Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Лабораторна робота № 2
з дисципліни «Мультипарадигмене програмування»

Виконав:
Студент групи ІО-23
Швед А. Д.

Київ - 2025

Завдання

Завдання: на мові функціонального 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.

Мова програмування

Racket.

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.

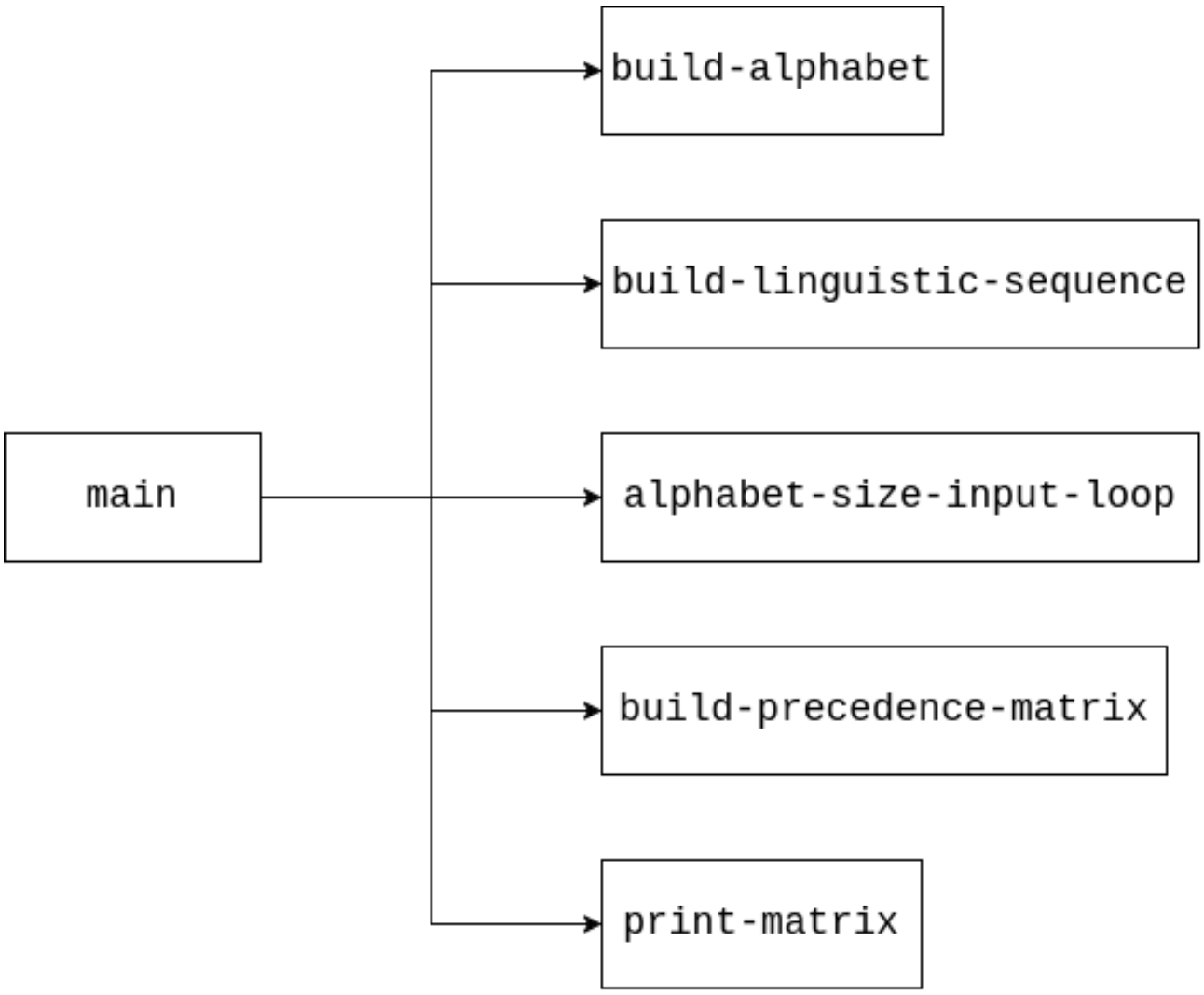

Рисунок 1: Функціональна схема

Результати виконання

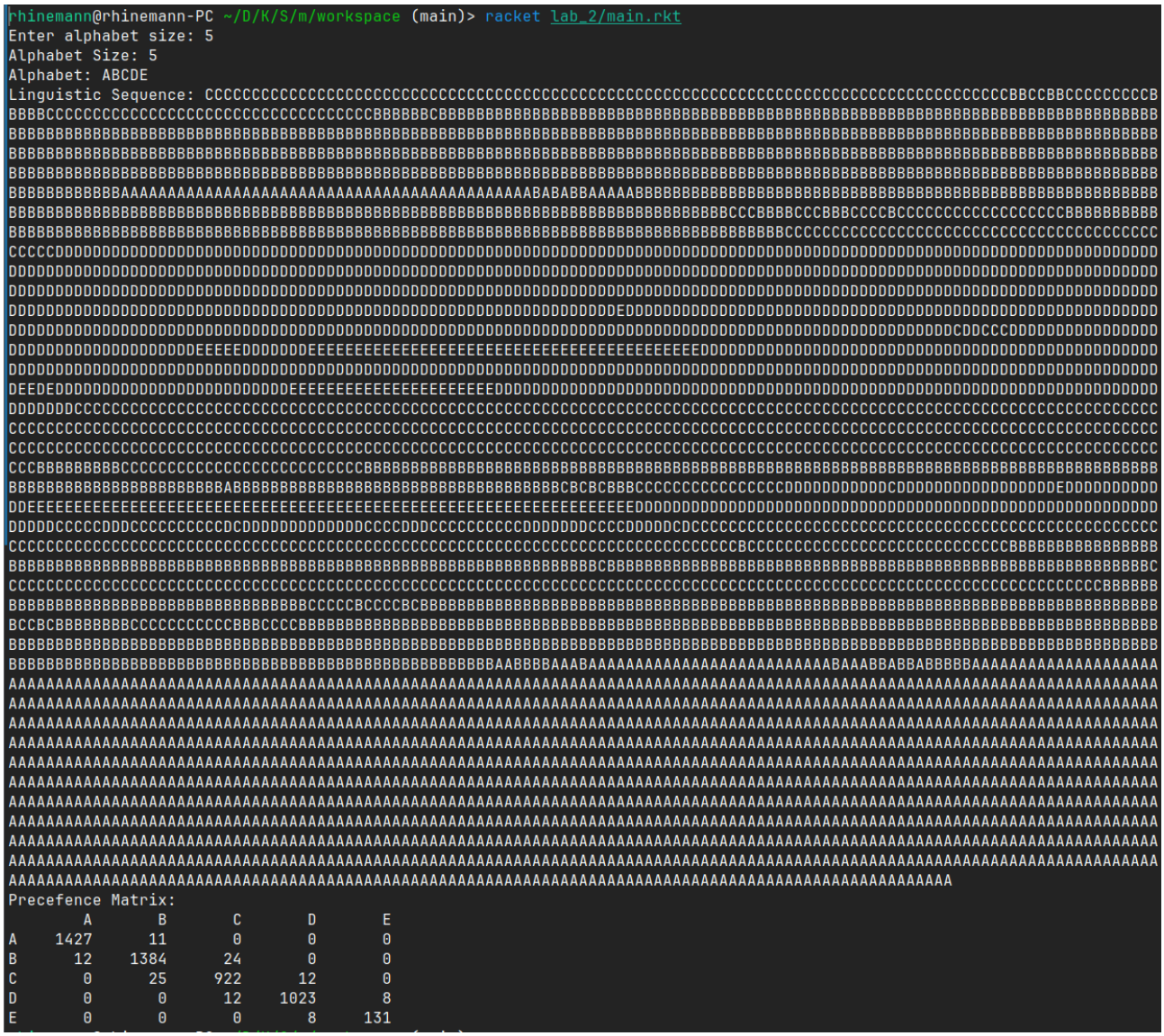

Рисунок 2: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 5 символів)

