SEIII - Logikprogrammierung

Übungsblatt 10

Nico Hahn 6990715

Hieu Nguyen 6632126

```
?- consult('dax2016.pl').
% ?- consult('dax2017.pl').
?- consult('display_pi.pl').

% Aufgabe 1
to_timestamp([Day,Month,Year],Timestamp) :-
  string_concat(Year,'-',Year_),
  string_concat(Year_,Month,Year_Month),
  string_concat(Year_Month,'-',Year_Month_),
  string_concat(Year_Month_,Day,Year_Month_Day),
  parse_time(Year_Month_Day, 'iso_8601', Timestamp).
string_to_timestamp(DatumStr,Timestamp) :-
  split_string(DatumStr, ".", "", DatumTeile),
  to_timestamp(DatumTeile,Timestamp).
dax_timestamp(Timestamp, X1, X2) :- dax(Datum,X1,X2),
string_to_timestamp(Datum,Timestamp).

sort_dax(DAX) :-
  findall((Timestamp, X1, X2), dax_timestamp(Timestamp, X1, X2), List),
  sort(1, @<, List, DAX).

get_eroeffnung((_,X1,_),NeuX1) :- NeuX1 is X1 / 500 - 7.
get_schluss((_,_,X2),NeuX2) :- NeuX2 is X2 / 500 - 7.

zip([], [], []).
zip([X|Xs], [Y|Ys], [X,Y|Zs]) :- zip(Xs,Ys,Zs).

get_daten(DAX_Daten) :-
  sort_dax(DAX),
  maplist(get_eroeffnung,DAX,DAX_Eroeffnung),
  maplist(get_schluss,DAX,DAX_Schluss),
  zip(DAX_Eroeffnung, DAX_Schluss, DAX_Daten).

draw_dax() :-
  get_daten(DAX_Daten),
  display('Zeitreihe', DAX_Daten).

% Aufgabe 2
window_average([_,_,_,_,_,_,_,_,_],[]).
```

```
window_average([X1,X2,X3,X4,X5,X6,X7,X8,X9,X10|Rest],[Mittelwert|RestMittelwe
rt]) :-
  Mittelwert is (X1+X2+X3+X4+X5+X6+X7+X8+X9+X10)/10,
  window_average([X2,X3,X4,X5,X6,X7,X8,X9,X10|Rest], RestMittelwert).

% Aufgabe 3
?- consult('display_pi_2.pl').
draw_dax_avg() :-
  get_daten(DAX_Daten),
  window_average(DAX_Daten, Avg),
  display('Zeitreihe', DAX_Daten, Avg).

% Aufgabe 4
% trend(+Start, +End, -Trend)
% start: Anfang des Zeitfenster z.B. 29.02.2016
% End: Ende des Zeitfenster z.B. 29.04.2016
% Trend: Das Trend der Indexentwicklung
%
% Ein Mittelpunkt wird aus der Daten in dem angegebenen Fenster
% berechnet und dann verglichen mit anderen Punkten.
% Falls mehr kleinere Punkte als größere vorliegt, ist es ein Aufwärtstrend.
% Andereseits ein Abwärtstrend. Falls die Anzahl von kleineren und größeren
Punkten
% gleich sind, dann ist es eine Seitwärtsbewegung.
trend(Start, End, Trend) :-
  findall(X, dax_between(Start,End,X), DAX_Betweens),
  maplist(middlepoint,DAX_Betweens,DAX_Middlepoints),
  average(DAX_Middlepoints, AVG),
  count_bigger_smaller(DAX_Middlepoints, AVG, CountBigger, CountSmaller),
  (CountSmaller > CountBigger, Trend = "Aufwärtstrend"
  ; CountBigger > CountSmaller, Trend = "Abwärtstrend"
  ; CountBigger = CountSmaller, Trend = "Seitwärtsbewegung").

count_bigger_smaller([], _, 0, 0).
count_bigger_smaller([A|Rest], X, Bigger, Smaller) :-
  A > X,
  count_bigger_smaller(Rest, X, BiggerRest, Smaller),
  Bigger is BiggerRest + 1.
count_bigger_smaller([A|Rest], X, Bigger, Smaller) :-
  A < X,
  count_bigger_smaller(Rest, X, Bigger, SmallerRest),
  Smaller is SmallerRest + 1.
count_bigger_smaller([A|Rest], X, Bigger, Smaller) :-
  A = X,
  count_bigger_smaller(Rest, X, Bigger, Smaller).

% Mittelwert einer Liste
average(List, Average) :- sum_list(List, Sum),
                          length(List, Count),
```

```
                              Average is Sum/Count.

% Mittelpunkt von Eroeffnung und Schluss
middlepoint((_,X1,X2), AVG) :- AVG is (X1 + X2) / 2.

% find all dax index between start and end
dax_between(Start, End, (Time, X1, X2)) :-
  string_to_timestamp(Start, StartTime),
  string_to_timestamp(End, EndTime),
  dax_timestamp(Time, X1, X2),
  StartTime =< Time,
  Time =< EndTime.

% Aufgabe 5
difference((_,X1,X2), Diff) :- Diff is X2 - X1.
guess_diff(Start, End, GuessedDiff) :-
  findall(X, dax_between(Start,End,X), DAX_Betweens),
  maplist(difference,DAX_Betweens,DAX_Differences),
  average(DAX_Differences, GuessedDiff).
```