H.M. Deitel/P.I. Deitel, Como programar en C/C++ Ed. Prentice

Problema 6.19 página 249

6.19 Escriba un programa en C que simule el tirar dos dados. El programa deberá utilizar **rand** para tirar el primer dado, y después volverá a utilizar **rand** para tirar el segundo. La suma de los dos valores deberá entonces ser calculada. *Nota:* en vista de que cada dado puede mostrar un valor entero de 1 a 6, entonces la suma de los dos valores variará desde 2 hasta 12, siendo 7 la suma más frecuente y 2 y 12 las menos frecuentes. En la figura 6.23 se muestran las 36 combinaciones posibles de los dos dados. Su programa deberá tirar 36,000 veces los dos dados. Utilice un arreglo de un subíndice para llevar cuenta del número de veces que aparece cada suma posible. Imprima los resultados en un formato tabular. También, determine si los totales son razonables, es decir, existen seis formas de llegar a un 7, por lo que aproximadamente una sexta parte de todas las tiradas deberán ser 7.

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

Problema 7.25 página 313

partición.

7.25 (*Atravesar laberintos*). La siguiente cuadrícula de unos y de ceros es un arreglo de doble subíndice, que representa un laberinto.

```
1 1 1 1 1 1 1 1 1 1 1 1
1 0 0 0 1 0 0 0 0 0 0 1
0 0 1 0 1 0 1 1 1 1 0 1
1 1 1 0 1 0 0 0 0 1 0 1
1 0 0 0 0 1 1 1 0 1 0 0
1 1 1 1 0 1 0 1 0 1 0 1
1 0 0 1 0 1 0 1 0 1 0 1
1 1 0 1 0 1 0 1 0 1 0 1
1 0 0 0 0 0 0 0 0 1 0 1
1 1 1 1 1 1 0 1 1 1 0 1
1 0 0 0 0 0 0 1 0 0 0 1
1 1 1 1 1 1 1 1 1 1 1 1
```

Los unos representan los muros del laberinto, y los ceros representan cuadros en las trayectorias posibles a través del mismo.

Existe un algoritmo simple para caminar a través de un laberinto que garantiza encontrar la salida (suponiendo que una exista). Si no existe salida, usted llegará de regreso a la posición inicial. Coloque su mano derecha sobre el muro a su derecha y empiece a caminar hacia adelante. No retire nunca su mano de la pared. Si el laberinto gira a la derecha, usted sigue el muro a la derecha. Siempre que no retire su mano de la pared, de forma eventual llegará a la salida del laberinto. Pudiera existir una trayectoria más corta de la que haya tomado, pero está garantizado de que saldrá del laberinto.

Escriba la función recursiva `mazeTraverse` para caminar a través del laberinto. La función deberá recibir como argumentos un arreglo de 12 por 12 caracteres, que representa el laberinto, y la posición inicial del laberinto. Conforme `mazeTraverse` intenta localizar la salida del laberinto, deberá colocar el carácter `x` en cada uno de los cuadros de la trayectoria. Después de cada movimiento la función deberá mostrar el laberinto, para que el usuario pueda observar conforme resuelve el laberinto.

Problema 7.26 página 313

7.26 (*Cómo generar laberintos en forma aleatoria*). Escriba una función `mazeGenerator`, que toma como argumento un arreglo de 12 por 12 de doble subíndice, y que produce en forma aleatoria un laberinto. La función también deberá proporcionar las posiciones iniciales y finales del mismo. Pruebe su función `mazeTreverse`, del ejercicio 7.25, utilizando varios laberintos generados al azar.