Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

**Лабораторна робота №6**

з дисципліни:

«Мультипарадигменне програмування»

Виконав:

студент групи ІС-23

Шимків Мирослав

Київ 2025

**ЛАБОРАТОРНАЯ РОБОТА №6**

**Завдання**: за допомогою системи алгебраїчного 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.

**Мова програмування**: MAXIMA.

**Варіант 24**

(завдання 4 відповідно)

Мій варіант 4, отже, я обрав Хі-розподіл і мову програмування MAXIMA

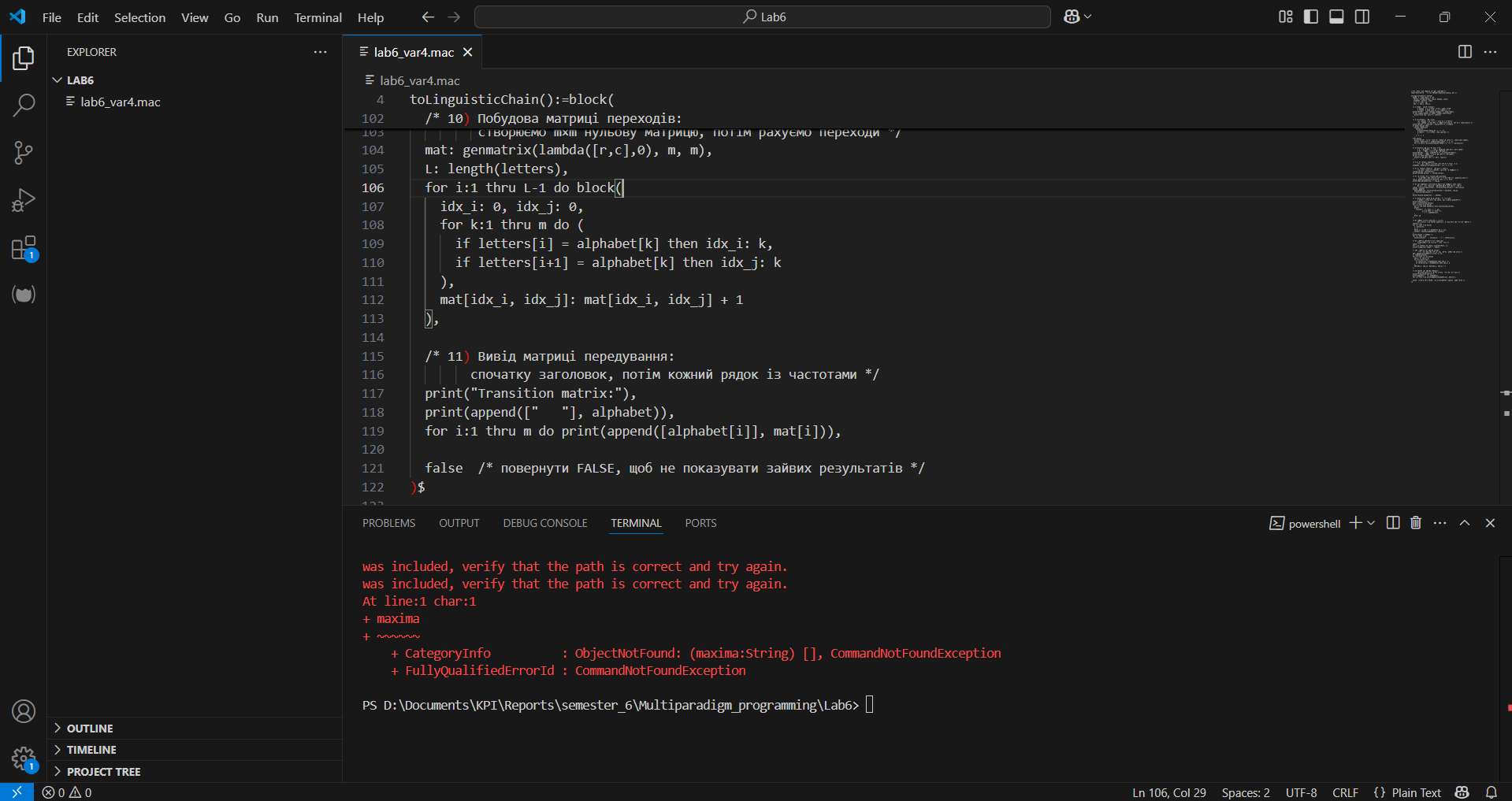
**Виконання роботи**

**Нижче наведено логічні блоки програми:**

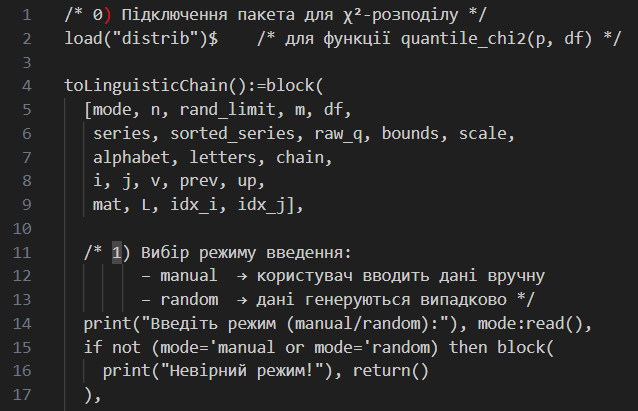
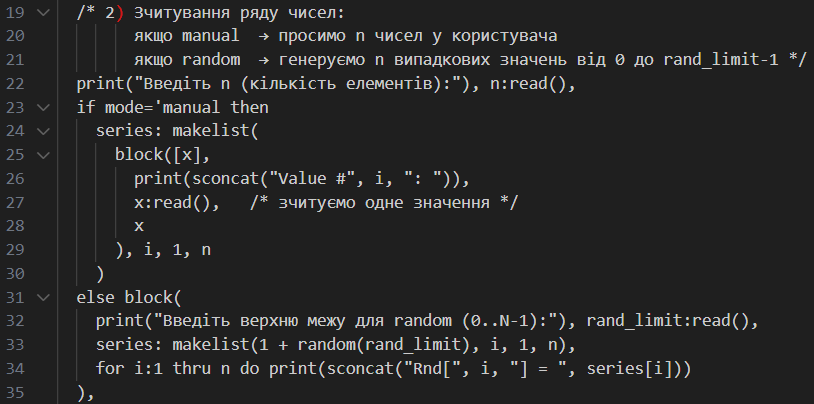
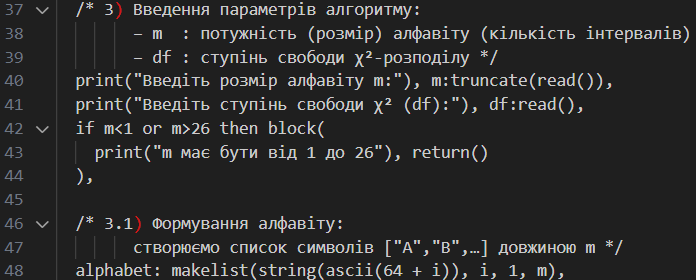
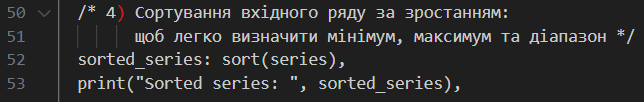
