Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота № 3
з дисципліни “Мультипарадигмене програмування”

Виконав:
Студент групи ІО-23
Швед А. Д.

Київ - 2025

Завдання

За допомогою логічного програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів.

Вхідні данні

Чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

Вихідні дані

Лінгвістичний ряд та матриця передування.

Мова програмування

SWI-Prolog.

Хід роботи

Програма реалізує перетворення числового ряду у лінгвістичний ланцюжок з наступним побудуванням матриці передування.

Правила

- Введення алфавіту користувачем
- Читання даних з файлу

Визначення діапазону значень

Правила встановлюють мінімальне та максимальне значення в числовому ряді для подальшого розбиття діапазону.

Побудова рівномірних інтервалів

Використовуючи знайдені границі та потужність алфавіту, визначаються рівномірні інтервали, кожен з яких асоціюється з символом алфавіту.

Визначення відповідності числа інтервалу

Для кожного елемента числового ряду визначається інтервал, в який він потрапляє. З цієї інформації числу надається відповідний символ алфавіту.

Побудова лінгвістичного ряду

На основі попереднього правила кожне число замінюється на відповідну літеру, утворюючи послідовність лінгвістичний ряд.

Побудова пар

Формується список усіх сусідніх пар символів у лінгвістичному ряді.

Побудова матриці передування

Кожна можлива пара символів розглядається як елемент матриці. Підраховується кількість кожної пари, що зустрічається у списку пар. Отримані частоти заносяться у таблицю – матрицю передування, яка є результатом аналізу.

Також були написані службові предикати що не мають особливого логічного сенсу але є частиною реалізації роботи.

