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 №3**

з дисципліни:

«Мультипарадигменне програмування»

Виконав:

студент групи ІС-23

Шимків Мирослав

Київ 2025

**ЛАБОРАТОРНАЯ РОБОТА №3**

**Завдання**: за допомогою мультипарадигменної мови R реалізувати перетворення чисельного ряду 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.

**Мова програмування**: R

**Варіант 24**

(завдання 4 відповідно)

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

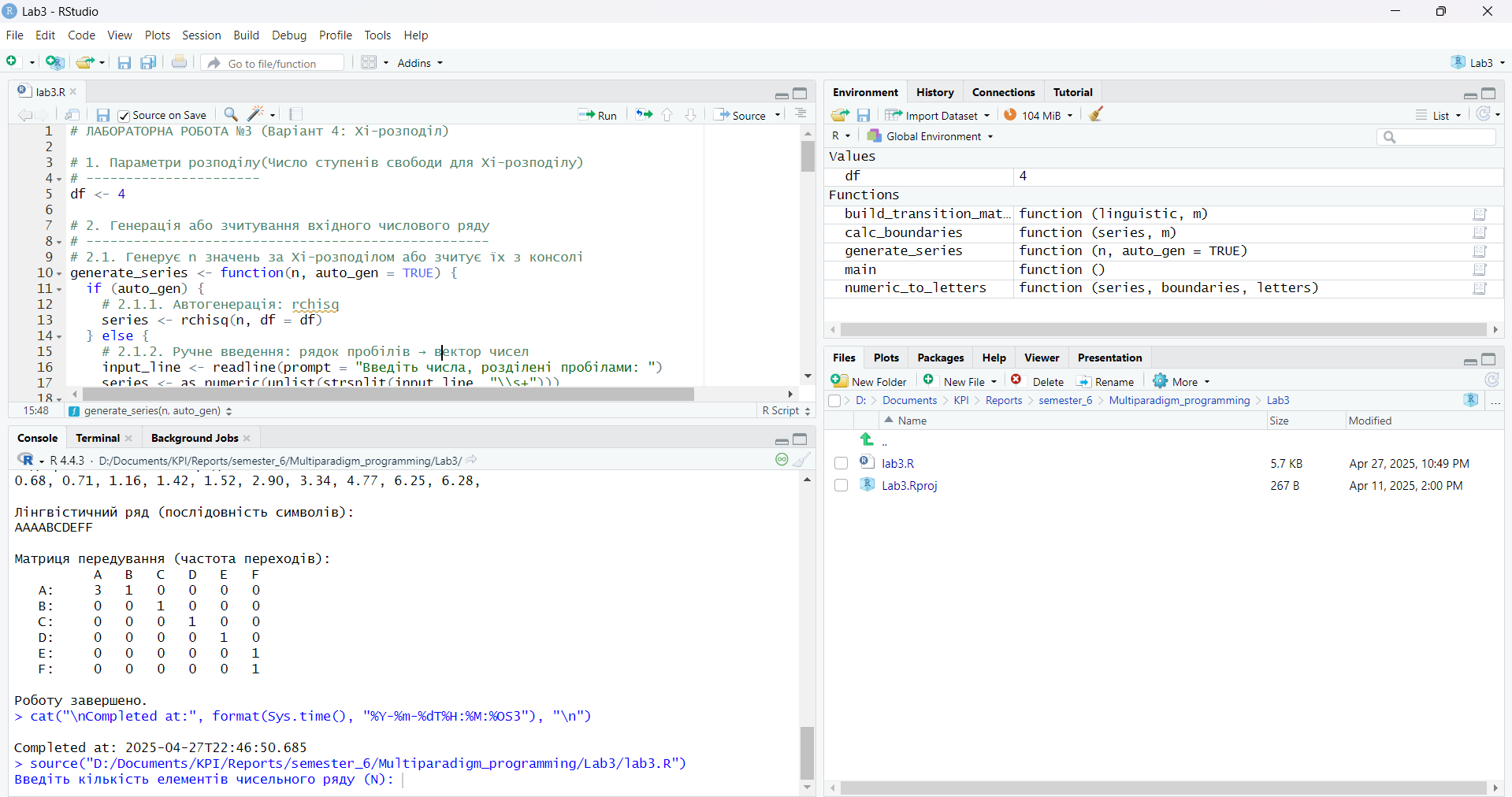
**Виконання роботи**

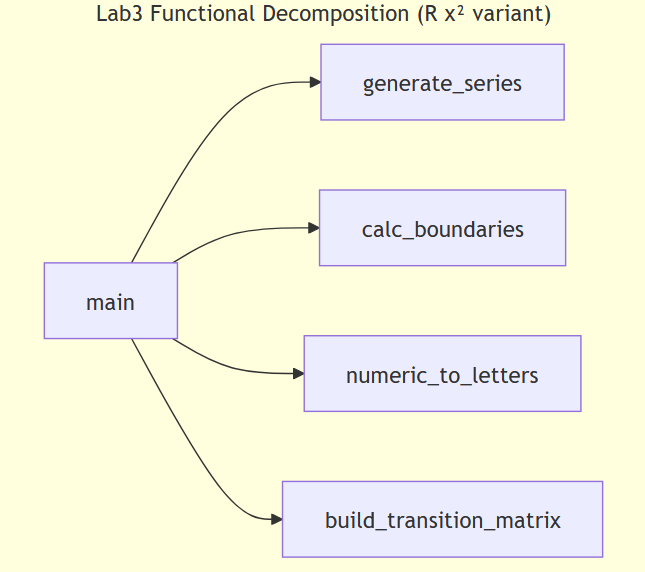
**Нижче наведено логічні блоки програми:**

1. Параметри розподілу(Число ступенів свободи для Χі-розподілу)
2. Генерація або зчитування вхідного числового ряду
   1. Автогенерація: rchisq
   2. Ручне введення: рядок пробілів → вектор чисел
3. Обчислення меж інтервалів за Χі-розподілом
   1. Визначаємо функції розподілу та обернену функцію розподілу (pchisq / qchisq)
   2. Розбиваємо ймовірнісний інтервал [P(min), P(max)] на m частин
   3. За квантілями повертаємо границі
4. Перетворення чисел у символи алфавіту
   1. Для кожного x визначаємо індекс інтервалу через findInterval
   2. Корегуємо вихід за межами [1,m]
   3. Повертаємо вектор символів
5. Побудова матриці переходів (матриці передування)
   1. Ініціалізація матриці нулями m×m
   2. Для кожного кроку i→i+1 інкрементуємо відповідну комірку
6. Головна функція: зчитування, обробка, вивід
   1. Зчитуємо N та перевіряємо
   2. Зчитуємо m (розмір алфавіту) та перевіряємо
   3. Визначаємо, чи автогенерувати ряд
   4. Генеруємо або зчитуємо ряд та сортуючи
   5. Обчислюємо межі інтервалів
   6. Формуємо вектор перших m літер латинського алфавіту
   7. Перетворюємо числовий ряд у літерний
   8. Будуємо матрицю передування
   9. Виводимо результати
7. Виклик головної функції