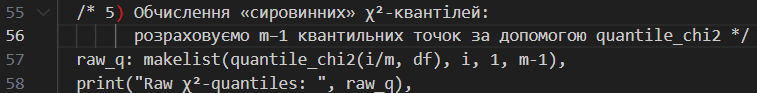
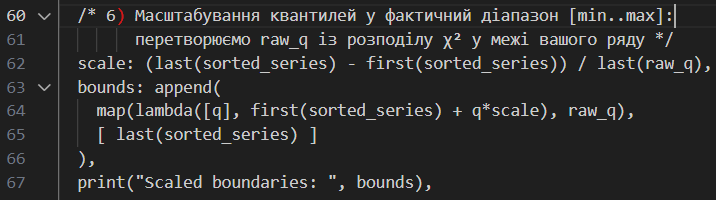
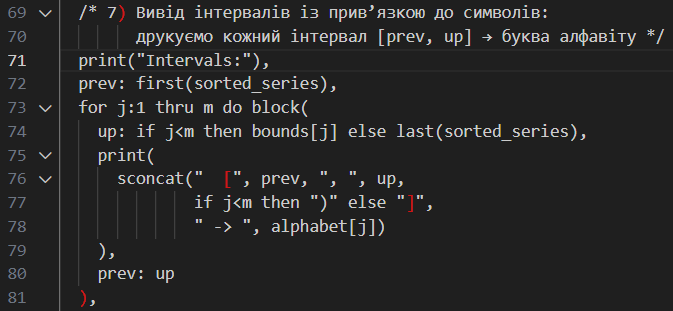
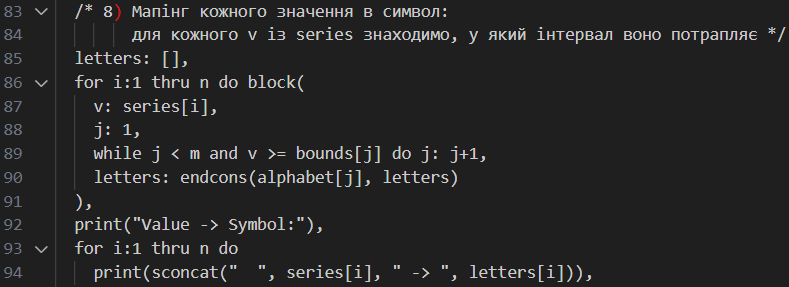
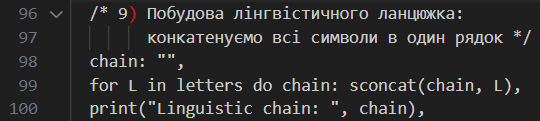
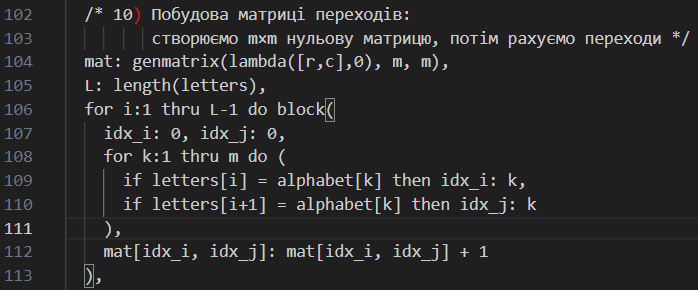
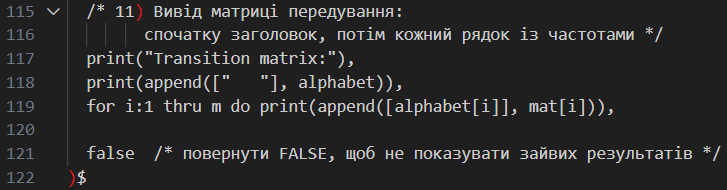
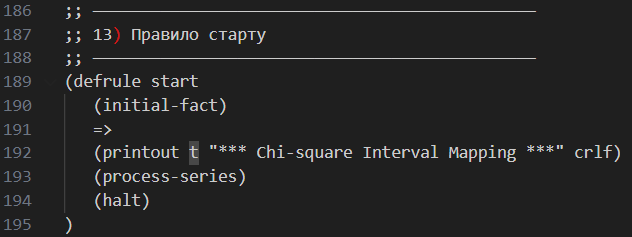
1. Вибір режиму введення:
   1. manual → користувач вводить дані вручну
   2. random → дані генеруються випадково
2. Зчитування ряду чисел:
   1. якщо manual → просимо n чисел у користувача
   2. якщо random → генеруємо n випадкових значень від 0 до rand\_limit-1
3. Введення параметрів алгоритму:
   1. m : потужність (розмір) алфавіту (кількість інтервалів)
   2. df : ступінь свободи χ²-розподілуСортування списку
   3. Формування алфавіту
4. Сортування вхідного ряду за зростанням, щоб легко визначити мінімум, максимум та діапазон
5. Обчислення «сировинних» χ²-квантілей: розраховуємо m−1 квантильних точок за допомогою quantile\_chi2
6. Масштабування квантилей у фактичний діапазон [min..max]: перетворюємо raw\_q із розподілу χ² у межі вашого ряду.
7. Вивід інтервалів із прив’язкою до символів: друкуємо кожний інтервал [prev, up] → буква алфавіту.
8. Мапінг кожного значення в символ: для кожного v із series знаходимо, у який інтервал воно потрапляє.
9. Побудова лінгвістичного ланцюжка: конкатенуємо всі символи в один рядок.
10. Побудова матриці переходів: створюємо m×m нульову матрицю, потім рахуємо переходи.
11. Вивід матриці передування: спочатку заголовок, потім кожний рядок із частотами.

