Trabajo Práctico de Programación Lógica

Martes 17 de Octubre de 2017 — Paradigmas de Lenguajes de Programación

Integrante	LU	Correo electrónico
Francisco Demartino	348/14	demartino.francisco@gmail.com
Martín Mongi Badía	422/13	martinmongi@gmail.com
Grupo: "Demo"		

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellon I/Planta Baja)
Intendente Güiraldes 2160 - C1428EGA
Ciudad Autonoma de Buenos Aires - Rep. Argentina
Tel/Fax: (54 11) 4576-3359
http://www.fcen.uba.ar

1 batalla_naval.pl

```
% para recargar el fuente con "r." en vez de "[batalla_naval]."
r :- [batalla_naval].

%------------------Predicados predefinidos:------------------%

%fliplength(?Longitud, ?Lista)
fliplength(N, L) :- length(L, N).

%matriz(?Matriz, ?Filas, ?Columnas)
matriz(M, F, C) :- length(M, F), maplist(fliplength(C), M).

%dif1(+N1, ?N2)
dif1(N1, N2) :- N2 is N1 + 1.
dif1(N1, N2) :- N2 is N1 - 1.

%adyacente(+F1, +C1, ?F2, ?C2)
adyacente(FF, C1, FF, C2) :- dif1(C1, C2).
adyacente(F1, CC, F2, CC) :- dif1(F1, F2).
adyacente(F1, C1, F2, C2) :- dif1(C1, C2), dif1(F1, F2).

%enRango(+Matriz, +Fila, +Columna)
enRango([Fila|Filas], F, C) :-
  F > 0,
  C > 0,
  length([Fila|Filas], FMax),
    F =< FMax,
  length(Fila, CMax),
    C =< CMax.

%adyacenteEnRango(+Tablero, +F1, +C1, ?F2, ?C2)
adyacenteEnRango(T, F1, C1, F2, C2) :-
  adyacente(F1, C1, F2, C2),
  enRango(T, F2, C2).

%------------------Predicados a definir:------------------%

%contenido(+?Tablero, ?Fila, ?Columna, ?Contenido)
contenido(T, F, C, Cont) :- nth1(F, T, Fila), nth1(C, Fila, Cont).

%vacio(+?Tablero, ?Fila, ?Columna)
vacio(T, F, C) :- contenido(T, F, C, Cont), var(Cont).

%disponible(+Tablero, ?Fila, ?Columna)
disponible(T, F, C) :- vacio(T, F, C),
                       forall(adyacenteEnRango(T, F, C, F_1, C_1),
                       vacio(T, F_1, C_1)).
```

```
%puedoColocar(+CantPiezas, ?Direccion, +Tablero, ?Fila, ?Columna)
puedoColocar(1, _, T, F, C) :- disponible(T, F, C).
puedoColocar(N, vertical, T, F, C) :- disponible(T, F, C), N1 is N - 1, F1 is F + 1,
                                      puedoColocar(N1, vertical, T, F1, C).
puedoColocar(N, horizontal, T, F, C) :- disponible(T, F, C), N1 is N - 1, C1 is C + 1,
                                        puedoColocar(N1, horizontal, T, F, C1).

%hayBarco(+CantPiezas, ?Direccion, +Tablero, ?Fila, ?Columna)
hayBarco(1, _, T, F, C) :- contenido(T, F, C, o).
hayBarco(N, vertical, T, F, C) :- contenido(T, F, C, o), N1 is N - 1, F1 is F + 1,
                                  hayBarco(N1, vertical, T, F1, C).
hayBarco(N, horizontal, T, F, C) :- contenido(T, F, C, o), N1 is N - 1, C1 is C + 1,
                                    hayBarco(N1, horizontal, T, F, C1).

%ubicarBarcos(+Barcos, +?Tablero)
ubicarBarcos([], _).
