

Encaje de Patrones: Eliza

Curso 2002/03, Fecha:17/01/2002

Enunciado 1 (busca) Construir un predicado $busca(C,D,V)$ que se cumple si V es el valor de la clave C en el diccionario D

El diccionario se representa como una lista de pares de la forma $[(C1,V1),\dots(Cn,Vn)]$

?- $busca(2, [(4, hola), (2, juan), (3, pepe)], V)$. $V = juan$

Enunciado 2 (encaja) Añadir al programa anterior el siguiente predicado.

$encaja([N|Ps], D, R)$: - $integer(N)$,
 $busca(N, D, LsIzq)$,
 $append(LsIzq, LsDer, R)$,
 $encaja(Ps, D, LsDer)$.

$encaja([P|Ps], D, [P|Rs])$: - $atom(P)$,
 $encaja(Ps, D, Rs)$.

$encaja([], -, [])$.

Observar y justificar el funcionamiento del predicado ante las siguientes preguntas:

- ?- $encaja([hola, 1], V, [hola, amigo, pepe])$.
- ?- $encaja([1, es, 2], V, [juan, lopez, es, un, poco, tonto])$.
- ?- $encaja([1, y, 2, son, amigos], [(1, [juan, lopez]), (2, [pepe])], V)$.

Enunciado 3 (leePals) El siguiente fragmento define un predicado que permite leer una lista de palabras en Prolog. Construir un predicado que repita el proceso de leer una lista de palabras e imprimir la lista en orden inverso hasta que se introduzca la palabra adios

$leePals(Ps)$: - $once((get_code(C)$,
 $leePs(C, Ps)))$.

$leePs(C, [])$: - $retorno(C), !$.

$leePs(C, [P|Ps])$: - $alfaNum(C), !$,
 $leeP(C, P, Csig)$,

```

        leePs (Csig, Ps).
leePs (_, Ps):- get_code (Csig),
                leePs (Csig, Ps).

leeP (C1, P, Csig):- cogeCs (C1, Cs, Csig),
                    atom_codes (P, Cs).

cogeCs (C1, [C1|Cars], Cfin):-
    alfaNum (C1),
    get_code (Csig),
    cogeCs (Csig, Cars, Cfin).
cogeCs (C, [], C):- \+ alfaNum (C).

alfaNum (C):- (C>=0'a, C<=0'z)
              ; (C>=0'A, C<=0'Z)
              ; (C>=0'0, C<=0'9).

retorno (10).

```

Enunciado 4 (eliza) Añadir el siguiente fragmento al programa anterior.

```

eliza :- saludo ,
        repeat ,
        leePals (Ps) ,
        trata (Ps) ,
        Ps = [adios] ,
        !.

saludo :- write ('Programa conversacional tipo Eliza '), nl.

trata (Ps):- once ((patron (Estimulo , Respta) ,
                    encaja (Estimulo , Dicc , Ps) ,
                    encaja (Respta , Dicc , Salida) ,
                    responde (Salida)
                    )).

% Patrones Estimulo/Respuesta
patron ([adios] , ['Hasta ' , la , proxima]).
patron ([si] , [puedes , ser , mas , expresivo , ?]).
patron ([no] , [puedes , ser , mas , positivo , ?]).
patron ([hola] , ['Cuentame ' , algo]).
patron ([eres , 1] , ['Por ' , que , crees , que , soy , 1 , ?]).
patron ([soy , 1] , ['Por ' , que , crees , que , eres , 1 , ?]).
patron ([estoy , 1] , ['Estas ' , 1 , a , menudo , ?]).
patron ([quiero , 1] , ['Conoces ' , a , alguien , mas , que , quiera , 1 , ?]).

```

patron ([1], ['Por', favor, continua]).

responde ([X|Xs]): - write (X), write (' '), responde (Xs).
responde ([]): - nl.

Se pide:

- *Compilar y ejecutar el programa anterior*
- *Modificar y añadir más capacidad conversacional al programa anterior*

Enunciado 5 (conceptos) *Añadir patrones con el siguiente formato:*

patron ([1, mi, X, 2], ['Dime', algo, mas, sobre, tu, X]): - *puedePoseerse* (X).
patron ([1, X, 2], ['Quieres', que, hablemos, de, Y, ?]): - *concepto* (X, Y).

Para ello, se pueden añadir predicados del tipo:

puedePoseerse (madre).
puedePoseerse (coche).

concepto (amigo, amistad).
concepto (tonto, insultos).
concepto (imbecil, insultos).

Nota: Los patrones deben añadirse antes del patrón *patron* ([1],...) que encaja con cualquier frase. En caso contrario encajará siempre dicho patrón por defecto

Enunciado 6 (tema(Opcional)) *Añadir al programa anterior información que le permita mantener una conversación sobre un tema concreto, por ejemplo: deportes, música, etc.*

Enunciado 7 (respuestaAlea(Opcional)) *Modificar el programa eliza para que el formato de los patrones sea: *patron*(E,[R1,R2...Rn]). Es decir que a cada estímulo E le corresponden una serie de respuestas y el sistema selecciona una de dichas respuestas de forma aleatoria.*