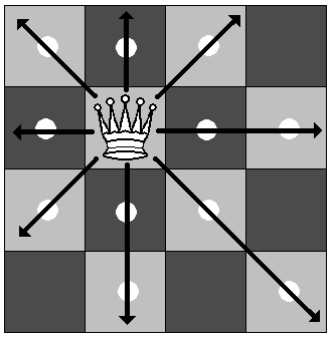
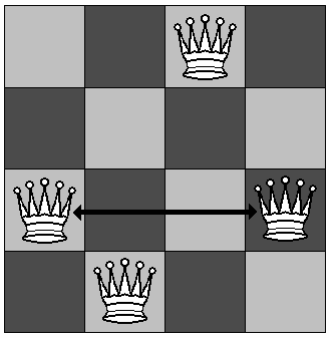
El Problema de las Ocho Reinas

Ángel Aguirre Pérez,
Profesor de Matemáticas del I.E.S. "Benedicto Nieto", de Pola de Lena (Asturias)
angelap@educastur.princast.es

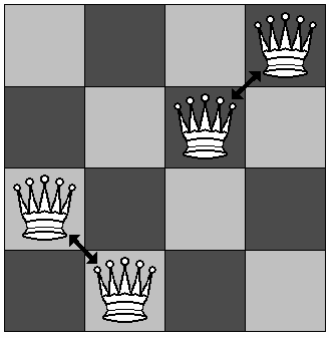
Exploraremos la potencia de la calculadora ClassPad 300 en el cálculo con listas. Vamos a resolver el problema de las ocho reinas: consiste en colocar ocho reinas sobre un tablero de tal manera que ninguna esté amenazada por cualquiera de las restantes.



Quizá convenga recordar que, en el juego del ajedrez, la reina amenaza a aquellas fichas que se encuentren en su misma fila, columna o diagonales. Por comodidad y concisión vamos a hacer todas nuestras consideraciones para un tablero 4 x 4, cuatro filas y cuatro columnas. Posteriormente generalizaremos para un tablero de dimensiones arbitrarias n x n.

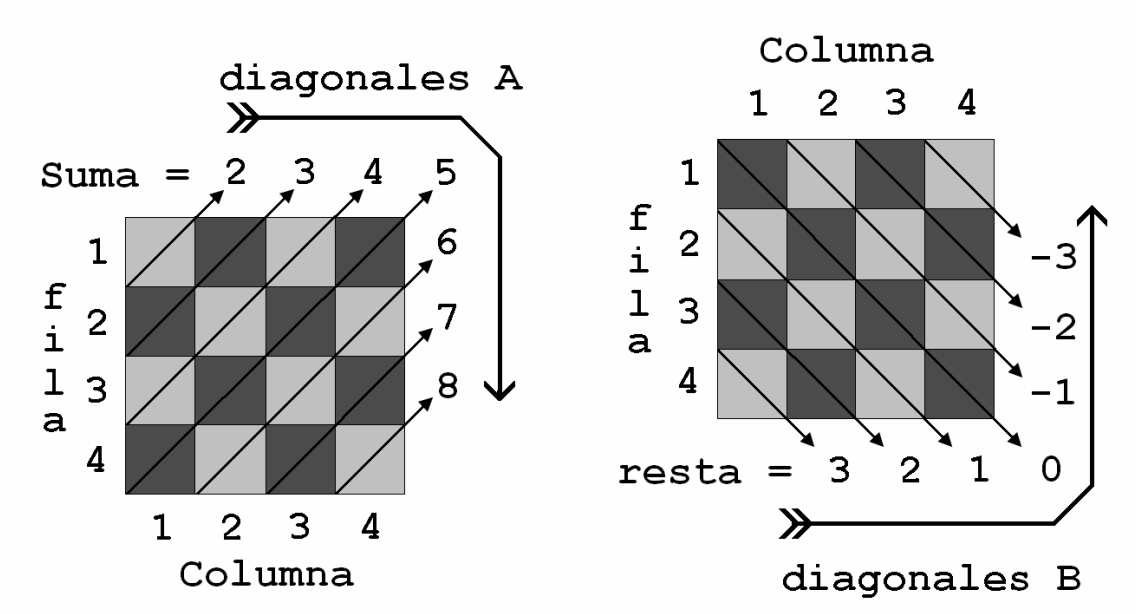


En primer lugar debemos buscar una notación para poder representar la posición de las reinas en el tablero. Vamos a utilizar una lista (un vector) para denotar la posición de las reinas. Cada componente de esta lista hace referencia a una columna del tablero, la primera componente a la primera columna, la segunda componente a la segunda, etc. El valor de la componente nos indica la fila en la que se encuentra la reina en esa columna. Por ejemplo, la lista {2, 1, 4, 2}, representa la posición de la figura de la derecha.