Результати виконання

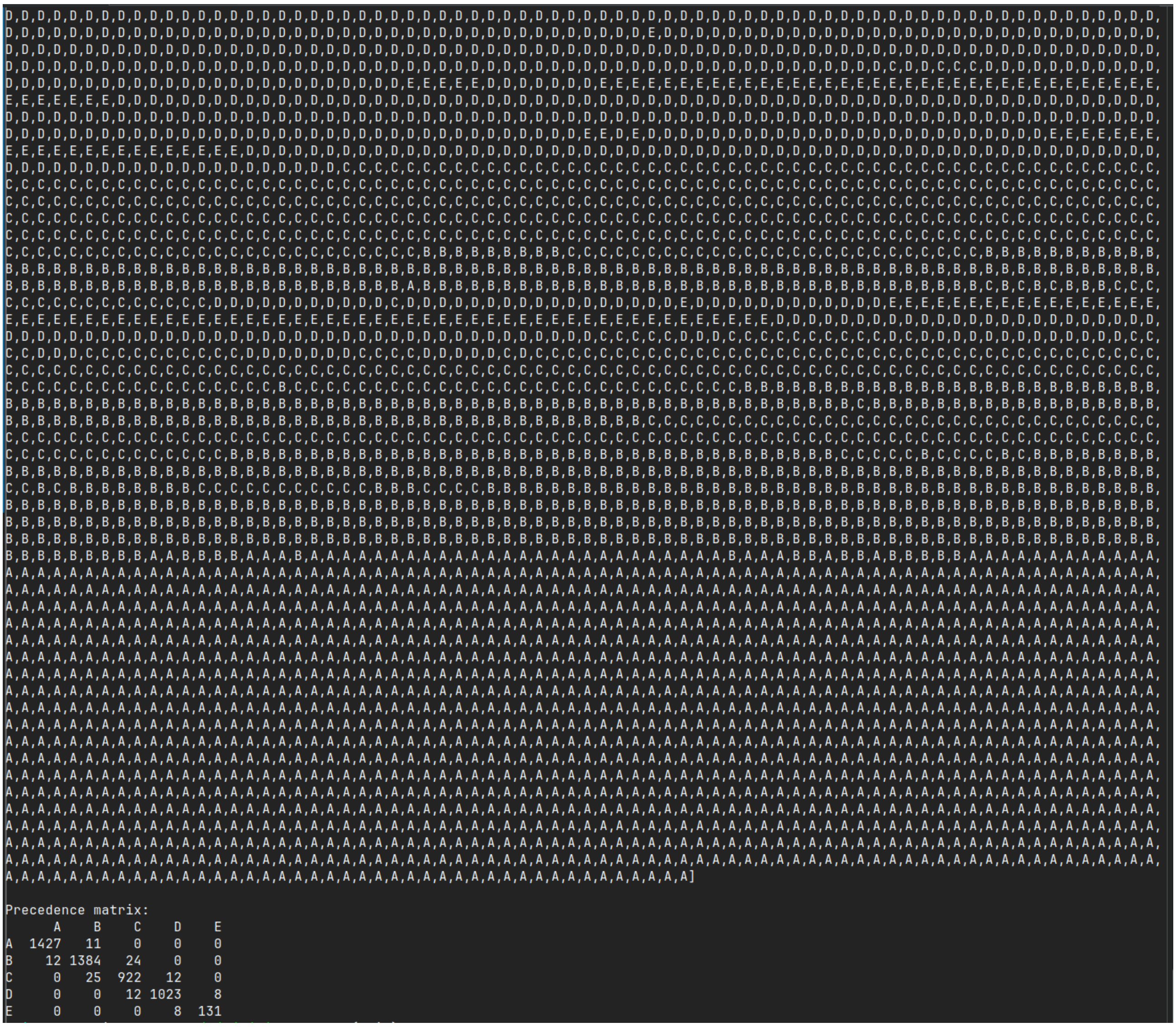
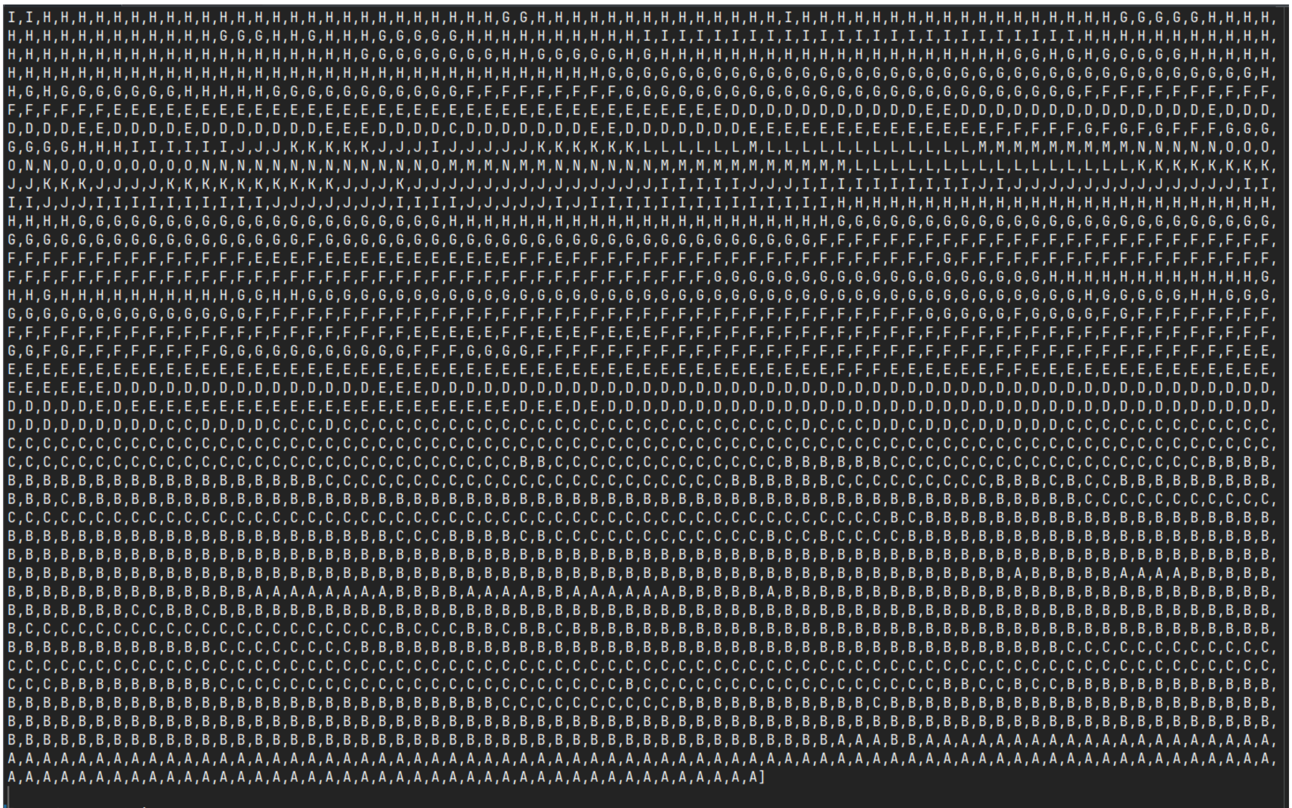


Figure 1: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 5 символів)



Precedence matrix:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
A	154	7	0	0	0	0	0	0	0	0	0	0	0	0	0
B	8	710	29	0	0	0	0	0	0	0	0	0	0	0	0
C	0	30	489	11	0	0	0	0	0	0	0	0	0	0	0
D	0	0	12	269	26	0	0	0	0	0	0	0	0	0	0
E	0	0	0	27	511	29	0	0	0	0	0	0	0	0	0
F	0	0	0	0	30	492	24	0	0	0	0	0	0	0	0
G	0	0	0	0	0	25	404	26	0	0	0	0	0	0	0
H	0	0	0	0	0	0	27	311	7	0	0	0	0	0	0
I	0	0	0	0	0	0	0	7	140	12	0	0	0	0	0
J	0	0	0	0	0	0	0	0	12	136	13	0	0	0	0
K	0	0	0	0	0	0	0	0	0	13	393	46	0	0	0
L	0	0	0	0	0	0	0	0	0	0	46	376	8	0	0
M	0	0	0	0	0	0	0	0	0	0	0	8	88	3	0
N	0	0	0	0	0	0	0	0	0	0	0	0	2	22	3
O	0	0	0	0	0	0	0	0	0	0	0	0	1	2	10

