

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 мультипарадигмної мови R реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів з подальшою побудовою матриці передування.

## Вхідні дані

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 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 інтервалів з сортованого числового списку.
5. Побудова лінгвістичного ряду.
6. Побудова матриці передування.
7. Виведення результатів.

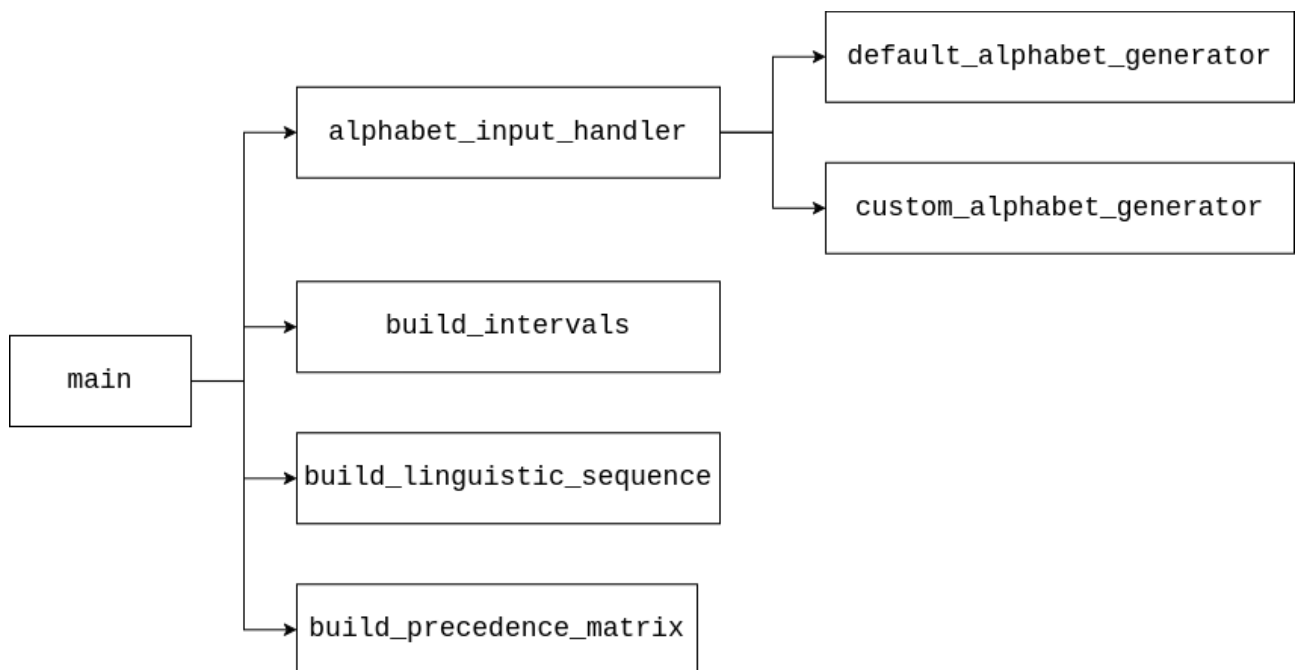


Figure 1: Функціональна схема

[illegible]

[illegible][illegible]

СИМВОЛІВ)

## Лістинг коду

main.r

```
1 # Default alphabet generator function
2 default_alphabet_generator <- function() {
3   while (TRUE) {
4     input <- readline("Enter alphabet size: ")
5     alphabet_size <- as.integer(input)
6
7     if (alphabet_size <= 1) {
8       cat("Number is too small, try again\n")
9     } else if (alphabet_size < 26L) {
10      break
11    } else {
12      cat("Number is too large, try again\n")
13    }
14  }
15
16  upper_limit <- alphabet_size + 64
17  alphabet <- unlist(strsplit(rawToChar(as.raw(65:upper_limit)), NULL))
18  return(list(size = alphabet_size, alphabet = alphabet))
19 }
20
21 # Custom alphabet input function
22 custom_alphabet_generator <- function() {
23   while (TRUE) {
24     input <- readline("Enter alphabet size: ")
25     alphabet_size <- as.integer(input)
26
27     if (alphabet_size > 1) {
28       break
29     } else {
30       cat("Number is too small, try again\n")
31     }
32   }
33
34   cat("Input the alphabet letters: ")
35   alphabet <- scan(what = "", quiet = TRUE, nmax = alphabet_size)
36   return(list(size = alphabet_size, alphabet = alphabet))
37 }
38
39 # Alphabet input type handler
40 alphabet_input_handler <- function() {
41   while (TRUE) {
42     input <- readline("Use default alphabet? [Y/n]: ")
43
44     if (input == "" | input == "Y" | input == "y") {
45       return(default_alphabet_generator())
46     } else if (input == "N" | input == "n") {
47       return(custom_alphabet_generator())
48     } else {
49       cat("Wrong input, try again\n")
50     }
51   }
52
53   return(input_data)
54 }
55
56 # Interval builder function
57 build_intervals <- function(series, alphabet_size) {
58   min_val <- min(series)
59   max_val <- max(series)
60   step <- (max_val - min_val) / alphabet_size
61   intervals <- seq(min_val, max_val, by = step)
62
63   if (length(intervals) == alphabet_size) {
64     intervals <- c(intervals, max_val)
65   }
66   return(intervals)
67 }
```

```

68
69 # Linguistic sequence builder function
70 build_linguistic_sequence <- function(series, intervals, alphabet_data) {
71   indices <- findInterval(series, intervals, rightmost.closed = TRUE)
72   indices[indices == 0] <- 1
73   indices[indices > alphabet_data$size] <- alphabet_data$size
74   return(alphabet_data$alphabet[indices])
75 }
76
77 # Precedence matrix builder function
78 build_precedence_matrix <- function(linguistic_sequence, alphabet_data) {
79   matrix <- matrix(0, nrow = alphabet_data$size, ncol = alphabet_data$size, dimnames =
80     list(alphabet_data$alphabet, alphabet_data$alphabet))
81   for (i in 1:(length(linguistic_sequence) - 1)) {
82     from <- linguistic_sequence[i]
83     to <- linguistic_sequence[i + 1]
84     matrix[from, to] <- matrix[from, to] + 1
85   }
86   return(matrix)
87 }
88
89 # Main function
90 main <- function() {
91   alphabet_data <- alphabet_input_handler()
92
93   numbers <- scan(file = "data.txt", what = numeric(), quiet = TRUE)
94   sorted_numbers <- sort(numbers)
95
96   intervals <- build_intervals(sorted_numbers, alphabet_data$size)
97   linguistic_sequence <- build_linguistic_sequence(numbers, intervals, alphabet_data)
98   precedence_matrix <- build_precedence_matrix(linguistic_sequence, alphabet_data)
99
100   cat("\nЛінгвістичний ряд: ")
101   cat(linguistic_sequence, sep = " ")
102   cat("\n")
103   cat("Матриця передування:\n")
104   print(precedence_matrix)
105 }
106
107
108 main()

```