

Міністерство освіти і науки України  
Національний технічний університет України «КПІ»  
імені Ігоря Сікорського

ЗВІТ  
з лабораторної роботи №5  
з дисципліни «Мультипарадигмне програмування»

Виконав:  
Студент 3 курсу кафедри ОТ ФІОТ,  
Навчальної групи ІО-23  
Прохоренко Артем

Київ 2025

**Завдання:** за допомогою продукційного програмування реалізувати перетворення чисельного ряду до лінгвістичного ланцюжка за певним розподілом ймовірностей потрапляння значень до інтервалів.

**Вхідні данні:** чисельний ряд, вид розподілу ймовірностей, потужність алфавіту.

**Вихідні дані:** лінгвістичний ряд та матриця передування.

**Мова програмування:** CLIPS.

**Варіант:** 20

**Розподіл ймовірностей:** Розбиття на рівні інтервали

### **Хід розв'язання задачі:**

#### **Кроки реалізації:**

1. Дані зчитуються з текстового файлу за допомогою функції `read-numbers-from-file`. Кожне значення файлу додається до факту типу `number`, що дозволяє зберігати числовий ряд для подальшої обробки.
2. Числовий ряд сортується за допомогою функції `insert-in-order`. Всі числа додаються до списку, а потім цей список сортується для подальшої побудови інтервалів.
3. Після сортування числового ряду, інтервали для кожного символу алфавіту розраховуються за допомогою функції `assign-symbols`. Кількість інтервалів визначається кількістю символів алфавіту. Кожне число в числовому ряді потрапляє в один з інтервалів, і йому присвоюється відповідний символ з алфавіту.
4. Після того як числа прив'язуються до символів, створюється лінгвістичний ряд, що відображає розподіл чисел по символах.
5. На основі лінгвістичного ряду будується матриця передування, що показує кількість випадків, коли одна буква слідує за іншою. Це дозволяє побачити структуру і закономірності в ланцюжку.
6. Виведення результатів

Я помітив, що при реалізації цієї роботи на CLIPS обробка 5000 значень відбувалася значно довше, ніж у прикладах з попередніх лабораторних робіт. Це відбулося через кілька причин. По-перше, CLIPS не має вбудованого швидкого сортування, тому для сортування числового ряду я використовував власну реалізацію через алгоритм вставки, що є досить повільним при великих масивах. По-друге, в CLIPS кожен факт обробляється окремо, що при великій кількості значень призводить до великого навантаження на процесор. Оскільки CLIPS більше орієнтований на невеликі обсяги даних і фактично працює з базою фактів, обробка великих масивів значень стає значно менш ефективною порівняно з іншими мовами програмування, що підтримують більш оптимізовані алгоритми та структури даних.

## Результати виконання

**Перший числовий ряд (5 значень):** 3.2, 7.8, 1.5, 9.0, 4.6

```
CLIPS> (load "D:/Vidzet/ΦNOT/6/MPP/5/5.clp")
```

\*\*\*\*\*

TRUE

CLIPS&gt; (reset)

```
CLIPS> (read-numbers-from-file "D:/Vidzet/ΦΙΟΤ/6/MPP/5/1.txt")
```

CLIPS&gt; (run)

Лінгвістичний ряд: а с а с b

Матриця передування:

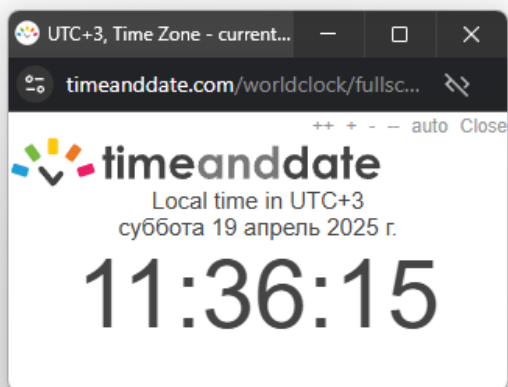
	a	b	c
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	1	1
7	1	1	1
8	1	1	1
9	1	1	1
10	1	1	1
11	1	1	1
12	1	1	1
13	1	1	1
14	1	1	1
15	1	1	1
16	1	1	1
17	1	1	1
18	1	1	1
19	1	1	1
20	1	1	1
21	1	1	1
22	1	1	1
23	1	1	1
24	1	1	1
25	1	1	1
26	1	1	1
27	1	1	1
28	1	1	1
29	1	1	1
30	1	1	1
31	1	1	1
32	1	1	1
33	1	1	1
34	1	1	1
35	1	1	1
36	1	1	1
37	1	1	1
38	1	1	1
39	1	1	1
40	1	1	1
41	1	1	1
42	1	1	1
43	1	1	1
44	1	1	1
45	1	1	1
46	1	1	1
47	1	1	1
48	1	1	1
49	1	1	1
50	1	1	1
51	1	1	1
52	1	1	1
53	1	1	1
54	1	1	1
55	1	1	1
56	1	1	1
57	1	1	1
58	1	1	1
59	1	1	1
60	1	1	1
61	1	1	1
62	1	1	1
63	1	1	1
64	1	1	1
65	1	1	1
66	1	1	1
67	1	1	1
68	1	1	1
69	1	1	1
70	1	1	1
71	1	1	1
72	1	1	1
73	1	1	1
74	1	1	1
75	1	1	1
76	1	1	1
77	1	1	1
78	1	1	1
79	1	1	1
80	1	1	1
81	1	1	1
82	1	1	1
83	1	1	1
84	1	1	1
85	1	1	1
86	1	1	1
87	1	1	1
88	1	1	1
89	1	1	1
90	1	1	1
91	1	1	1
92	1	1	1
93	1	1	1
94	1	1	1
95	1	1	1
96	1	1	1
97	1	1	1
98	1	1	1
99	1	1	1
100	1	1	1