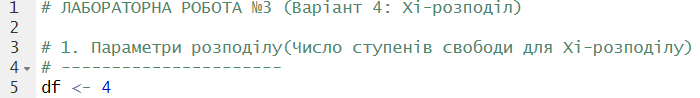
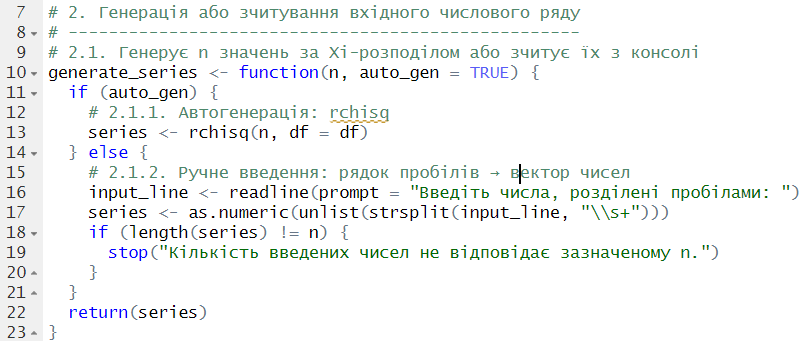
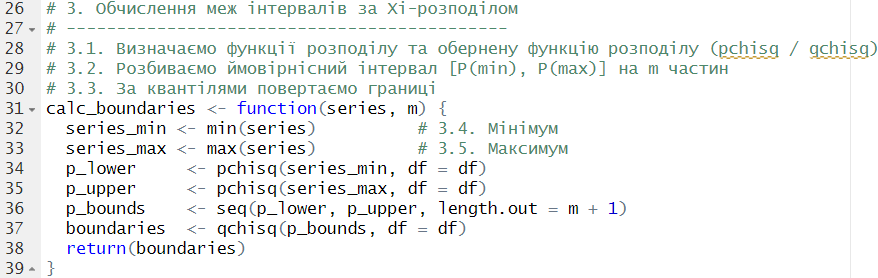
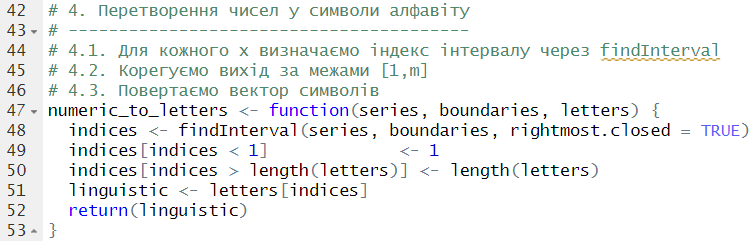
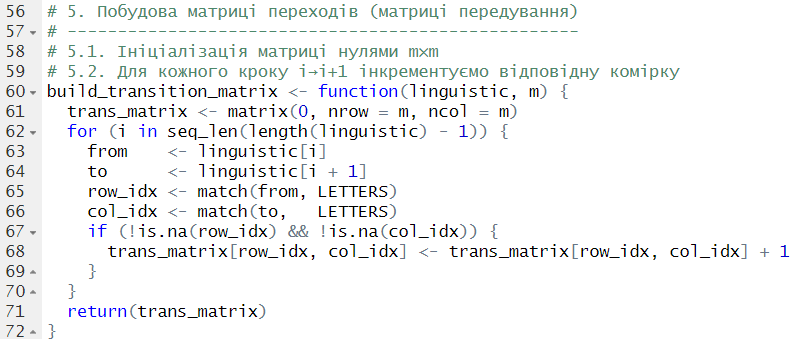
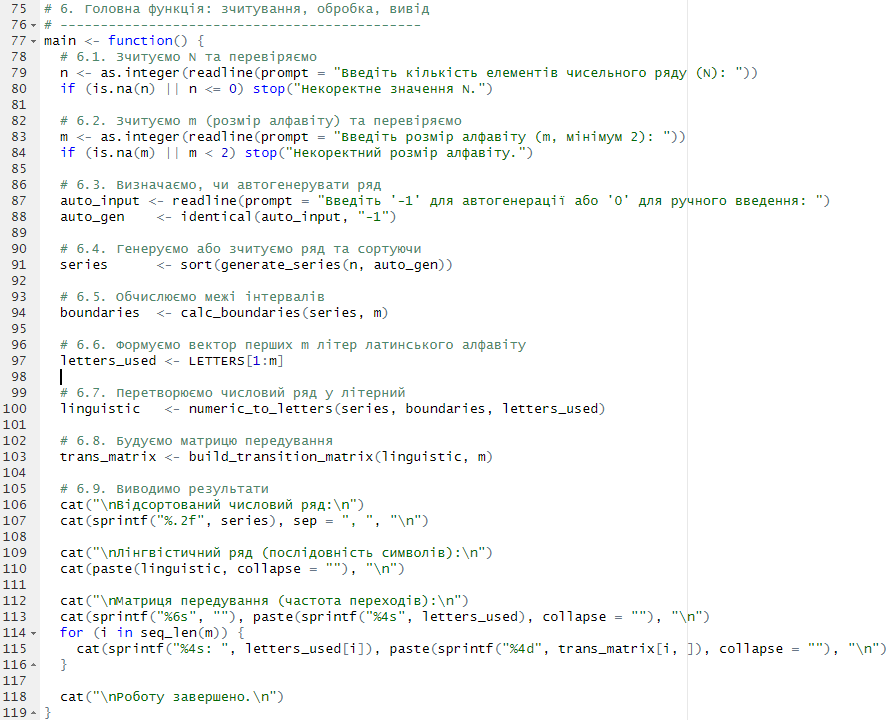
**Давайте розглянемо код для кожного пункту, в якому наведено коментарі українською мовою для кращого розуміння:**

