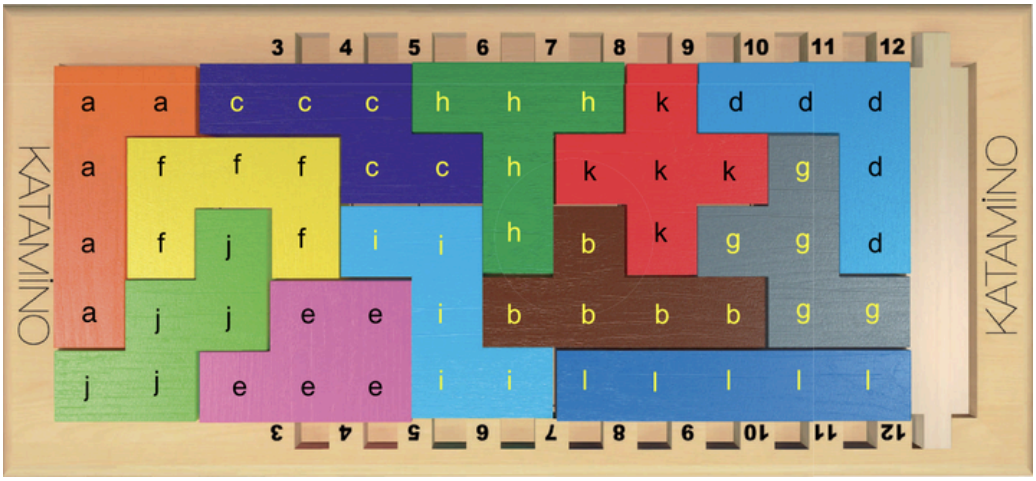
KATAMINO

El Katamino es un juego que contiene 12 piezas que hay que acomodar ocupando todos los espacios del tablero. El tablero es de 5 filas y entre 3 y 12 columnas. En este trabajo vamos a implementar un programa que nos ayude a encontrar las soluciones posibles.



Archivos base

El archivo katamino.pl incluye a piezas.pl. Deben estar situados en el mismo directorio.

```
% cat katamino.pl
:- use_module(piezas).


% swipl katamino.pl
?-
```

Piezas

Las piezas van a estar identificadas por una letra a,....,l . El predicado nombrePiezas/1 que nos indica los nombres de las piezas.

```
?- nombrePiezas(L).
L = [a, b, c, d, e, f,  g,  h,  i,  j,  k,  l].
```

También disponemos del predicado pieza(+Identificador,-Forma) que nos permitirá obtener cada pieza como una matriz como lista de filas.

```
?- pieza(a, F).
F = [[a,a],
[a,_],
[a,_],
[a,_]] .
```

```
?-  pieza(b, F).
F =  [[_, b, _, _],
      [b, b, b, b]] .
```

En cada matriz, hay elementos instanciados con el identificador de la pieza donde la pieza ocupa espacio y valores no instanciados donde la pieza no ocupa espacio.

El predicado pieza/2 nos va a enumerar también todas las posibles orientaciones de la pieza. La enumeración es sin repetidos.

```
?- pieza(e, F).
F = [[_, e, e],
     [e, e, e]] ;

F = [[e, e],
     [e, e],
     [_, e]] ;
...
```

Mostrar

Se dispone de mostrar(+M) que imprime en pantalla tanto una pieza como un tablero. Cada ficha tiene un color/patrón asignado.

```
?- pieza(a, P), mostrar(P).
```

```
P = [[a, a], [a, _], [a, _], [a, _]] ;
```

```
P = [[a, _, _, _], [a, a, a, a]] ;
```

Sublista

El predicado sublista(+Descartar, +Tomar, +L, -R) es cierto cuando R es la sublista de longitud Tomar luego de descartar los primeros Descartar elementos de L. El siguiente ejemplo muestra la única solución posible para el caso de descartar 2 elementos y tomar 3 de la lista [a,b,c,d,e,f].

```
?- sublista(2, 3, [a,b,c,d,e,f], R).
R = [c, d, e].
```

Tablero

El predicado tablero(+K, -T) genera un tablero vacío de K > 0 columnas. El tablero será una matriz de 5×K representada como lista de filas. Cada casilla del tablero debe ser una variable no instanciada distinta.

```
?- tablero(3, T) T
= [[_,_,_],
   [_,_,_],
   [_,_,_],
   [_,_,_],
   [_,_,_]].
```

