Segunda práctica (Programación Lógica Pura)

Alejandro Náger Fernández-Calvo (c200070)

Código empleado en la práctica

A continuación, se encuentra el código comentado con un lenguaje natural para dar una explicación de cada fragmento del código lo mejor posible.

```
:- module(_,_,[classic, assertions]).
:- use_module(library(lists)).

author_data('Nager','Fernandez-Calvo','Alejandro','C200070').

%tableros
board1([cell(pos(1 ,1),op(*,-3)),
    cell(pos(1,2),op(-,1)),
    cell(pos(1,3),op(-,4)),
    cell(pos(1,4),op(-,555)),
    cell(pos(2,1),op(-,3)),
    cell(pos(2,2),op(+,2000)),
    cell(pos(2,3),op(*,133)),
    cell(pos(2,4),op(-,444)),
    cell(pos(3,1),op(*,0)),
    cell(pos(3,2),op(*,155)),
    cell(pos(3,3),op(//,2)),
    cell(pos(3,4),op(+,20)),
    cell(pos(4,1),op(-,2)),
    cell(pos(4,2),op(-,1000)),
    cell(pos(4,3),op(-,9)),
    cell(pos(4,4),op(*,4))]).

board2([cell(pos(1,1),op(*,-3)),
    cell(pos(1,2),op(-,1)),
    cell(pos(2,1),op(-,3)),
    cell(pos(2,2),op(+,2000))]).

board3([cell(pos(1,1),op(*,-3)),
    cell(pos(1,2),op(-,1)),
    cell(pos(1,3),op(-,1)),
    cell(pos(2,1),op(-,3)),
    cell(pos(2,2),op(+,2000)),
    cell(pos(2,3),op(-,3)),
    cell(pos(3,1),op(+,2000)),
```

```
    cell(pos(3,2),op(+,2000)),
    cell(pos(3,3),op(+,2000))]).

%movimiento_tableros
dirs1([dir(n,5), dir(s,6), dir(e,7), dir(o,4)]).
dirs2([dir(n,2), dir(s,2), dir(e,2), dir(o,6)]).

%efectuar_movimiento/3
%cada movimiento desplazara la posicion en una direccion cambiando los
valores de la fila y columna
efectuar_movimiento(pos(Row, Col), n, pos(NewRow, Col)) :- NewRow is
Row - 1.
efectuar_movimiento(pos(Row, Col), s, pos(NewRow, Col)) :- NewRow is
Row + 1.
efectuar_movimiento(pos(Row, Col), e, pos(Row, NewCol)) :- NewCol is
Col + 1.
efectuar_movimiento(pos(Row, Col), o, pos(Row, NewCol)) :- NewCol is
Col - 1.
efectuar_movimiento(pos(Row, Col), no, pos(NewRow, NewCol)) :- NewRow
is Row - 1, NewCol is Col - 1.
efectuar_movimiento(pos(Row, Col), ne, pos(NewRow, NewCol)) :- NewRow
is Row - 1, NewCol is Col + 1.
efectuar_movimiento(pos(Row, Col), so, pos(NewRow, NewCol)) :- NewRow
is Row + 1, NewCol is Col - 1.
efectuar_movimiento(pos(Row, Col), se, pos(NewRow, NewCol)) :- NewRow
is Row + 1, NewCol is Col + 1.

%my_between/3
%metodo auxiliar para movimiento_valido
%devuelve true si el valor esta entre el limite inferior y superior
my_between(Lower, Upper, Lower) :-
    Lower =< Upper.
my_between(Lower, Upper, Result) :-
    Lower < Upper,
    NewLower is Lower + 1,
    my_between(NewLower, Upper, Result).

%movimiento_valido/3
```

```
%nos movemos una posicion y comprobamos que la nueva posicion este
dentro del tablero
movimiento_valido(N, pos(Row, Col), Dir) :-
    efectuar_movimiento(pos(Row, Col), Dir, pos(NewRow, NewCol)),
    my_between(1, N, NewRow),
    my_between(1, N, NewCol).