Інтерфейс RStudio for R Lang для запуску програми



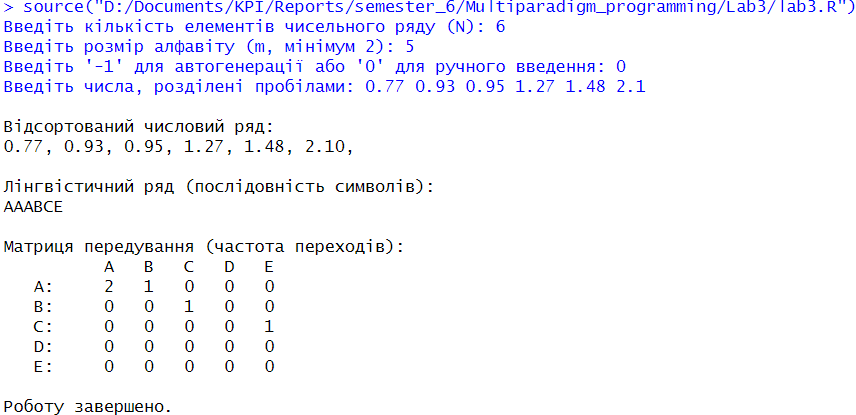
Функціональна декомпозиція і діаграма виконання програми:  


Етапи:

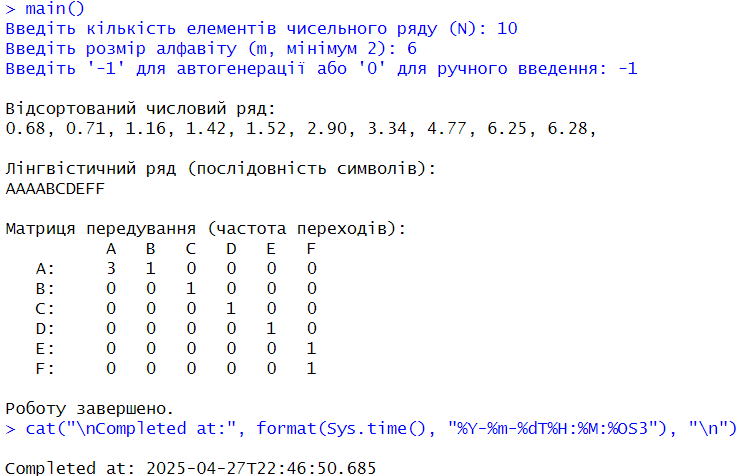
1. 
2. 
3. 
4. 
5. 
6. 
7. 

**Вхідні данні і результат виконання програми:**

Введення значень від руки:



Генерація випадкових значень:



**Висновок**

У рамках лабораторної роботи №3 реалізовано програму на мові R, яка:

* Зчитує з консолі кількість елементів чисельного ряду (N), потужність алфавіту (m) та спосіб формування ряду (ручний ввід або автогенерація за Χі-розподілом з df = 4) ​
* Сортує вхідний числовий ряд у порядку зростання та визначає мінімальне і максимальне значення для подальшого розбиття на інтервали
* Обчислює межі m інтервалів за допомогою рівномірного поділу імовірності P[a,b] на інтервал [F(series\_min), F(series\_max)] і зворотної функції qchisq, тим самим гарантує, що кожен інтервал містить приблизно 1/m від загальної маси розподілу
* Перетворює відсортовані числа у лінгвістичний рядок, зіставляючи кожне значення з відповідною літерою алфавіту відповідно до його інтервалу
* Будує матрицю передування – підрахунок частот переходів між парами сусідніх символів у лінгвістичному рядку
* Виводить у консоль відсортований ряд, межі інтервалів, лінгвістичний рядок та матрицю передування у форматі, аналогічному до програм ЛР №1 та ЛР №2.

Результати тестів (ручний ввід і автогенерація) підтверджують правильність алгоритму:

* Відсортовані значення відповідають очікуваним порядкам.
* Інтервали побудовані так, щоб кожен містив рівну частку імовірності χ²-розподілу.
* Лінгвістичний рядок точно відображає відповідність чисел обраним буквам.
* Матриця передування коректно фіксує кількість появ кожної пари символів.

Таким чином, завдання ЛР №3 виконане повністю й відповідає вимогам лабораторної роботи з мультипарадигменного програмування.

**Додатки**

Код програми:

# ЛАБОРАТОРНА РОБОТА №3 (Варіант 4: Χі-розподіл)

# 1. Параметри розподілу(Число ступенів свободи для Χі-розподілу)

# ----------------------

df <- 4

# 2. Генерація або зчитування вхідного числового ряду

# ---------------------------------------------------

# 2.1. Генерує n значень за Χі-розподілом або зчитує їх з консолі

generate\_series <- function(n, auto\_gen = TRUE) {

if (auto\_gen) {

# 2.1.1. Автогенерація: rchisq

series <- rchisq(n, df = df)

} else {

# 2.1.2. Ручне введення: рядок пробілів → вектор чисел

input\_line <- readline(prompt = "Введіть числа, розділені пробілами: ")

series <- as.numeric(unlist(strsplit(input\_line, "\\s+")))

if (length(series) != n) {

stop("Кількість введених чисел не відповідає зазначеному n.")