ubicarBarcos([B|Bs], T) :- puedoColocar(B, Dir, T, F, C), hayBarco(B, Dir, T, F, C),
                           ubicarBarcos(Bs, T).

%completarConAgua(+?Tablero)
completarConAgua(T) :- maplist(completarFilaConAgua, T).

%completarFilaConAgua(+?Fila)
completarFilaConAgua(F) :- maplist(completarCasilleroConAgua, F).

%completarCasilleroConAgua(+?Casillero)
completarCasilleroConAgua(~) :- !.
completarCasilleroConAgua(o).

%reemplazar(+Lista, +Indice, +Elemento, -Resultado)
reemplazar([], _, _, []).
reemplazar([_|Xs], 1, Z, [Z|Xs]).
reemplazar([X|Xs], N, Z, [X|Ys]) :- N > 1, N1 is N - 1, reemplazar(Xs, N1, Z, Ys).

%golpear(+Tablero, +NumFila, +NumColumna, -NuevoTab)
golpear(T, F, C, T) :- contenido(T, F, C, ~).
golpear(T, F, C, NuevoT) :- contenido(T, F, C, o), nth1(F, T, Fila),
                            reemplazar(Fila, C, ~, NuevaFila),
                            reemplazar(T, F, NuevaFila, NuevoT).

% Completar instanciación soportada y justificar.
%atacar(+Tablero, ?Fila, ?Columna, ?Resultado, ?NuevoTab)
atacar(T, F, C, agua, T) :- golpear(T, F, C, T).
atacar(T, F, C, hundido, NuevoT) :- contenido(T, F, C, o), golpear(T, F, C, NuevoT),
                                    forall(adyacenteEnRango(T, F, C, F_1, C_1),
                                    contenido(T, F_1, C_1, ~)).
atacar(T, F, C, tocado, NuevoT) :- contenido(T, F, C, o), golpear(T, F, C, NuevoT),
                                   \+ forall(adyacenteEnRango(T, F, C, F_1, C_1),
                                   contenido(T, F_1, C_1, ~)).
```

```
%------------------Tests:------------------%

test(1) :-
  matriz(M, 2, 3),
  adyacenteEnRango(M, 2, 2, 2, 3).

test(2) :-
  matriz(M, 2, 3),
  setof(
    (F, C),
    adyacenteEnRango(M, 1, 1, F, C),
    [(1, 2), (2, 1), (2, 2)]
  ).

test(3) :-
  contenido([[o,~]], 1, 1, o),
  contenido([[o,~]], 1, 2, ~).

test(4) :-
  matriz(M, 1, 1),
  vacio(M, 1, 1).

test(5) :-
  matriz(M, 3, 3),
  contenido(M, 1, 2, o),
  setof(
    (F, C),
    disponible(M, F, C),
    [(3, 1), (3, 2), (3, 3)]
  ).

test(6) :-
  matriz(M, 2, 4),
  setof(
    (F, C, Dir),
    puedoColocar(3, Dir, M, F, C),
    [(1, 1, horizontal), (1, 2, horizontal), (2, 1, horizontal), (2, 2, horizontal)]
  ).

test(7) :-
  matriz(M,2,3),
  contenido(M,2,1,o),
  setof(
    (F, C, Dir),
    puedoColocar(2,Dir,M,F,C),
    [(1, 3, vertical)]
  ).
```

```
test(8) :-
  matriz(M,3,2),
  setof(
    M,
    ubicarBarcos([2,1],M),
    [
      [[o, o], [_, _], [o, _]],
      [[o, o], [_, _], [_, o]],
      [[o, _], [_, _], [o, o]],
      [[_, o], [_, _], [o, o]]
    ]
  ).

test(9) :-
  matriz(M,1,2),
  contenido(M,1,1,~),
  contenido(M,1,2,o),
  golpear(M,1,1,IntM),
  golpear(IntM,1,2,NuevaM),
  contenido(NuevaM,1,1,~),
  contenido(NuevaM,1,2,~).

tests :- forall(between(1, 9, N), test(N)).
```