**Давайте розглянемо код для кожного пункту, в якому наведено коментарі українською мовою для кращого розуміння:**

Інтерфейс Visual Studio Code for MAXIMA для запуску програми

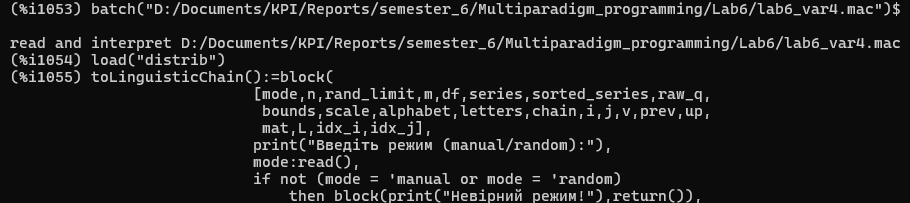


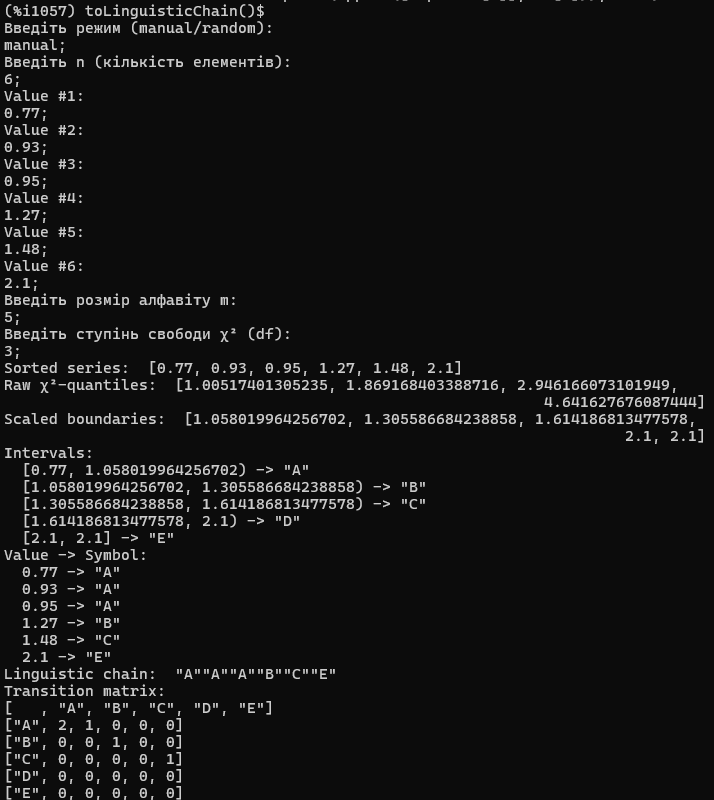
Етапи:

