Sopa de Letras

En este ejercicio vamos a implementar un resolutor de sopas-donut de letras. Dada una sopa de letras (una matriz de letras) y una secuencia de palabras a buscar, el resolutor buscará cada una de esas palabras en la sopa. Para las que encuentre, indicará la localización. Para las que no estén en la sopa indicará que no están presentes. Por ejemplo, supongamos que tenemos la siguiente sopa (numeramos filas y columnas para manejar después coordenadas):

	0	1	2	3	4	5	6	7	8
0	C	O	A	B	C	D	B	A	R
1	E	K	L	M	N	O	P	Q	R
2	H	T	A	V	I	O	N	O	R
3	C	G	R	T	U	I	T	X	B
4	O	R	O	H	F	O	V	A	Z
5	C	M	P	P	M	E	V	A	N

y la secuencia de palabras a buscar: COCHE AVION BARCO MOTO PATINES.

Podemos observar que COCHE aparece en las coordenadas (5,0), en dirección norte; MOTO en las coordenadas (5,4) en dirección noreste; mientras que PATINES no aparece en la sopa. Pero nuestras sopas-donut esconden una sorpresa: los bordes izquierdo y derecho están unidos, así como el borde inferior y superior, es decir, tienen forma de donut. De este modo, la palabra BARCO se encuentra en la posición (0,6) en la dirección este. Para codificar las 8 posibles direcciones, en vez de norte, noreste, etc., vamos a utilizar vectores de incremento, i.e., vectores que sumados reiteradamente a una posición de la sopa proporcionan una secuencia de letras en una dirección en la sopa. Así por ejemplo, localizamos COCHE en las coordenadas (5,0), en la dirección (-1,0): la posición (5,0) contiene la primera letra C y si vamos sumando a esa posición el vector de incremento (-1,0) obtenemos la posición (4,0) que tiene la letra O, (3,0) que tiene la C, etc. La palabra MOTO se localiza en la sopa en la posición (5,4), dirección (-1,1). Tendremos que tener en cuenta que estos incrementos operan sobre una forma de donut, i.e., cuando se llega a un borde, se continúa por el opuesto. La codificación de direcciones es la siguiente:

	↑	(-1,-1)	(-1,0)	(-1,1)
←	→	(0,-1)		(0,1)
	↓	(1,-1)	(1,0)	(1,1)

Obsérvese que tanto las posiciones como las direcciones se pueden representar mediante pares de enteros (x,y). Para la sopa tendremos las dimensiones alto y ancho, y un array de strings, para representar las filas de la sopa de letras.

Observemos que la matriz de letras se representa como un array de strings (en vez de utilizar un array bidimensional de chars), donde cada string representa una fila de la sopa de letras. Esto facilita la inicialización, la lectura de archivo, etc, y esta representación permite un manejo similar al array bidimensional, ya que el tipo string permite el acceso a componentes de manera similar a un array.

El algoritmo de resolución que codificaremos es como sigue: cada palabra a localizar en la sopa se busca en cada una de las coordenadas de la matriz, en cada una de las direcciones posibles, hasta encontrarla (sí está) o determinar que no está. Si una palabra aparece más de una vez en la sopa, basta con obtener una de las apariciones. Para ello codificar este algoritmo implementaremos las siguientes funciones (se indican los parámetros y su tipo, pero no la forma de paso de parámetros que debe determinarla el alumno):

- Par[] dirs(): devuelve un array con las 8 direcciones codificadas como pares, según se ha explicado. Pista: estas direcciones son de la forma (i,j) con i recorriendo los valores [-1,0,1] y j recorriendo los valores [-1,0,1], a excepción del par (0,0), que no codifica ninguna dirección.

- bool compruebaPosDir(Sopa s, string pal, Par pos, Par dir): determina si la palabra pal se encuentra en la sopa s, en la posición pos en la dirección dir.

  Por ejemplo, con la sopa del ejemplo, la palabra ”MOTO”, la posición (5,4) y la dirección (-1,1) devolvería true. Con este ejemplo y otros similares, pueden hacerse algunas pruebas para comprobar el correcto funcionamiento de la función.

- bool buscaDir(Sopa s, string pal, Par pos, Par dir): utilizando el función anterior, determina si la palabra pal se encuentra en la sopa s, en la posición pos, en alguna de las direcciones posibles, proporcionadas por la función dirs. Si es así, devolverá true y en dir devolverá la dirección en la que se encuentra; en otro caso devolverá false.

  Por ejemplo, para la palabra ”MOTO”, en la posición (5,4) debe devolver true y asignar a dir el valor (-1,1).

- bool buscaPal(Sopa s, string pal, Par pos, Par dir): utilizando la función anterior determina si la palabra pal está en la sopa s. Si es así, devolverá true, así como la posición pos y la dirección dir donde la ha localizado; en caso contrario, devolverá false.

  Por ejemplo, para la palabra ”MOTO” devolverá true, asignará (5,4) al parámetro pos y (-1,1) al parámetro dir.

- void resuelve(Sopa s, string [] pals): dada una sopa s y un array de palabras pals, para cada una de ellas determina utilizando la función anterior si está o no en la sopa, indicando posición y dirección en caso afirmativo. Para la sopa y las palabras de nuestro ejemplo, la salida sería:

  Encontrada COCHE en Posicion (5,0) direccion (-1,0)
  Encontrada AVION en Posicion (2,2) direccion (0,1)
  Encontrada BARCO en Posicion (0,4) direccion (0,1)
  Encontrada MOTO en Posicion (5,4) direccion (-1,1)
  No encontrada PATINES

- void leeSopa(Sopa s, string[] pals): esta función permitirá leer de un archivo sopa.txt una sopa de letras y la lista de la palabras a buscar, y las devolverá en s y pals respectivamente. Modificar la función Main para leer de archivo la sopa en vez de inicializarla ad-hoc tal como se ha presentado arriba.

  El archivo sopa.txt contendrá en este orden: el alto y el ancho de la sopa, las filas de la sopa (una por línea), el número de palabras a buscar, las palabras en cuestión (una por línea). Así, para nuestra sopa de ejemplo el archivo sería:

```
6
9
COABCDBAR
EKLMNOPQR
HTAVIONOR
CGRTUITXB
OROHFOVAZ
CMPPMEVAN
5
COCHE
AVION
BARCO
NAVE
MOTO
```