Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

Мультипарадигменне програмування

Виконав
студент групи ІА-23:
Курач В.А.

Київ 2025

Завдання: на процедурній мові програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів з подальшою побудовою матриці передування.

Вхідні данні: чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

Вихідні дані: лінгвістичний ряд та матриця передування.

1. Постановка задачі

Метою лабораторної роботи було створення програми, що виконує перетворення числового часового ряду у лінгвістичний ряд. Це досягається дискретизацією числових значень на символи алфавіту, який може бути заданий користувачем за потужністю (кількістю символів) та змістом (набором символів). Результатом роботи є матриця переходів між символами, що відображає статистику послідовності подій у часовому ряді.

Для перевірки роботи використовувалися часові ряди з лабораторних робіт №1 і №2, №3 що забезпечувало можливість порівняння результатів.

2. Хід роботи та розв’язання задачі

Для реалізації рішення було розроблено набір предикатів, які виконують такі основні кроки:

- Зчитування числового ряду з CSV-файлу та вибір потрібного стовпця.
- Визначення алфавіту: якщо заданий набір символів — використати його, інакше — згенерувати алфавіт з латинських літер заданої довжини.
- Рівномірний поділ діапазону значень часового ряду на інтервали відповідно до розміру алфавіту.
- Відображення кожного числового значення у символ алфавіту відповідно до інтервалу.
- Формування матриці переходів між символами лінгвістичного ряду шляхом підрахунку переходів з символу у позиції і до символу у позиції i+1.
- Виведення лінгвістичного ряду та матриці переходів.

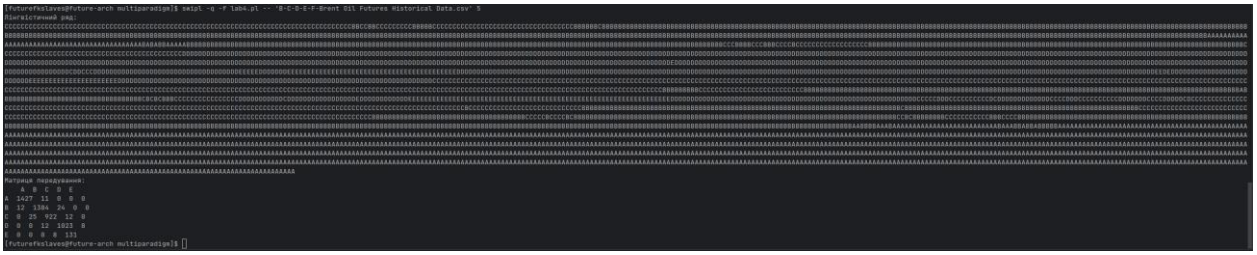
Всі предикати реалізовані з урахуванням принципів функціонального програмування: дані передаються через аргументи, глобальні змінні не використовуються, що забезпечує чистоту обчислень та простоту відладки.

3. Використані інструменти та бібліотеки

- Стандартна бібліотека для роботи з CSV-файлами (library(csv)).
- Вбудовані предикати для роботи зі списками та обчисленнями.
- Середовище виконання — Prolog (наприклад, SWI-Prolog).

4. Результати роботи

Програма успішно виконала перетворення числового часового ряду у лінгвістичний ряд з заданим алфавітом та побудувала матрицю переходів. Вивід містить послідовність символів, що відповідають часовому ряду, та матрицю, що демонструє статистику переходів.