}

}

return(series)

}

# 3. Обчислення меж інтервалів за Χі-розподілом

# --------------------------------------------

# 3.1. Визначаємо функції розподілу та обернену функцію розподілу (pchisq / qchisq)

# 3.2. Розбиваємо ймовірнісний інтервал [P(min), P(max)] на m частин

# 3.3. За квантілями повертаємо границі

calc\_boundaries <- function(series, m) {

series\_min <- min(series) # 3.4. Мінімум

series\_max <- max(series) # 3.5. Максимум

p\_lower <- pchisq(series\_min, df = df)

p\_upper <- pchisq(series\_max, df = df)

p\_bounds <- seq(p\_lower, p\_upper, length.out = m + 1)

boundaries <- qchisq(p\_bounds, df = df)

return(boundaries)

}

# 4. Перетворення чисел у символи алфавіту

# ----------------------------------------

# 4.1. Для кожного x визначаємо індекс інтервалу через findInterval

# 4.2. Корегуємо вихід за межами [1,m]

# 4.3. Повертаємо вектор символів

numeric\_to\_letters <- function(series, boundaries, letters) {

indices <- findInterval(series, boundaries, rightmost.closed = TRUE)

indices[indices < 1] <- 1

indices[indices > length(letters)] <- length(letters)

linguistic <- letters[indices]

return(linguistic)

}

# 5. Побудова матриці переходів (матриці передування)

# ---------------------------------------------------

# 5.1. Ініціалізація матриці нулями m×m

# 5.2. Для кожного кроку i→i+1 інкрементуємо відповідну комірку

build\_transition\_matrix <- function(linguistic, m) {

trans\_matrix <- matrix(0, nrow = m, ncol = m)

for (i in seq\_len(length(linguistic) - 1)) {

from <- linguistic[i]

to <- linguistic[i + 1]

row\_idx <- match(from, LETTERS)

col\_idx <- match(to, LETTERS)

if (!is.na(row\_idx) && !is.na(col\_idx)) {

trans\_matrix[row\_idx, col\_idx] <- trans\_matrix[row\_idx, col\_idx] + 1

}

}

return(trans\_matrix)

}

# 6. Головна функція: зчитування, обробка, вивід

# ---------------------------------------------

main <- function() {

# 6.1. Зчитуємо N та перевіряємо

n <- as.integer(readline(prompt = "Введіть кількість елементів чисельного ряду (N): "))

if (is.na(n) || n <= 0) stop("Некоректне значення N.")

# 6.2. Зчитуємо m (розмір алфавіту) та перевіряємо

m <- as.integer(readline(prompt = "Введіть розмір алфавіту (m, мінімум 2): "))

if (is.na(m) || m < 2) stop("Некоректний розмір алфавіту.")

# 6.3. Визначаємо, чи автогенерувати ряд

auto\_input <- readline(prompt = "Введіть '-1' для автогенерації або '0' для ручного введення: ")

auto\_gen <- identical(auto\_input, "-1")

# 6.4. Генеруємо або зчитуємо ряд та сортуючи

series <- sort(generate\_series(n, auto\_gen))

# 6.5. Обчислюємо межі інтервалів

boundaries <- calc\_boundaries(series, m)

# 6.6. Формуємо вектор перших m літер латинського алфавіту

letters\_used <- LETTERS[1:m]

# 6.7. Перетворюємо числовий ряд у літерний

linguistic <- numeric\_to\_letters(series, boundaries, letters\_used)

# 6.8. Будуємо матрицю передування

trans\_matrix <- build\_transition\_matrix(linguistic, m)

# 6.9. Виводимо результати

cat("\nВідсортований числовий ряд:\n")

cat(sprintf("%.2f", series), sep = ", ", "\n")

cat("\nЛінгвістичний ряд (послідовність символів):\n")

cat(paste(linguistic, collapse = ""), "\n")

cat("\nМатриця передування (частота переходів):\n")

cat(sprintf("%6s", ""), paste(sprintf("%4s", letters\_used), collapse = ""), "\n")

for (i in seq\_len(m)) {

cat(sprintf("%4s: ", letters\_used[i]), paste(sprintf("%4d", trans\_matrix[i, ]), collapse = ""), "\n")

}

cat("\nРоботу завершено.\n")

}

# 7. Виклик головної функції

# --------------------------

main()