Tamaño

Dada una matriz M representada como lista de filas. El predicado tamaño(+M, -F, -C) será verdadero cuando M tenga F filas y C columnas. Este predicado va a ser usado tanto para piezas como para tableros.

```
?- tablero(3, T),      tamaño(T,  F,  C).
 F=5, C=3.
```

```
?- pieza(e, E),      tamaño(E,  F,   C).

 F=2, C=3;
 F=3, C=2;
 F=2, C=3; ...
```

Coordenadas

Dado un tablero T de 5 × K, coordenadas(+T, -IJ) es verdadero para todo par IJ que sea una coordenada de elementos del tablero. Los índices irán de 1...5 y 1...K donde (1,1) es la esquina superior izquierda. Puede dar los elementos en cualquier orden pero sin repetidos.

```
?- tablero(3, T),       coordenadas(T, IJ).
IJ = (1,1) ;
IJ = (1,2);
...
IJ = (5,3).
```

K-Piezas

Debemos cubrir todos los espacios del tablero de 5 filas y K columnas. Para esto sabemos que vamos a necesitar exactamente K piezas ya que todas las piezas ocupan 5 espacios. kPiezas(+K, -PS) es verdadero cuando PS es una lista de longitud K de identificadores de piezas. No repite soluciones (ni permutaciones).

```
?-   kPiezas(3,   PS).
PS = [a,b,c] ;
PS = [a,b,d] ;
PS = [a,b,e] ;
PS = [a,b,f] ;
...
PS = [b,c,d] ;
...

?- kPiezas(12,   PS).
PS = [a ,b, c, d, e, f, g, h, i, j, k, l].
```

SeccionTablero

Para ubicar una pieza vamos a seleccionar una sección del tablero. El predicado seccionTablero(+T, +ALTO, +ANCHO, +IJ, ?ST) será verdadero cuando ST sea una sección de tamaño ATLO x ANCHO del tablero T a partir de la coordenada IJ.

```
?- tablero(3, T),       seccionTablero(T,   3,   2,   (1,2),      ST).

 T  =[[_C11, _C12, _C13],
     [_C21, _C22, _C23],
     [_C31, _C32, _C33],
     [_C41, _C42, _C43],
     [_C51, _C52, _C53]],

 ST =    [[_C12, _C13],
          [_C22, _C23],
          [_C32, _C33]].
```

Si no hay posible sección de tablero de este tamaño entonces el predicado da False.

```
?- tablero(3, T),       seccionTablero(T,   3,   3,   (1,2),    ST).
false.
```

Con esto sabremos si una pieza encaja.

```
?- tablero(3, T),    pieza(e, E),  tamaño(E, F, C),  seccionTablero(T,  F,  C, (1,1),  E).
T   =  [[_,e,e],
        [e,e,e],
        [_,_,_],
        [_,_,_],
        [_,_,_]] ;
...
T   =  [[e,e,_],
        [e,e,_],
        [_,e,_],
        [_,_,_],
        [_,_,_]] ;
...
```

El predicado mostrar/1 puede usarse para ver el estado del tablero.

```
?- tablero(3, T), pieza(e, E), tamaño(E, F, C), seccionTablero(T, F, C, (1,1), E), mostrar(T).
```

```
T = [[_A, e, e], [e, e, e], [_, _, _], [_, _, _], [_, _, _]],
E = [[_A, e, e], [e, e, e]],
F = 2,
C = 3 ;
```

```
T = [[e, e, _], [e, e, _], [_A, e, _], [_, _, _], [_, _, _]],
E = [[e, e], [e, e], [_A, e]],
F = 3,
C = 2 ;
```

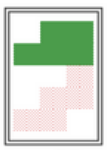
Ubicar pieza

Dado un tablero, ubicarPieza(+Tablero, +Identificador) genera todas las posibles ubicaciones de la pieza dada en el tablero. El Tablero va a ser una matriz con estructura instanciada pero con casillas posiblemente no instanciadas para indicar casillas libres. Soluciones similares via rotaciones de tablero se consideran distintas.