%select_cell/4
%selecciona la celda actual y la elimina del tablero
select_cell(IPos, Op, Board, NewBoard) :-
    select_cell_aux(IPos, Op, Board, [], NewBoard).

%la variable Acc actua como una lista de acumuladores para guardar las
celdas que no son la actual
%hay dos reglas select_cell_aux/5, la primera es cuando la celda
actual es la que buscamos y la
%segunda es cuando no es la celda que buscamos. %En caso de que no sea
la celda que buscamos se
%aniade a la lista de acumuladores y se llama de nuevo a
select_cell_aux/5. En caso de que sea la
%celda que buscamos, se invierte la lista de acumuladores para que
quede en el orden correcto y
%se concatena con el resto del tablero
select_cell_aux(IPos, Op, [Cell|Rest], Acc, NewBoard) :-
    Cell = cell(IPos, Op),
    reverse(Acc, ReverseAcc),
    append(ReverseAcc, Rest, NewBoard).

select_cell_aux(IPos, Op, [Cell|Rest], Acc, NewBoard) :-
    Cell \= cell(IPos, Op),
    select_cell_aux(IPos, Op, Rest, [Cell|Acc], NewBoard).

%select_dir/3
%selecciona la direccion actual y la elimina de la lista de
direcciones
%hay tres reglas select_dir/3, la primera es cuando la direccion
actual es la que buscamos y la
%segunda es cuando no es la direccion que buscamos. En caso de que no
sea la direccion que buscamos
```

```
%se aniade a la lista de acumuladores y se llama de nuevo a
select_dir/3. En caso de que sea la
%direccion que buscamos, se decrementa el numero de veces que se puede
usar esa direccion y se
%concatena con el resto de direcciones
select_dir(Dir, [dir(Dir, 1) | Rest], Rest).
select_dir(Dir, [dir(Dir, N) | Rest], [dir(Dir, NewN) | Rest]) :-
    N > 1,
    NewN is N - 1.
select_dir(Dir, [OtherDir | Rest], [OtherDir | NewRest]) :-
    select_dir(Dir, Rest, NewRest).

%aplicar_op/3
%aplica la operacion a un valor
aplicar_op(op(+, X), Valor, Valor2) :- Valor2 is Valor + X.
aplicar_op(op(-, X), Valor, Valor2) :- Valor2 is Valor - X.
aplicar_op(op(*, X), Valor, Valor2) :- Valor2 is Valor * X.
aplicar_op(op(//, X), Valor, Valor2) :- Valor2 is Valor // X.

%generar_recorrido/6
%Predicado para generar el recorrido y valor a partir de la posición
inicial y el tablero
generar_recorrido(IPos, N, Board, DireccionesPermitidas, Recorrido,
Valor) :-
    select_cell(IPos, Op, Board, NewBoard), %Seleccionar celda actual
    aplicar_op(Op, 0, ValorActual), %Valor actual es 0 al inicio
    generar_recorrido_aux(IPos, N, NewBoard, DireccionesPermitidas,
[(IPos, ValorActual)], Recorrido, ValorActual, Valor).

%Caso base: no hay mas celdas en el tablero
generar_recorrido_aux(_, _, [], _, RecorridoInvertido, Recorrido,
Valor, Valor) :-
    reverse(RecorridoInvertido, Recorrido). %Invertir el recorrido
para que quede en el orden correcto

%Caso recursivo: realizar el recorrido
generar_recorrido_aux(IPos, N, Board, DireccionesPermitidas,
RecorridoParcial, Recorrido, ValorAux, Valor) :-
    %Seleccionar una dirección
```

```
    select_dir(Dir, DireccionesPermitidas, NewDireccionesPermitidas),
    %Obtener la nueva posición
    efectuar_movimiento(IPos, Dir, NewPos),
    %Verificar si el movimiento es valido
    movimiento_valido(N, IPos, Dir),
    %Seleccionar celda actual
    select_cell(NewPos, Op, Board, NewBoard),
    %Aplicar la operación al valor actual
    aplicar_op(Op, ValorAux, ValorActual),
    %Llamada recursiva
    generar_recorrido_aux(NewPos, N, NewBoard,
NewDireccionesPermitidas, [(NewPos, ValorActual) | RecorridoParcial],
Recorrido, ValorActual, Valor).