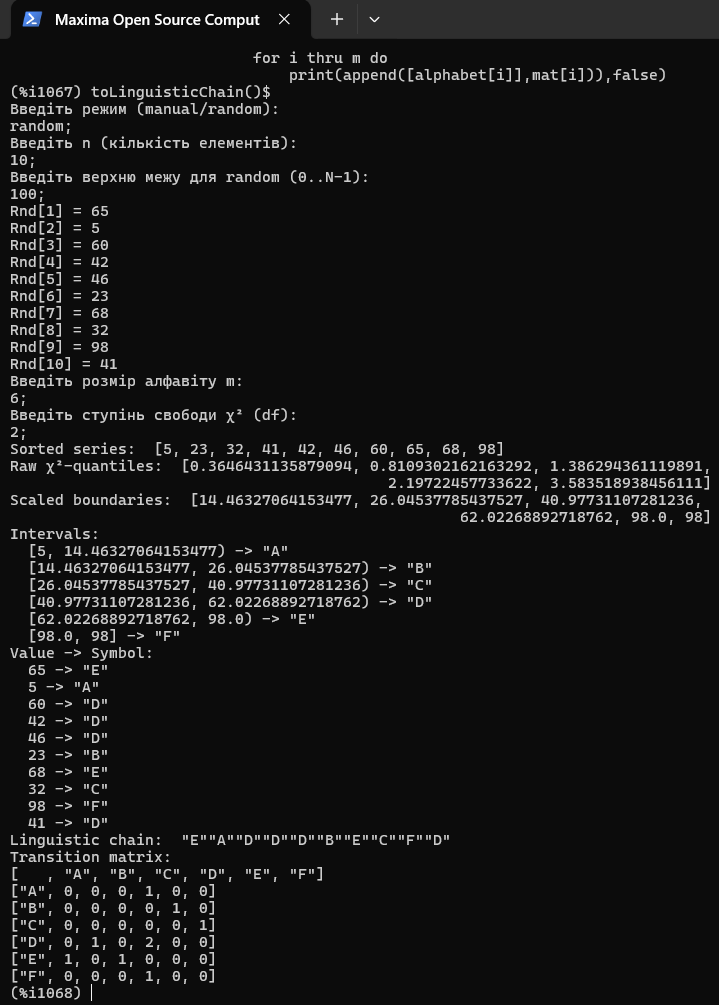
1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 
10. 
11. 
12. 

**Вхідні данні і результат виконання програми:**

Введення значень від руки:





Генерація випадкових значень: 

**Висновок**

У рамках лабораторної роботи №6 реалізовано програму на базі Maxima, яка:  
• Зчитує чисельний ряд, обраний розподіл ймовірностей та потужність алфавіту.  
• Сортує ряд у порядку зростання і визначає мінімальне й максимальне значення (область припустимих значень).  
• Розбиває область на інтервали відповідно до кумулятивної функції обраного розподілу, враховуючи потужність алфавіту.  
• Мапує кожне числове значення на символ алфавіту за індексом інтервалу.  
• Формує лінгвістичний рядок із отриманих символів.  
• Виводить у консоль відсортований ряд, межі інтервалів і згенерований лінгвістичний рядок.

Результати тестів (як за ручного введення, так і при випадковій генерації) доводять коректність алгоритму:  
• Відсортовані значення точно відповідають очікуваному порядку.  
• Межі інтервалів відтворюють розбиття області згідно з обраним розподілом, забезпечуючи правильні «ритми» попадання в інтервали.  
• Лінгвістичний рядок коректно відображає відповідність кожного числа своєму символу алфавіту.

Отже, завдання ЛР №6 виконане повністю й відповідає вимогам лабораторної роботи з алгебраїчного програмування.

**Додатки**

Код програми:

/\* 0) Підключення пакета для χ²-розподілу \*/

load("distrib")$ /\* для функції quantile\_chi2(p, df) \*/

toLinguisticChain():=block(

[mode, n, rand\_limit, m, df,

series, sorted\_series, raw\_q, bounds, scale,

alphabet, letters, chain,

i, j, v, prev, up,

mat, L, idx\_i, idx\_j],

/\* 1) Вибір режиму введення:

– manual → користувач вводить дані вручну

– random → дані генеруються випадково \*/

print("Введіть режим (manual/random):"), mode:read(),

if not (mode='manual or mode='random) then block(

print("Невірний режим!"), return()

),

/\* 2) Зчитування ряду чисел:

