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 мультипарадигменної мови R 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.

Мова програмування

R.

Хід роботи

Програма реалізує перетворення числового ряду у лінгвістичний ланцюжок з наступним побудуванням матриці передування. Алгоритм було реалізовано мовою Fortran стандарту F90. У імплементації використовуються динамічні масиви для роботи з великою кількістю даних у пам’яті та гнучкості програми.

Алгоритм можна поділити на 6 основних етапів:

1. Створення алфавіту довжини визначеної користувачем.
2. Читання числового ряду з файлу у список.
3. Сортування числового списку.
4. Побудова лінгвістичного ряду.
5. Побудова матриці передування.
6. Виведення результатів.

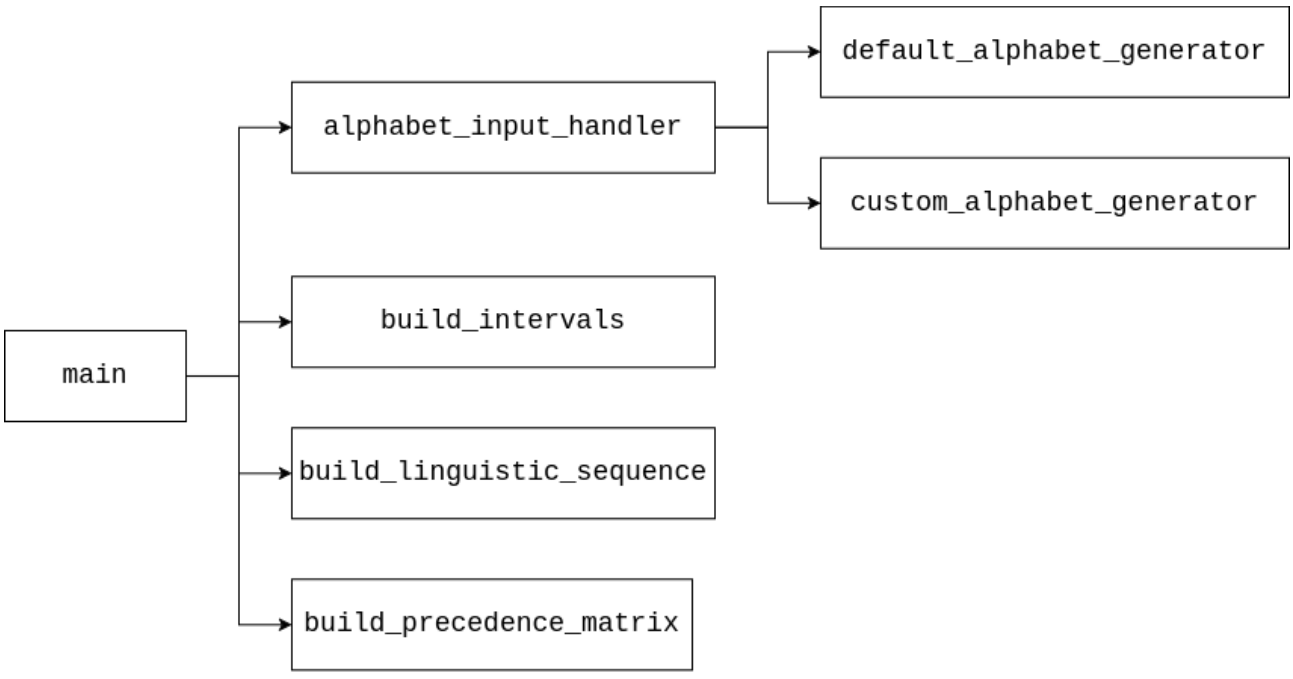

Рисунок 1: Функціональна схема

Результати виконання

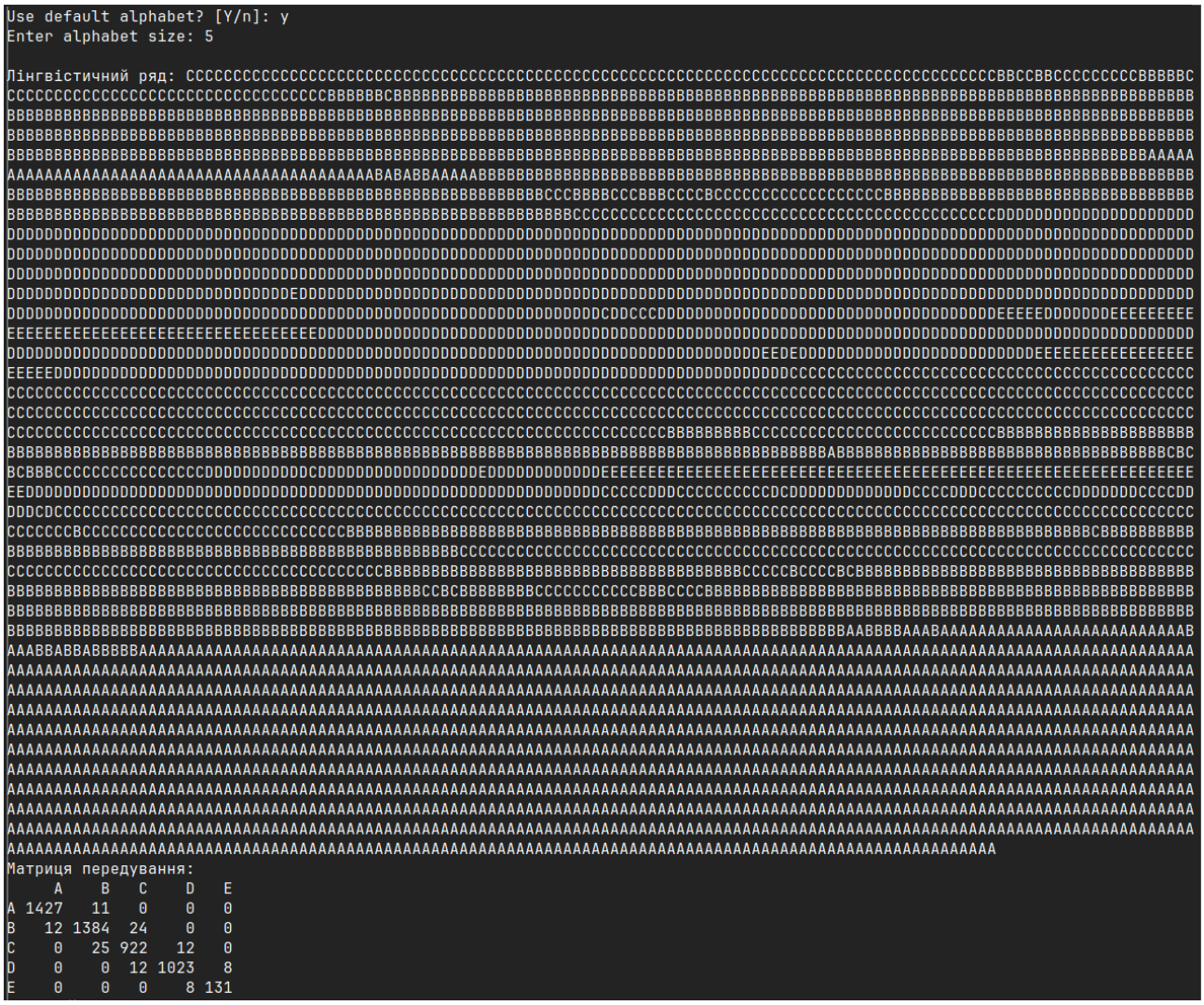

Рисунок 2: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 5 символів)