Figure 2: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 15 символів)

Лістинг коду

main.pl

```
% Read a string list from terminal
read_atom_list(List) :-
    read_line_to_string(user_input, Line),
    split_string(Line, " ", "", StringList),
    maplist(atom_string, List, StringList).

% Read a list of floats from a file
read_floats_from_file(FilePath, Floats) :-
    open(FilePath, read, Stream),
    read_lines(Stream, Lines),
    close(Stream),
    maplist(atom_number, Lines, Floats).

% Read lines from file and return them as atom string list
read_lines(Stream, []) :-
    at_end_of_stream(Stream), !.
read_lines(Stream, [LineAtom|Rest]) :-
    read_line_to_string(Stream, Line),
    atom_string(LineAtom, Line),
    read_lines(Stream, Rest).

% Find min element in a list
min([H|T], Min) :- foldl(min, T, H, Min).
min(A, B, Min) :- Min is min(A, B).

% Find max element in a list
max([H|T], Max) :- foldl(max, T, H, Max).
max(A, B, Max) :- Max is max(A, B).

% Building [A, B) intervals
build_intervals(Min, Max, N, Intervals) :-
    Step is (Max - Min) / N,
    build_intervals_helper(Min, Step, N, Intervals).

build_intervals_helper(_, _, 0, []) :- !.
build_intervals_helper(Min, Step, N, [[Min, Max1]|Rest]) :-
    Max1 is Min + Step,
    N1 is N - 1,
    build_intervals_helper(Max1, Step, N1, Rest).

% Finding the interval to which a value belongs
value_interval(Value, [[A,B]|_], 0) :-
    Value >= A, Value < B, !.
value_interval(Value, [_|T], Index) :-
    value_interval(Value, T, Temp),
    Index is Temp + 1.
value_interval(_, [], 0) :- !. % right limit fallback

% Converting a value to an alphabet symbol
value_to_symbol(Value, Intervals, Alphabet, Symbol) :-
    value_interval(Value, Intervals, Index),
    length(Alphabet, L),
    (Index >= L -> LastIndex is L - 1 ; LastIndex is Index),
    nth0(LastIndex, Alphabet, Symbol).

% Converting a number series to a linguistic series
build_linguistic_sequence([], _, _, []).
build_linguistic_sequence([H|T], Intervals, Alphabet, [S|Rest]) :-
    value_to_symbol(H, Intervals, Alphabet, S),
    build_linguistic_sequence(T, Intervals, Alphabet, Rest).


% === Building a precedence matrix ===
% Builds a list of pairs (a->b, b->c, ...)
pairs([], []).
pairs([_], []).
pairs([A,B|T], [(A,B)|Rest]) :-
```

```
    pairs([B|T], Rest).

% Count the occurrences of each pair
count_pairs([], _, 0).
count_pairs([(A,B)|T], (A,B), N) :-
    count_pairs(T, (A,B), N1),
    N is N1 + 1.
count_pairs([(X,Y)|T], (A,B), N) :-
    (X \= A ; Y \= B),
    count_pairs(T, (A,B), N).

% Building a matrix row
build_matrix_row(_, [], _, []).
build_matrix_row(From, [To|T], Transitions, [Count|Rest]) :-
    count_pairs(Transitions, (From, To), Count),
    build_matrix_row(From, T, Transitions, Rest).

% Building a full precedence matrix
build_precedence_matrix(_, [], _, []).
build_precedence_matrix(Alphabet, [From|RestFrom], Transitions, [Row|MatrixRest]) :-
    build_matrix_row(From, Alphabet, Transitions, Row),
    build_precedence_matrix(Alphabet, RestFrom, Transitions, MatrixRest).


% === Formatted matrix printing ===
% Pretty print the matrix
print_matrix(Matrix, Labels) :-
    print_header(Labels),
    print_rows(Matrix, Labels).

% Print aligned matrix header
print_header(Labels) :-
    tab(2),  % Top-left corner space
    forall(member(Label, Labels),
           format('~|~t~w~5+', [Label])),
    nl.

% Print aligned matrix rows
print_rows([], []).
print_rows([Row|RestMatrix], [Label|RestLabels]) :-
    format('~w ', [Label]),
    forall(member(Cell, Row),
           format('~|~t~d~5+', [Cell])),
    nl,
    print_rows(RestMatrix, RestLabels).


% Main function
main :-
    write("Input space-separated alphabet: "),
    read_atom_list(Alphabet),
    read_floats_from_file("data.txt", Series),
    min(Series, Min),
    max(Series, Max),
    length(Alphabet, N),
    build_intervals(Min, Max, N, Intervals),
    build_linguistic_sequence(Series, Intervals, Alphabet, Linguistic),

    write('Linguistic sequence: '), writeln(Linguistic),
    pairs(Linguistic, Transitions),
    build_precedence_matrix(Alphabet, Alphabet, Transitions, Matrix),
    nl, writeln('Precedence matrix:'),
    print_matrix(Matrix, Alphabet),
    halt.
```