якщо manual → просимо n чисел у користувача

якщо random → генеруємо n випадкових значень від 0 до rand\_limit-1 \*/

print("Введіть n (кількість елементів):"), n:read(),

if mode='manual then

series: makelist(

block([x],

print(sconcat("Value #", i, ": ")),

x:read(), /\* зчитуємо одне значення \*/

x

), i, 1, n

)

else block(

print("Введіть верхню межу для random (0..N-1):"), rand\_limit:read(),

series: makelist(1 + random(rand\_limit), i, 1, n),

for i:1 thru n do print(sconcat("Rnd[", i, "] = ", series[i]))

),

/\* 3) Введення параметрів алгоритму:

– m : потужність (розмір) алфавіту (кількість інтервалів)

– df : ступінь свободи χ²-розподілу \*/

print("Введіть розмір алфавіту m:"), m:truncate(read()),

print("Введіть ступінь свободи χ² (df):"), df:read(),

if m<1 or m>26 then block(

print("m має бути від 1 до 26"), return()

),

/\* 3.1) Формування алфавіту:

створюємо список символів ["A","B",…] довжиною m \*/

alphabet: makelist(string(ascii(64 + i)), i, 1, m),

/\* 4) Сортування вхідного ряду за зростанням:

щоб легко визначити мінімум, максимум та діапазон \*/

sorted\_series: sort(series),

print("Sorted series: ", sorted\_series),

/\* 5) Обчислення «сировинних» χ²-квантілей:

розраховуємо m−1 квантильних точок за допомогою quantile\_chi2 \*/

raw\_q: makelist(quantile\_chi2(i/m, df), i, 1, m-1),

print("Raw χ²-quantiles: ", raw\_q),

/\* 6) Масштабування квантилей у фактичний діапазон [min..max]:

перетворюємо raw\_q із розподілу χ² у межі вашого ряду \*/

scale: (last(sorted\_series) - first(sorted\_series)) / last(raw\_q),

bounds: append(

map(lambda([q], first(sorted\_series) + q\*scale), raw\_q),

[ last(sorted\_series) ]

),

print("Scaled boundaries: ", bounds),

/\* 7) Вивід інтервалів із прив’язкою до символів:

друкуємо кожний інтервал [prev, up] → буква алфавіту \*/

print("Intervals:"),

prev: first(sorted\_series),

for j:1 thru m do block(

up: if j<m then bounds[j] else last(sorted\_series),

print(

sconcat(" [", prev, ", ", up,

if j<m then ")" else "]",

" -> ", alphabet[j])

),

prev: up

),

/\* 8) Мапінг кожного значення в символ:

для кожного v із series знаходимо, у який інтервал воно потрапляє \*/

letters: [],

for i:1 thru n do block(

v: series[i],

j: 1,

while j < m and v >= bounds[j] do j: j+1,

letters: endcons(alphabet[j], letters)

),

print("Value -> Symbol:"),

for i:1 thru n do

print(sconcat(" ", series[i], " -> ", letters[i])),

/\* 9) Побудова лінгвістичного ланцюжка:

конкатенуємо всі символи в один рядок \*/

chain: "",

for L in letters do chain: sconcat(chain, L),

print("Linguistic chain: ", chain),

/\* 10) Побудова матриці переходів:

створюємо m×m нульову матрицю, потім рахуємо переходи \*/

mat: genmatrix(lambda([r,c],0), m, m),

L: length(letters),

for i:1 thru L-1 do block(

idx\_i: 0, idx\_j: 0,

for k:1 thru m do (

if letters[i] = alphabet[k] then idx\_i: k,

if letters[i+1] = alphabet[k] then idx\_j: k

),

mat[idx\_i, idx\_j]: mat[idx\_i, idx\_j] + 1

),

/\* 11) Вивід матриці передування:

спочатку заголовок, потім кожний рядок із частотами \*/

print("Transition matrix:"),

print(append([" "], alphabet)),

for i:1 thru m do print(append([alphabet[i]], mat[i])),

false /\* повернути FALSE, щоб не показувати зайвих результатів \*/

)$