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 №3

## 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 завдання

Метою лабораторної роботи є реалізація програми для обробки часових рядів з використанням функціонального програмування. Завдання полягає в перетворенні числового часового ряду у лінгвістичний ряд з допомогою дискретизації значень на символи алфавіту.

Важливою умовою є можливість змінювати потужність алфавіту як кількісно (визначати розмір алфавіту), так і за змістом (задати набір символів). Результатом роботи має бути матриця переходів між символами лінгвістичного ряду, що відображає статистику послідовності подій.

Для тестування використовувалися часові ряди з лабораторних робіт №1 та №2, що забезпечує порівнянність результатів.

## 2. Розв’язання задачі

### Алгоритм (процедурний підхід)

1. Зчитування числових даних з CSV-файлу, витягування необхідного стовпця з часовим рядом.
2. Визначення алфавіту:
    a. Якщо задано набір символів — використати їх.
    b. Інакше сформувати алфавіт із літер латинського алфавіту заданої довжини.
3. Рівномірне поділ діапазону значень часового ряду на інтервали за кількістю символів алфавіту.
4. Відображення кожного числового значення у символ алфавіту відповідно до інтервалу.
5. Формування матриці переходів між символами — підрахунок переходів від символу в позиції і до символу в позиції і+1.

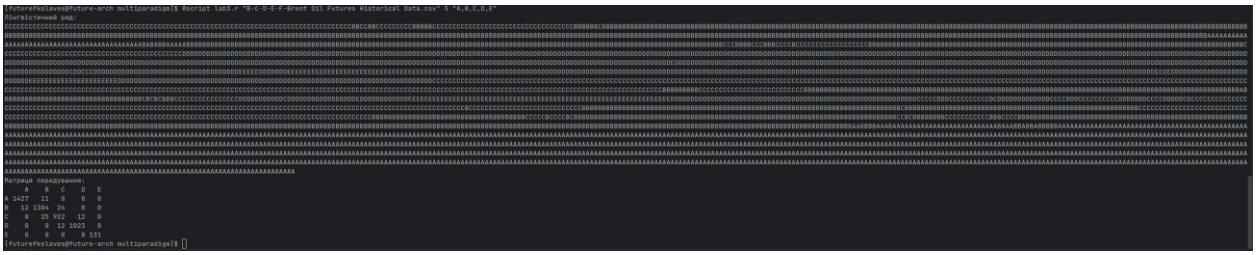
6. Виведення лінгвістичного ряду та матриці переходів.

**Бібліотека функцій і їх взаємозв'язок (функціональний підхід)**

Розв'язок реалізовано у вигляді набору функцій, що послідовно обробляють дані:

- read_csv_column(filename) — зчитування числового ряду з CSV.
- generate_alphabet(size, symbols) — генерація алфавіту за розміром або списком символів.
- get_intervals_uniform(series, alpha_size) — розбиття ряду на рівномірні інтервали.
- map_to_alphabet(series, breaks, alphabet) — відображення чисел у символи.
- build_transition_matrix(ling_series, alphabet) — підрахунок матриці переходів.
- print_ling_series(ling_series) — вивід лінгвістичного ряду.
- print_matrix(matrix) — вивід матриці переходів.

Ці функції викликаються послідовно у головній функції main(). Дані не змінюються глобально, а передаються як аргументи, що відповідає принципам функціонального програмування.



**Лістинг програми:**

```
read_csv_column <- function(filename) {
 if (!file.exists(filename)) {
  cat("Файл не знайдено.\n")
  return(numeric(0))
 }

 data <- read.csv(filename, header = TRUE, stringsAsFactors = FALSE)
 if (!"Price" %in% names(data)) {
  cat("Стовпець 'Price' не знайдено у файлі.\n")
  return(numeric(0))
 }

 price <- as.numeric(gsub(",", "", data$Price))
 clean_price <- price[!is.na(price)]

 if (length(clean_price) == 0) {
```

```
    cat("Файл порожній або не містить числових значень.\n")
  }

  return(clean_price)
}



generate_alphabet <- function(size = NULL, symbols = NULL) {
  if (!is.null(symbols)) {
    return(symbols)
  } else if (!is.null(size)) {
    return(sapply(1:size, function(i) LETTERS[i]))
  } else {
    stop("Потрібно вказати або розмір, або символи алфавіту.")
  }
}



get_intervals_uniform <- function(series, alpha_size) {
  breaks <- seq(min(series), max(series), length.out = alpha_size + 1)
  return(breaks)
}



map_to_alphabet <- function(series, breaks, alphabet) {
  intervals <- cut(series, breaks = breaks, include.lowest = TRUE, labels = alphabet)
  return(as.character(intervals))
}



build_transition_matrix <- function(ling_series, alphabet) {
  size <- length(alphabet)
  matrix <- matrix(0, nrow = size, ncol = size, dimnames = list(alphabet, alphabet))

  for (i in 1:(length(ling_series) - 1)) {
    current <- ling_series[i]
    next_state <- ling_series[i + 1]
    matrix[current, next_state] <- matrix[current, next_state] + 1
  }

  return(matrix)
}



print_ling_series <- function(ling_series) {
  cat("Лінгвістичний ряд:\n")
  cat(paste(ling_series, collapse = ""))
  cat("\n")
}



print_matrix <- function(matrix) {
  cat("Матриця передування:\n")
  print(matrix)
}
```

```
main <- function(
  filename = "B-C-D-E-F-Brent Oil Futures Historical Data.csv",
  alphabet_size = 5,
  alphabet_symbols = NULL
) {
 series <- read_csv_column(filename)
 if (length(series) == 0) return()

 alphabet <- generate_alphabet(size = alphabet_size, symbols = alphabet_symbols)
 breaks <- get_intervals_uniform(series, length(alphabet))
 ling_series <- map_to_alphabet(series, breaks, alphabet)
 matrix <- build_transition_matrix(ling_series, alphabet)

 print_ling_series(ling_series)
 print_matrix(matrix)
}

args <- commandArgs(trailingOnly = TRUE)

if (length(args) >= 1) {
 filename <- args[1]
} else {
 filename <- "B-C-D-E-F-Brent Oil Futures Historical Data.csv"
}

if (length(args) >= 2) {
 alphabet_size <- as.numeric(args[2])
 if (is.na(alphabet_size)) {
  cat("Неправильне значення для alphabet_size, встановлено за замовчуванням 5\n")
  alphabet_size <- 5
 }
} else {
 alphabet_size <- 5
}

if (length(args) >= 3) {
 alphabet_symbols <- unlist(strsplit(args[3], ","))
} else {
 alphabet_symbols <- NULL
}

main(filename, alphabet_size, alphabet_symbols)
```

**Висновки:** У результаті виконання лабораторної роботи було успішно реалізовано перетворення числового ряду в лінгвістичний ланцюжок з урахуванням заданого розподілу ймовірностей та потужності алфавіту. Побудовано матрицю передування, яка відображає частоту переходів між символами і може бути використана для подальшого статистичного аналізу. Отримано практичні навички роботи з файлами, обробки числових даних, маніпуляції масивами, сортуванням, а також реалізації функціональних процедур у мові Common Lisp.