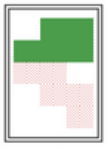
```
?- tablero(3, T),    ubicarPieza(T, e).

T   = [[_,e,e],                    T  =  [[_,_,_],
       [e,e,e],                           [_,e,e],
       [_,_,_],                           [e,e,e],
       [_,_,_],                           [_,_,_],
       [_,_,_]]      ;                    [_,_,_]]      ;
...                                 ...
```

```
?- tablero(3, T), ubicarPieza(T, e), ubicarPieza(T, j), mostrar(T).
```



```
T = [[_, e, e], [e, e, e], [_, _, j], [_, j, j], [j, j, _]] ;
```



```
T = [[_, e, e], [e, e, e], [j, j, _], [_, j, j], [_, _, j]] ;
```

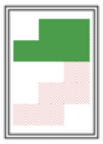
Ubicar piezas

Similar a ubicarPieza/2, el predicado ubicarPiezas(+Tablero, +Poda, +Identificadores) lista todas las posibles opciones de ubicar todas las piezas mencionadas en Identificadores. El valor de Poda indica qué estrategia usar para podar el espacio de búsqueda. El Tablero va a ser una matriz con estructura instanciada pero con casillas posiblemente no instanciadas para indicar casillas libres.

Al momento contamos con una única estrategia sinPoda que no poda. Y equivale a ubicar una pieza tras otra sin hacer chequeos adicionales entre una y otra.
El predicado poda(+Poda, +Tablero) es verdadero si el tablero satisface la poda. Por ahora el único valor para Poda será sinPoda.

```
poda(sinPoda, _).
```

```
?- tablero(3, T), ubicarPiezas(T, sinPoda, [e, j]), mostrar(T).
```



```
T = [[_, e, e], [e, e, e], [_, _, j], [_, j, j], [j, j, _]] ;
```

Llenar Tablero

Dada la cantidad de columnas de un tablero, llenarTablero(+Poda, +Columnas, -Tablero) enumera todas las formas distintas de llenar un tablero con la cantidad de columnas (y piezas) indicada. El valor de Poda determina la estrategia de poda que ubicarPiezas/3 va a usar. Notar que todas las casillas del tablero van a estar instanciadas en Tablero. Nuevamente mostrar/1 puede usarse para ver el estado del tablero.

```
?- llenarTablero(sinPoda, 3, T), mostrar(T).
```

```
T = [[a, a, b], [a, b, b], [a, h, b], [a, h, b], [h, h, h]] ;
```

```
T = [[h, h, h], [b, h, a], [b, h, a], [b, b, a], [b, a, a]] ;
```

Es posible que para valores K>=5 este predicado tarde más de lo que se espera.

Medición

El predicado cantSoluciones/3 calcula cuántas soluciones distintas hay para un tablero con la cantidad de columnas dada de la siguiente manera.

```
cantSoluciones(Poda, Columnas, N) :-
  findall(T, llenarTablero(Poda, Columnas, T), TS),
  length(TS, N).
```

Se muestra cuánto tarda el predicado cantSoluciones/3 en dar respuesta para las distintas cantidades de columnas con el predicado time/1, que mide el tiempo de ejecución. Mediciones para K=3 y K=4.

Optimización

Actualmente, cada vez que se ubica una pieza puede que el tablero quede de tal forma que nunca va a poder completarse. Ej: Si alguno de los grupos de lugares libres no es divisible por 5. La estrategia de poda podaMod5 asegura que ubicarPiezas(podaMod5, ES, T) realiza este chequeo a medida que va ubicando cada pieza.

El predicado todosGruposLibresModulo5(+Tablero) recibe un Tablero que va a ser una matriz con estructura instanciada pero con casillas posiblemente no instanciadas para indicar casillas libres. Es verdadero si las casillas libres agrupadas son todas de tamaño módulo 5.

Se dispone del predicado agrupar(+L, -G) que dada una lista de coordenadas (I,J), es verdadero si G es una lista de grupos de dichas posiciones. Dos casillas se agrupan si comparten un lado.

?- agrupar([(1,1), (2,2), (2,1), (3,3), (4,4), (3,4)], G).

G = [[(1, 1), (2,2), (2, 1)], [(3, 3), (4, 4), (3, 4)]].