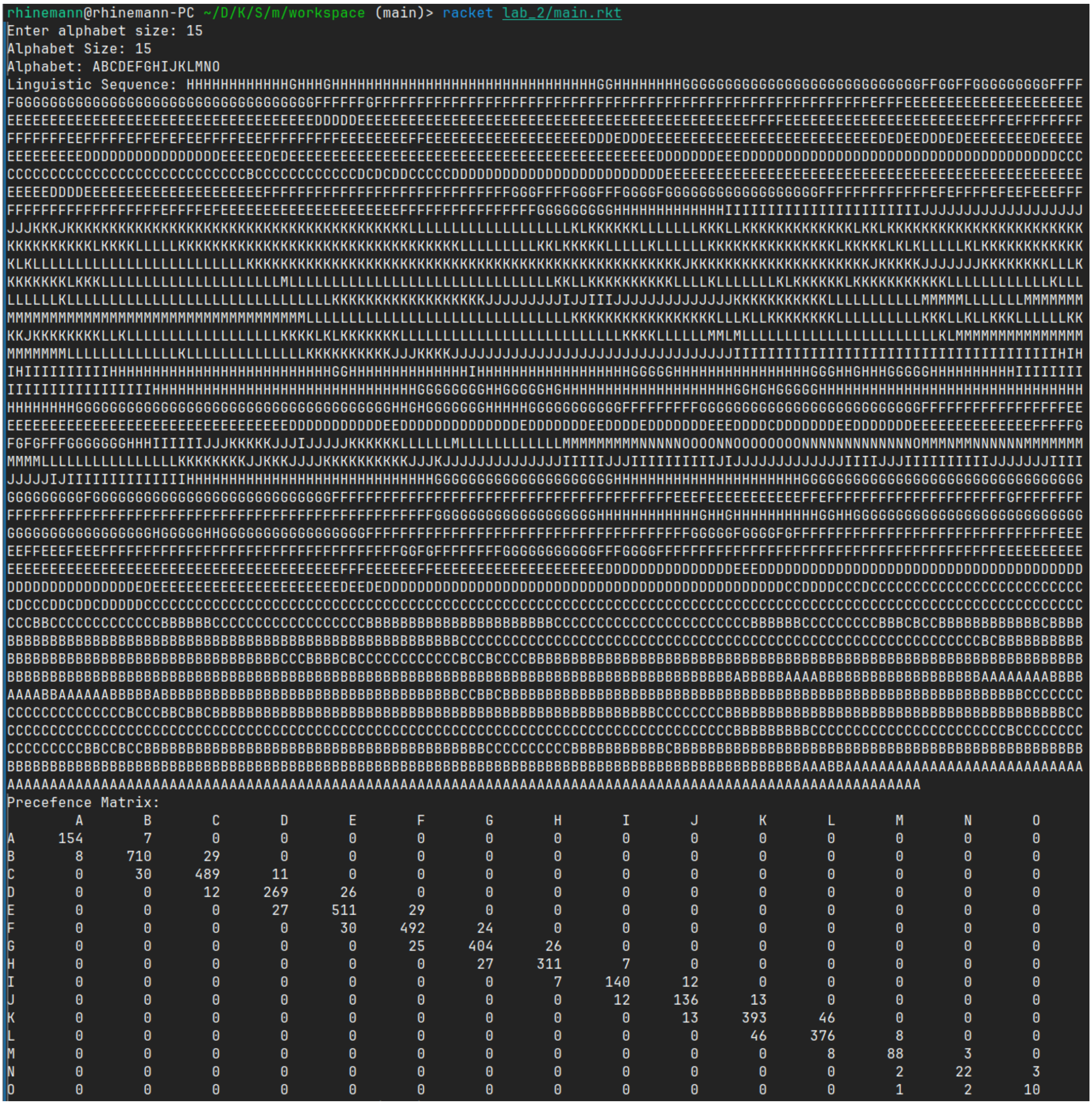

Рисунок 3: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 15 символів)

Лістинг коду

main.rkt

```
#lang racket
(require racket/format)


; Alphabet builder function
(define (build-alphabet size)
  (map (lambda (i) (string (integer->char (+ 65 i)))) (range size)))


; Linguistic sequence builder function
(define (build-linguistic-sequence numbers sorted alphabet)
  (define min_number (first sorted))
  (define max_number (last sorted))
  (define size (length alphabet))
  (define interval (/ (- max_number min_number) size))
  (map (lambda (num)
         (define idx (inexact->exact (ceiling (/ (- num min_number) interval))))
         (list-ref alphabet (cond [(< idx 1) 0]
                                  [(>= idx size) (sub1 size)]
                                  [else (sub1 idx)])))
       numbers))


; Build precedence matrix (vector of vectors)
(define (build-precedence-matrix sequence alphabet)
  (define size (length alphabet))
  (define matrix (build-vector size (lambda (_) (make-vector size 0))))
  (define get-alphabet-index (lambda (val) (index-of alphabet (list-ref sequence val))))

  ; Iterating over the linguistic sequence,
  ; checking the letter precedence and counting the pairs into the matrix
  (for ((i (in-range (sub1 (length sequence)))))
    (let* ([row (get-alphabet-index i)]
           [col (get-alphabet-index (add1 i))]
           [counter (vector-ref (vector-ref matrix row) col)])
      (vector-set! (vector-ref matrix row) col (add1 counter))))

  matrix)


; Matrix printer function
(define (print-matrix matrix alphabet)
  (display " ")
  (for-each (lambda (ch) (display (~a ch #:min-width 8 #:align 'right))) alphabet)
  (newline)
  (for ((i (in-range (vector-length matrix))))
    (printf "~a" (list-ref alphabet i))
    (for ((val (in-vector (vector-ref matrix i))))
      (display (~a val #:min-width 8 #:align 'right)))
    (newline)))


; Alphabet size input loop function
(define (alphabet-size-input-loop)
  (let loop ()
    (display "Enter alphabet size: ")
    (flush-output)
    (define input (read))

    ;; Checking the alphabet size
    (if (> input 26)
        (begin (display "Number is too large, try again\n") (loop))
        input)))

; Main program
(define (main)
  ; Reading alphabet size and constructing the alphabet
```

```
  (define alphabet-size (alphabet-size-input-loop))
  (define alphabet (build-alphabet alphabet-size))

  ; Reading number sequence from the file and copy-sorting it
  (define filename "data.txt")
  (define numbers (file->list filename))
  (define sorted (sort numbers <))

  ; Building a linguistic sequence and a precedence matrix
  (define linguistic-sequence (build-linguistic-sequence numbers sorted alphabet))
  (define precedence-matrix (build-precedence-matrix linguistic-sequence alphabet))

  ; Outputting the data
  (printf "Alphabet Size: ~a\n" alphabet-size)
  (printf "Alphabet: ~a\n" (string-join alphabet ""))
  (printf "Linguistic Sequence: ~a\n" (string-join linguistic-sequence ""))
  (printf "Precefence Matrix:\n")
  (print-matrix precedence-matrix alphabet))

(main)
```