%generar_recorridos/5
%Vamos a coger cada cell del tablero y generar_recorrido con cada una
de ellas. El metodo auxiliar
%esta para poder coger cada cell del tablero pero aun pudiendo llamar
el tablero entero en  generar_recorrido
generar_recorridos(N, Board, DireccionesPermitidas, Recorrido, Valor)
:-
    member(cell(IPos,_), Board),
    generar_recorrido(IPos, N, Board, DireccionesPermitidas,
Recorrido, Valor).

%my_min_list/2
%metodo auxiliar para tablero/5
%devuelve el valor minimo de una lista
%comparamos de dos en dos los elementos de la lista y nos quedamos con
el menor
my_min_list([X], X). %caso base, lista con 1 elem
my_min_list([X, Y | Tail], Min) :-
    X =< Y,
    my_min_list([X | Tail], Min).
my_min_list([X, Y | Tail], Min) :-
    X > Y,
    my_min_list([Y | Tail], Min).

%tablero/5
```

```
tablero(N, Board, DireccionesPermitidas, ValorMinimo,
NumeroDeRutasConValorMinimo) :-
    %guardamos todos los la lista Valores
    findall(Valor, generar_recorridos(N, Board, DireccionesPermitidas,
Recorrido, Valor), Valores),
    %buscamos el valor minimo de la lista Valores
    my_min_list(Valores, ValorMinimo),
    %guardamos los recorridos con ValorMinimo la lista Rutas
    findall(Recorrido, generar_recorridos(N, Board,
DireccionesPermitidas, Recorrido, ValorMinimo), Rutas),
    %la longitud de la lista Rutas sera el numero de rutas con valor
minimo
    length(Rutas, NumeroDeRutasConValorMinimo).
```

Consultas realizadas con el programa y respuestas obtenidas para dichas consultas

A continuación, podemos ver las distintas pruebas realizadas para asegurar un mínimo funcionamiento del código.

```
?- board1(_B),
   dirs1(_D),
   generar_recorrido(pos(1,2),2,_B,_D,R,V).

R = [(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996),(pos(1,1),-5988)],
V = -5988 ? ;

R = [(pos(1,2),-1),(pos(1,1),3),(pos(2,1),0),(pos(2,2),2000)],
V = 2000 ? ;

no
?- board1(_B),
   dirs1(_D),
   generar_recorrido(pos(2,2),4,_B,_D,R,V).

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-1002),(pos(3,2),-
155310),(pos(3,3),-77655),(pos(2,3),-10328115),(pos(1,3),-
```

```
10328119),(pos(1,4),-10328674),(pos(2,4),-10329118),(pos(3,4),-
10329098),(pos(4,4),-41316392),(pos(4,3),-41316401)],
V = -41316401 ? ;

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-1002),(pos(3,2),-
155310),(pos(3,3),-77655),(pos(4,3),-77664),(pos(4,4),-
310656),(pos(3,4),-310636),(pos(2,4),-311080),(pos(1,4),-
311635),(pos(1,3),-311639),(pos(2,3),-41447987)],
V = -41447987 ? ;

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-1002),(pos(3,2),-
155310),(pos(3,3),-77655),(pos(4,3),-77664),(pos(4,4),-
310656),(pos(3,4),-310636),(pos(2,4),-311080),(pos(2,3),-
41373640),(pos(1,3),-41373644),(pos(1,4),-41374199)],
V = -41374199 ? ;

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-1002),(pos(4,3),-
1011),(pos(4,4),-4044),(pos(3,4),-4024),(pos(2,4),-4468),(pos(1,4),-
5023),(pos(1,3),-5027),(pos(2,3),-668591),(pos(3,3),-
334295),(pos(3,2),-51815725)],
V = -51815725 ? ;

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000),(pos(3,1),0),(pos(3,2),0),(pos(3,3),0),(pos(2,3),0),(pos(1,3),-
4),(pos(1,4),-559),(pos(2,4),-1003),(pos(3,4),-983),(pos(4,4),-
3932),(pos(4,3),-3941),(pos(4,2),-4941),(pos(4,1),-4943)],
V = -4943 ? ;

yes
?- board1(_B),
   dirs1(_D),
   generar_recorridos(4,_B,_D,R,V).