a: 0 0 2

b: 0 0 0

c: 1 1 0

CLIPS&gt;



**Другий числовий ряд (5000 значень - 5 символів):** B-C-D-E-Gold Futures Historical Data (Price)

CLIPS (6.4.2 1/14/25)

CLIPS&gt; (clear)

```
CLIPS> (load "D:/Vidzet/0NOT/6/MPP/5/5.clp")
```

%%%%!! ! ! \* \* \* \* \*

TRUE

CLIPS&gt; (reset)

```
CLIPS> (read-numbers-from-file "D:/Vidzet/ΦNOT/6/MPP/5/2.txt")
```

CLIPS&gt;

(run)

Лінгвістичний ряд: с

Матриця передування:

a   b   c   d   e

```
a: 1159 5 0 0 0
```

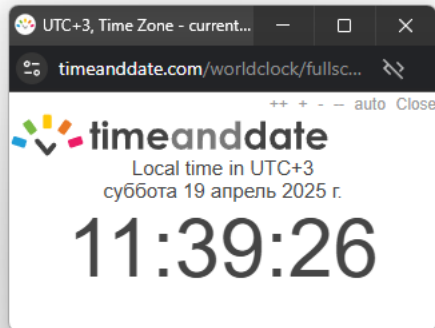
```
b:  6  550  13  0  0
```

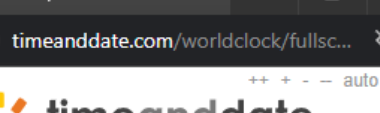
c: 0 14 762 24 0

d: 0 0 24 816 9

e: 0 0 0 9 390

CLIPS&gt;