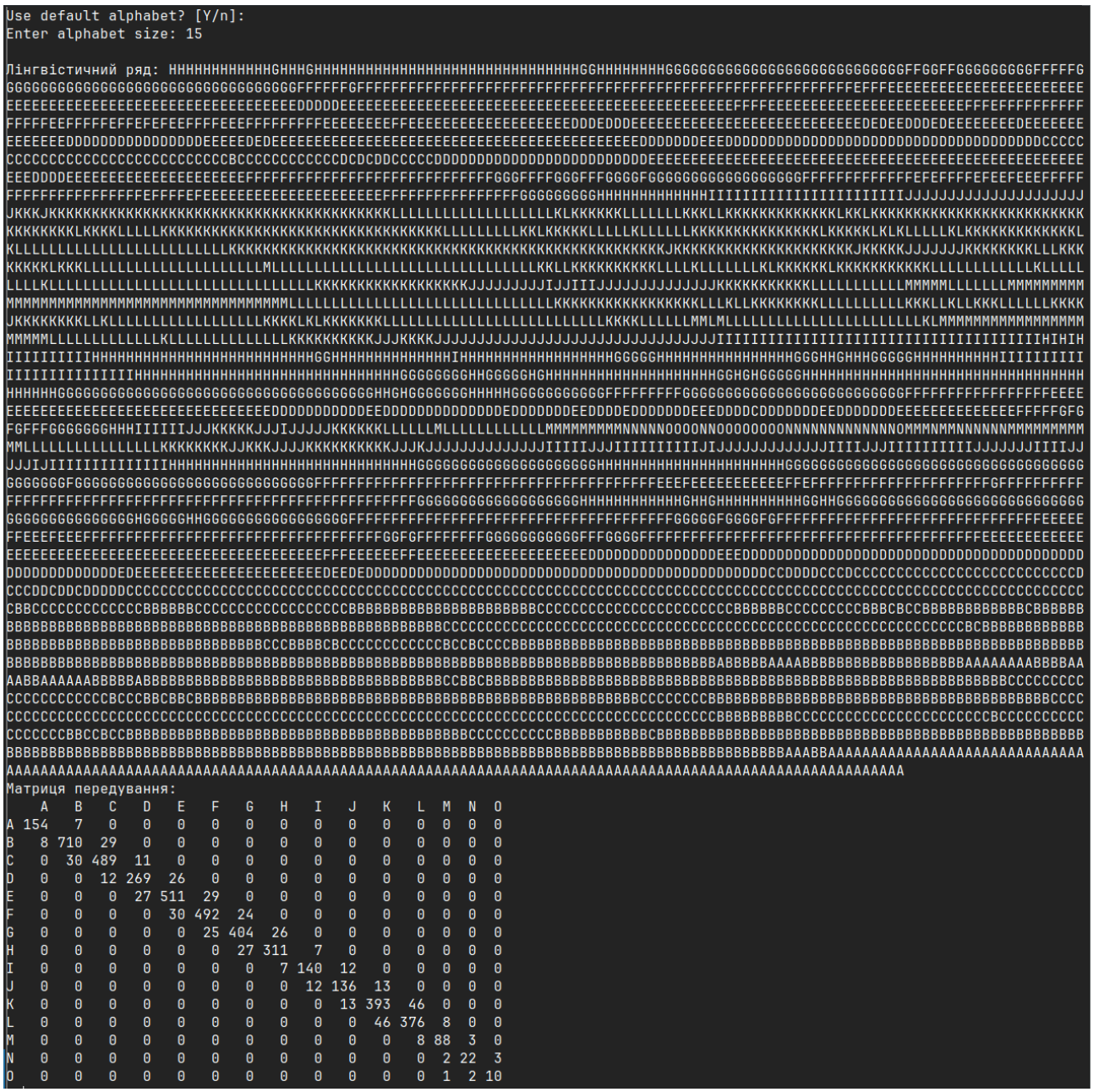

Рисунок 3: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 15 символів)

Лістинг коду

main.r

```r
# Default alphabet generator function
default_alphabet_generator <- function() {
    while (TRUE) {
        input <- readline("Enter alphabet size: ")
        alphabet_size <- as.integer(input)

        if (alphabet_size <= 1) {
            cat("Number is too small, try again\n")
        } else if (alphabet_size < 26L) {
            break
        } else {
            cat("Number is too large, try again\n")
        }
    }

    upper_limit <- alphabet_size + 64
    alphabet <- unlist(strsplit(rawToChar(as.raw(65:upper_limit)), NULL))
    return(list(size = alphabet_size, alphabet = alphabet))
}

# Custom alphabet input function
custom_alphabet_generator <- function() {
    while (TRUE) {
        input <- readline("Enter alphabet size: ")
        alphabet_size <- as.integer(input)

        if (alphabet_size > 1) {
            break
        } else {
            cat("Number is too small, try again\n")
        }
    }

    cat("Input the alphabet letters: ")
    alphabet <- scan(what = "", quiet = TRUE, nmax = alphabet_size)
    return(list(size = alphabet_size, alphabet = alphabet))
}

# Alphabet input type handler
alphabet_input_handler <- function() {
    while (TRUE) {
        input <- readline("Use default alphabet? [Y/n]: ")

        if (input == "" | input == "Y" | input == "y") {
            return(default_alphabet_generator())
        } else if (input == "N" | input == "n") {
            return(custom_alphabet_generator())
        } else {
            cat("Wrong input, try again\n")
        }
    }

    return(input_data)
}

# Interval builder function
build_intervals <- function(series, alphabet_size) {
    min_val <- min(series)
    max_val <- max(series)
    step <- (max_val - min_val) / alphabet_size
    intervals <- seq(min_val, max_val, by = step)

    if (length(intervals) == alphabet_size) {
        intervals <- c(intervals, max_val)
    }
    return(intervals)
}
```

```r

# Linguistic sequence builder function
build_linguistic_sequence <- function(series, intervals, alphabet_data) {
    indices <- findInterval(series, intervals, rightmost.closed = TRUE)
    indices[indices == 0] <- 1
    indices[indices > alphabet_data$size] <- alphabet_data$size
    return(alphabet_data$alphabet[indices])
}

# Precedence matrix builder function
build_precedence_matrix <- function(linguistic_sequence, alphabet_data) {
    matrix <- matrix(0, nrow = alphabet_data$size, ncol = alphabet_data$size, dimnames =
list(alphabet_data$alphabet, alphabet_data$alphabet))

    for (i in 1:(length(linguistic_sequence) - 1)) {
        from <- linguistic_sequence[i]
        to <- linguistic_sequence[i + 1]
        matrix[from, to] <- matrix[from, to] + 1
    }
    return(matrix)
}

# Main function
main <- function() {
    alphabet_data <- alphabet_input_handler()

    numbers <- scan(file = "data.txt", what = numeric(), quiet = TRUE)
    sorted_numbers <- sort(numbers)


    intervals <- build_intervals(sorted_numbers, alphabet_data$size)
    linguistic_sequence <- build_linguistic_sequence(numbers, intervals, alphabet_data)
    precedence_matrix <- build_precedence_matrix(linguistic_sequence, alphabet_data)

    cat("\nЛінгвістичний ряд: ")
    cat(linguistic_sequence, sep = "")
    cat("\n")
    cat("Матриця передування:\n")
    print(precedence_matrix)
}

main()
```