R = [(pos(1,1),0),(pos(2,1),-3),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-
1002),(pos(3,2),-155310),(pos(2,2),-153310),(pos(1,2),-
153311),(pos(1,3),-153315),(pos(1,4),-153870),(pos(2,4),-
```

```
154314),(pos(3,4),-154294),(pos(4,4),-617176),(pos(4,3),-
617185),(pos(3,3),-308592),(pos(2,3),-41042736)],
V = -41042736 ? ;

R = [(pos(1,1),0),(pos(2,1),-3),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-
1002),(pos(3,2),-155310),(pos(2,2),-153310),(pos(1,2),-
153311),(pos(1,3),-153315),(pos(1,4),-153870),(pos(2,4),-
154314),(pos(2,3),-20523762),(pos(3,3),-10261881),(pos(4,3),-
10261890),(pos(4,4),-41047560),(pos(3,4),-41047540)],
V = -41047540 ? ;

R = [(pos(1,1),0),(pos(2,1),-3),(pos(3,1),0),(pos(4,1),-2),(pos(4,2),-
1002),(pos(3,2),-155310),(pos(2,2),-153310),(pos(1,2),-
153311),(pos(1,3),-153315),(pos(1,4),-153870),(pos(2,4),-
154314),(pos(2,3),-20523762),(pos(3,3),-10261881),(pos(3,4),-
10261861),(pos(4,4),-41047444),(pos(4,3),-41047453)],
V = -41047453 ?

yes
?- board2(_B),
   dirs1(_D),
   generar_recorridos(2,_B,_D,R,V).

R = [(pos(1,1),0),(pos(2,1),-3),(pos(2,2),1997),(pos(1,2),1996)],
V = 1996 ? ;

R = [(pos(1,1),0),(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996)],
V = 1996 ? ;

R = [(pos(1,2),-1),(pos(2,2),1999),(pos(2,1),1996),(pos(1,1),-5988)],
V = -5988 ? ;

R = [(pos(1,2),-1),(pos(1,1),3),(pos(2,1),0),(pos(2,2),2000)],
V = 2000 ? ;

R = [(pos(2,1),-3),(pos(1,1),9),(pos(1,2),8),(pos(2,2),2008)],
V = 2008 ? ;

R = [(pos(2,1),-3),(pos(2,2),1997),(pos(1,2),1996),(pos(1,1),-5988)],
```

```
V = -5988 ? ;

R = [(pos(2,2),2000),(pos(1,2),1999),(pos(1,1),-5997),(pos(2,1),-
6000)],
V = -6000 ? ;

R = [(pos(2,2),2000),(pos(2,1),1997),(pos(1,1),-5991),(pos(1,2),-
5992)],
V = -5992 ? ;

no
?- board2(_B),
   dirs1(_D),
   tablero(2,_B,_D,VMin,NumR).

NumR = 1,
VMin = -6000 ? ;

no
?- board1(_B),
   dirs1(_D),
   tablero(4,_B,_D,VMin,NumR).

NumR = 1,
VMin = -246992940 ? ;

no
```