## Лістинг програмного тексту

```
;;;;;;;;;;;;;;
;; Шаблони
;;;;;;;;;;;;;;

(deftemplate number (slot value)) ;; Шаблон для збереження чисел
(deftemplate symbol-mapped (slot value) (slot symbol)) ;; Шаблон для
відображення чисел на символи
(deftemplate transition
  (slot from) ;; Шаблон для зберігання переходів
  (slot to)
  (slot pair-id))
(deftemplate status (slot stage)) ;; Шаблон для збереження статусу
етапів
(deftemplate sorted-list (multislot values)) ;; Шаблон для збереження
відсортованого ряду

;;;;;;;;;;;;;;
;; Функції
;;;;;;;;;;;;;;

(deffunction read-numbers-from-file (?filename)
  ;; Функція для зчитування чисел з файлу
  (bind ?opened (open ?filename filein))
  (if (eq ?opened FALSE) then
    (printout t "Не вдалося відкрити файл: " ?filename crlf)
    (return))
  (loop-for-count (?i 1 10000)
    (bind ?val (read filein))
    (if (eq ?val EOF) then (return))
    (assert (number (value ?val)))) ;; Зчитуємо значення та додаємо
їх як факти
  (close filein)
  (printout t "Дані зчитано з файлу: " ?filename crlf)
)

(deffunction insert-in-order (?val ?sorted)
  ;; Функція для вставки значення в відсортований ряд
  (bind ?result (create$))
  (bind ?inserted FALSE)
  (foreach ?x ?sorted
    (if (and (not ?inserted) (< ?val ?x)) then
      (bind ?result (create$ ?result ?val))
```

```

    (bind ?inserted TRUE))
    (bind ?result (create$ ?result ?x)))
  (if (not ?inserted) then
    (bind ?result (create$ ?result ?val)))
  ?result)

(defun assign-symbols (?vals ?alphabet)
  ;; Функція для відображення чисел на символи
  (bind ?count (length$ ?alphabet)) ;; Отримуємо кількість символів
  (bind ?min (nth$ 1 ?vals)) ;; Мінімальне значення
  (bind ?max (nth$ (length$ ?vals) ?vals)) ;; Максимальне значення
  (bind ?step (/ (- ?max ?min) ?count)) ;; Ширина інтервалу

  ;; Створення інтервалів
  (bind ?intervals (create$))
  (bind ?i 1)
  (while (<= ?i ?count)
    (bind ?start (+ ?min (* (- ?i 1) ?step)))
    (bind ?end (+ ?start ?step))
    (bind ?sym (nth$ ?i ?alphabet)) ;; Призначаємо символ
    (bind ?intervals (create$ ?intervals ?start ?end ?sym))
    (bind ?i (+ ?i 1)))

  ;; Привязка чисел до символів
  (do-for-all-facts ((?n number)) TRUE
    (bind ?v ?n:value)
    (bind ?j 0)
    (while (< ?j (* ?count 3))
      (bind ?a (nth$ (+ ?j 1) ?intervals))
      (bind ?b (nth$ (+ ?j 2) ?intervals))
      (bind ?s (nth$ (+ ?j 3) ?intervals))
      (if (or (and (>= ?v ?a) (< ?v ?b))
              (and (= ?v ?max) (= ?b ?max))) then
        (assert (symbol-mapped (value ?v) (symbol ?s))) ;;
        Відображаємо символ
        (bind ?j (* ?count 3))) ; вихід з циклу
      (bind ?j (+ ?j 3))))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Правила
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defrule sort-values
  ;; Правило для сортування чисел

```

```

=>
(bind ?raw (create$))
(do-for-all-facts ((?n number)) TRUE
  (bind ?raw (create$ ?raw ?n:value)))
(bind ?sorted (create$))
(foreach ?val ?raw
  (bind ?sorted (insert-in-order ?val ?sorted))) ;; Використовуємо
функцію для сортування
(assert (sorted-list (values ?sorted))) ;; Зберігаємо відсортований
ряд
(assert (status (stage ready))) ;; Статус готовий
)

(defrule map-values-to-symbols
  ?sorted <- (sorted-list (values $?vals)) ;; Перевіряємо, що ряд
відсортовано
  (status (stage ready)) ;; Перевіряємо, чи готовий етап
=>
  (bind ?alphabet (create$ a b c d e f g h i j)) ;; Алфавіт можна
змінювати
  (assign-symbols ?vals ?alphabet) ;; Викликаємо функцію для
відображення
)

(defrule build-transitions
  ;; Правило для побудови матриці передування
=>
  (bind ?all (find-all-facts ((?s symbol-mapped)) TRUE)) ;; Отримуємо
всі факти
  (bind ?len (length$ ?all)) ;; Отримуємо кількість фактів
  (loop-for-count (?i 1 (- ?len 1))
    (bind ?from (fact-slot-value (nth$ ?i ?all) symbol)) ;; Отримуємо
символ з поточного факту
    (bind ?to (fact-slot-value (nth$ (+ ?i 1) ?all) symbol)) ;;
Отримуємо символ з наступного факту
    (assert (transition (from ?from) (to ?to) (pair-id ?i)))) ;;
Створюємо перехід між символами
)

(defrule print-symbol-sequence
  ;; Правило для виведення лінгвістичного ряду
=>
  (printout t crlf "Лінгвістичний ряд: ")
  (do-for-all-facts ((?s symbol-mapped)) TRUE
    (printout t ?s:symbol " ")) ;; Виводимо кожен символ

```



```

лінгвістичного ряду
  (printout t crlf)
)

(defrule print-transition-matrix
  ;; Правило для виведення матриці передування
  =>
  (bind ?alphabet (create$ a b c d e f g h i j)) ;; Алфавіт можна
змінювати
  (printout t crlf "Матриця передування:" crlf)
  (printout t " ")
  (foreach ?col ?alphabet
    (printout t " " ?col)) ;; Виводимо заголовок матриці
  (printout t crlf)

  (foreach ?row ?alphabet
    (printout t ?row ": ")
    (foreach ?col ?alphabet
      (bind ?count (length$ (find-all-facts ((?t transition))
                                                (and (eq ?t:from ?row) (eq ?t:to
?col)))) )
      (printout t " " ?count)) ;; Виводимо кількість переходів для
кожної пари символів
    (printout t crlf))
  )

```