A la vista de esto, parece obvio que nuestra solución debe contener números distintos, para que las reinas ocupen filas distintas. La solución ha de ser una permutación de los números 1, 2, 3 y 4. Esta elección garantiza que no existan amenazas por presencia de otra reina en esa misma fila. Sin embargo, no evita las amenazas debidas a reinas situadas en las mismas diagonales. Por ejemplo la disposición {3, 4, 2, 1}, representada en la figura anterior, presenta dos amenazas en diagonal. Debemos, por tanto, hacer un algo-

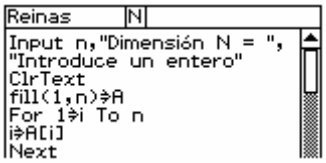
ritmo que nos permita encontrar todas las posibles amenazas. Para ello clasificamos previamente las diagonales en dos tipos: tipo A y tipo B. Según se puede ver en la siguiente figura:



Dos casillas o escaques pertenecen a la misma "diagonal A" si la suma de su fila y su columna da como resultado el mismo número. Del mismo modo, dos casillas distintas pertenecen a la misma "diagonal B" si la resta de su fila menos su columna es idéntica.

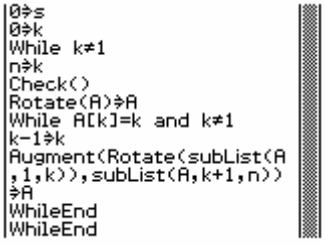
Nuestro proyecto se compone de dos partes: un programa principal que se llama **Reinas** y una subrutina de nombre **Check**. En el primero se generaran todas las permutaciones posibles mediante un algoritmo muy sencillo, aunque no en orden lexicográfico. El algoritmo ha sido tomado del libro "Combinatorial Algorithms" de Reingold, Deo y Nievergelt. En la segunda se comprobará si es solución y se mostrará por pantalla.

El programa comienza preguntando las dimensiones del tablero mediante una ventana de entrada de datos (Input). Se limpia la pantalla. Se asigna ese valor a la variable ***n*** y se crea una lista ***A*** que contiene los ***n*** primeros números naturales. Por ejemplo: si ***n*** vale 4, la lista ***A*** es {1, 2, 3, 4}.



A continuación, comienza el algoritmo de generación de permutaciones. La variable ***s*** contiene el número de soluciones halladas hasta el momento; inicialmente vale cero.

La generación de las permutaciones se hace a través del código mostrado en la figura de la derecha. La llamada a la rutina de comprobación se hace a través del comando **Check()**.



Se utiliza la función **Rotate,** que devuelve una lista en la que los elementos han sido rotados hacia la derecha o izquierda un cierto número de posiciones. La opción por defecto es una posición hacia la derecha; por ejemplo, cuando la lista ***A*** es {1, 2, 3, 4}, Rotate(***A***) da como resultado la lista {4, 1, 2, 3}.

También se utiliza la función **subList,** que extrae una parte concreta de una lista y la función **Augment,** que anexiona una lista con otra. Siguiendo con el ejemplo anterior, con **subList**(***A***, 2, 3) se obtiene {2, 3} y **Augment**(***A***,{5, 6}) devuelve la lista {1, 2, 3, 4, 5, 6}.

Para finalizar, se escribe el número total de soluciones halladas.

```
PrintNatural s,"Número de soluciones"
```

La subrutina **Check** toma cada pareja de fichas y comprueba, como hemos explicado anteriormente, si se amenazan. Es decir, si la suma de su fila y columna o la resta de su fila menos la columna son iguales. En tal caso, regresa al programa principal sin hacer nada. Si ninguna pareja se amenaza, aumenta uno el valor de ***s*** y escribe la solución por pantalla.

```
Check N
For 1→i To n-1
For i+1→j To n
If A[i]+i=A[j]+j or A[i]-i=A[j]-j
Then
Return
IfEnd
Next
Next
s+1→s
print A
Return
```

La ejecución del programa **Reinas** puede apreciarse en las siguientes pantallas: