del ciclo de renglones es `conde<=renglones-1` o `conde<renglones`
- El tope del ciclo de columnas es `conde<=columnas-1` o `conde<columnas`
- El tamaño máximo del arreglo es de 100 renglones y 100 columnas, desde el índice 0 hasta el 99 para cada uno
- **Es muy importante avisar al usuario la cantidad máxima de elementos y validar que la cantidad ingresada se encuentre en el rango permitido.**

Preguntar al usuario el tamaño



```
/bin/bash
Programa que lee e imprime arreglo bidimensional de mxn

Ingresa la cantidad de renglones: 3
Ingresa la cantidad de columnas: 3

Matriz a:

Ingresa el valor del elemento[0][0]: 1
Ingresa el valor del elemento[0][1]: 3
Ingresa el valor del elemento[0][2]: 5
Ingresa el valor del elemento[1][0]: 7
Ingresa el valor del elemento[1][1]: 9
Ingresa el valor del elemento[1][2]: 11
Ingresa el valor del elemento[2][0]: 13
Ingresa el valor del elemento[2][1]: 15
Ingresa el valor del elemento[2][2]: 17

Matriz a:

    1    3    5
    7    9   11
   13   15   17
```

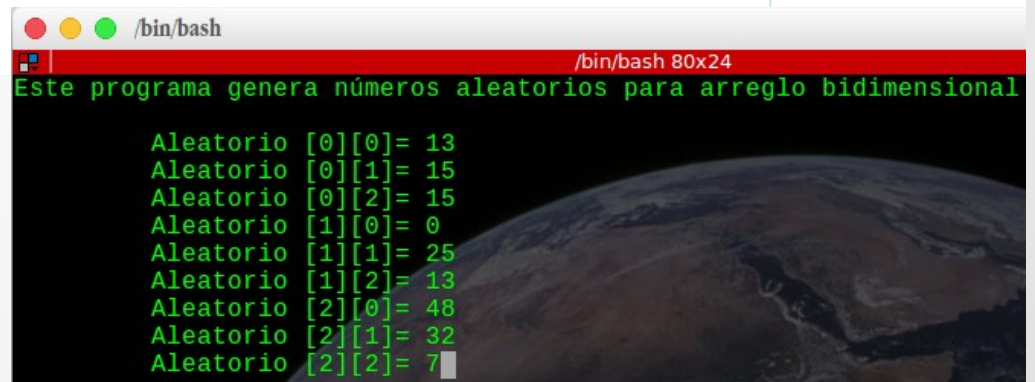
Operaciones sobre arreglos bidimensionales

OPERACIÓN	VARIABLE	ARREGLO BIDIMENSIONAL
Asignación de valor	<code>x=0;</code>	<code>arregloBidi[0][0]= 0;</code>
Comparación	<code>if (x==0)</code>	<code>if (arregloBidi[0][0]==0) neutros++; else if (arregloBidi[0][0]>0) positivos++; else if (arregloBidi[0][0]<0) negativos++;</code>
Acumulador para multiplicar	<code>x= x * 5;</code>	<code>aBidi[0][0]= aBidi[0][0] * 5;</code>
Acumulador para sumar	<code>x= x+conde;</code>	<code>aBidi[0][0]= aBidi[0][0]+conde;</code>

Números aleatorios

- Incluir `time.h`
- Incluir `stdlib.h`
- `srand(time(NULL)) ;`
genera el número semilla que cambia con el tiempo y a partir del cual se obtiene una serie de pseudoaleatorios.
- `rand() % limite`
genera números enteros en el rango de 0 hasta el valor de *limite-1*

```
1  /*programa para asignar pseudoaleatorios a un arreglo bidimensional
2  * hecho por huicho*/
3
4  #include <stdio.h> //para printf, scanf y getchar
5  #include <time.h> //para time
6  #include <stdlib.h> //para srand, rand
7
8  int main(int argc, char* argv[])
9  {
10     int aleatorioBidi[3][3];
11     short reng, col;
12
13     srand(time(NULL)); //genera número semilla a partir del tiempo
14
15     printf("Este programa genera n%cmoros aleatorios para arreglo bidimensional \n", 163);
16     for(reng=0; reng<=2; reng++)
17     {
18         for(col=0; col<=2; col++)
19         {
20             aleatorioBidi[reng][col]= rand() % 50; //genera un aleatorio desde 0 hasta 49
21             printf("\n\t Aleatorio [%hd][%hd]= %d", reng, col, aleatorioBidi[reng][col]);
22         }
23     }
24
25     getchar();
26     return 0;
27 }
```



```
/bin/bash
Este programa genera números aleatorios para arreglo bidimensional

Aleatorio [0][0]= 13
Aleatorio [0][1]= 15
Aleatorio [0][2]= 15
Aleatorio [1][0]= 0
Aleatorio [1][1]= 25
Aleatorio [1][2]= 13
Aleatorio [2][0]= 48
Aleatorio [2][1]= 32
Aleatorio [2][2]= 7
```