Лістинг програми:

```
:- use_module(library(csv)).
:- use_module(library(lists)).

file_exists(File) :-
    access_file(File, read).

read_csv_column(Filename, Prices) :-
    ( file_exists(Filename) ->
        csv_read_file(Filename, [HeaderRow|DataRows], [functor(row), strip(true)]),
        HeaderRow =.. [_|Headers],
        ( nth1(PriceIndex, Headers, 'Price') ->
            findall(PriceNum, (
                member(Row, DataRows),
                Row =.. [_|Fields],
                nth1(PriceIndex, Fields, PriceAtom),
                atom_string(PriceAtom, Str),
                re_replace(",", "", Str, CleanStr),
                catch(number_string(PriceNum, CleanStr), _, fail)
            ), Prices),
            ( Prices = [] -> format("Файл порожній або не містить числових значень.~n"), fail ; true )
        ; format("Стовпець 'Price' не знайдено у файлі.~n"), fail
        )
    ; format("Файл не знайдено.~n"), fail
    ).

generate_alphabet(Size, Alphabet) :-
```

```
    findall(Char, (between(1, Size, I), char_code('A', A), Code is A + I - 1, char_code(Char, Code)), Alphabet).

get_intervals_uniform(Series, AlphaSize, Breaks) :-
    min_list(Series, Min),
    max_list(Series, Max),
    Step is (Max - Min) / AlphaSize,
    findall(B, (between(0, AlphaSize, I), B is Min + I*Step), Breaks).

map_to_alphabet([], _, _, []).
map_to_alphabet([V|Vs], Breaks, Alphabet, [Char|Chars]) :-
    find_interval(V, Breaks, 1, Alphabet, Char),
    map_to_alphabet(Vs, Breaks, Alphabet, Chars).

find_interval(Value, [Low, High|_], Index, Alphabet, Char) :-
    Value >= Low,
    Value =< High,
    nth1(Index, Alphabet, Char), !.
find_interval(Value, [_|Rest], Index, Alphabet, Char) :-
    NextIndex is Index + 1,
    find_interval(Value, Rest, NextIndex, Alphabet, Char).

build_transition_matrix(LingSeries, Alphabet, Matrix) :-
    length(Alphabet, Size),
    length(Matrix0, Size),
    maplist(init_zeros(Size), Matrix0),
    build_matrix(LingSeries, Alphabet, Matrix0, Matrix).

init_zeros(Size, Row) :-
    length(Row, Size),
    maplist(=(0), Row).

build_matrix([_], _, Matrix, Matrix) :- !.
build_matrix([A,B|Rest], Alphabet, Matrix0, Matrix) :-
    nth1(I, Alphabet, A),
    nth1(J, Alphabet, B),
    nth1(I, Matrix0, Row),
    nth1(J, Row, Val),
    NewVal is Val + 1,
    replace_nth(Row, J, NewVal, NewRow),
    replace_nth(Matrix0, I, NewRow, Matrix1),
    build_matrix([B|Rest], Alphabet, Matrix1, Matrix).

replace_nth([_|T], 1, X, [X|T]).
replace_nth([H|T], N, X, [H|R]) :-
    N > 1,
    N1 is N - 1,
    replace_nth(T, N1, X, R).

print_ling_series(LingSeries) :-
    format("Лінгвістичний ряд:\n"),
    atomic_list_concat(LingSeries, '', Str),
    format("~w~n", [Str]).

print_matrix(Matrix, Alphabet) :-
    format("Матриця передування:\n"),
    print_header(Alphabet),
    print_rows(Matrix, Alphabet).

print_header(Alphabet) :-
```

```
    format("   "),
    forall(member(C, Alphabet), format(" ~w ", [C])),
    nl.

print_rows([], []).
print_rows([Row|Rows], [H|Hs]) :-
    format("~w ", [H]),
    forall(member(Val, Row), format(" ~w ", [Val])),
    nl,
    print_rows(Rows, Hs).

main(Filename, AlphabetSize) :-
    ( read_csv_column(Filename, Series) ->
        generate_alphabet(AlphabetSize, Alphabet),
        get_intervals_uniform(Series, AlphabetSize, Breaks),
        map_to_alphabet(Series, Breaks, Alphabet, LingSeries),
        build_transition_matrix(LingSeries, Alphabet, Matrix),
        print_ling_series(LingSeries),
        print_matrix(Matrix, Alphabet)
    ; true ).

:- initialization(main_from_args).

main_from_args :-
    current_prolog_flag(argv, Args),
    ( Args = [Filename|Rest] ->
        ( Rest = [AlphaSizeAtom|_] ->
            catch(atom_number(AlphaSizeAtom, AlphaSize), _, fail)
        ; AlphaSize = 5
        )
    ; Filename = "B-C-D-E-F-Brent Oil Futures Historical Data.csv",
      AlphaSize = 5
    ),
    main(Filename, AlphaSize),
    halt.
```

Висновки: В ході лабораторної роботи було підтверджено, що функціональний підхід на мові Пролог є ефективним для обробки та аналізу часових рядів. Виконана дискретизація числових даних та побудова матриці переходів дозволяють моделювати лінгвістичні представлення часових процесів. Реалізація без використання глобальних змінних підвищує надійність і підтримуваність коду. Отримані результати можуть бути використані для подальшого аналізу часових рядів та створення систем прогнозування на основі статистичних закономірностей