

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

Лабораторна робота № 5
з дисципліни «Мультипарадигмене програмування»

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

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.

Хід роботи

Програма реалізує перетворення числового ряду у лінгвістичний ланцюжок з наступним побудуванням матриці передування.

Правила

1. Дані зчитуються з текстового файлу за допомогою функції load-numbers. Кожне значення файлу додається до факту numbers, що дозволяє зберігати числовий ряд для подальшої обробки.
2. Числовий ряд сортується за допомогою функції insert-ordered. Числа додаються до списку, після чого список сортується для побудови інтервалів.
3. Після сортування числового ряду, інтервали для алфавіту розраховуються за допомогою функції assign-symbols. Кількість інтервалів визначається кількістю символів алфавіту. Кожне число в числовому ряді потрапляє в один з інтервалів, і відображається у відповідний символ з алфавіту.
4. Після того як числа прив'язуються до символів, створюється лінгвістичний ряд, що відображає розподіл чисел по символах.
5. За лінгвістичним рядом створюється матриця передування, що показує кількість випадків, коли одна буква слідує за іншою.
6. Виведення результатів

При виконанні роботи було помічено, що обробка 5000 значень на CLIPS відбувалася значно довше, ніж у попередніх лабораторних роботах. На це є кілька причин.

По-перше, CLIPS не має вбудованого швидкого сортування, тому для сортування числового ряду було написано код для сортування алгоритмом вставки, що є дуже повільним на великих масивах.

По-друге, обробка кожного факту у CLIPS відбувається окремо, що призводить до значного навантаження на процесор за великої кількості значень.

Оскільки CLIPS орієнтований на невеликі обсяги даних і обробку бази фактів лімітованого розміру, робота з великими обсягами даних стає значно менш ефективною порівняно з

іншими мовами програмування, що підтримують більш оптимальні для цього алгоритми та структури даних.

Результати виконання

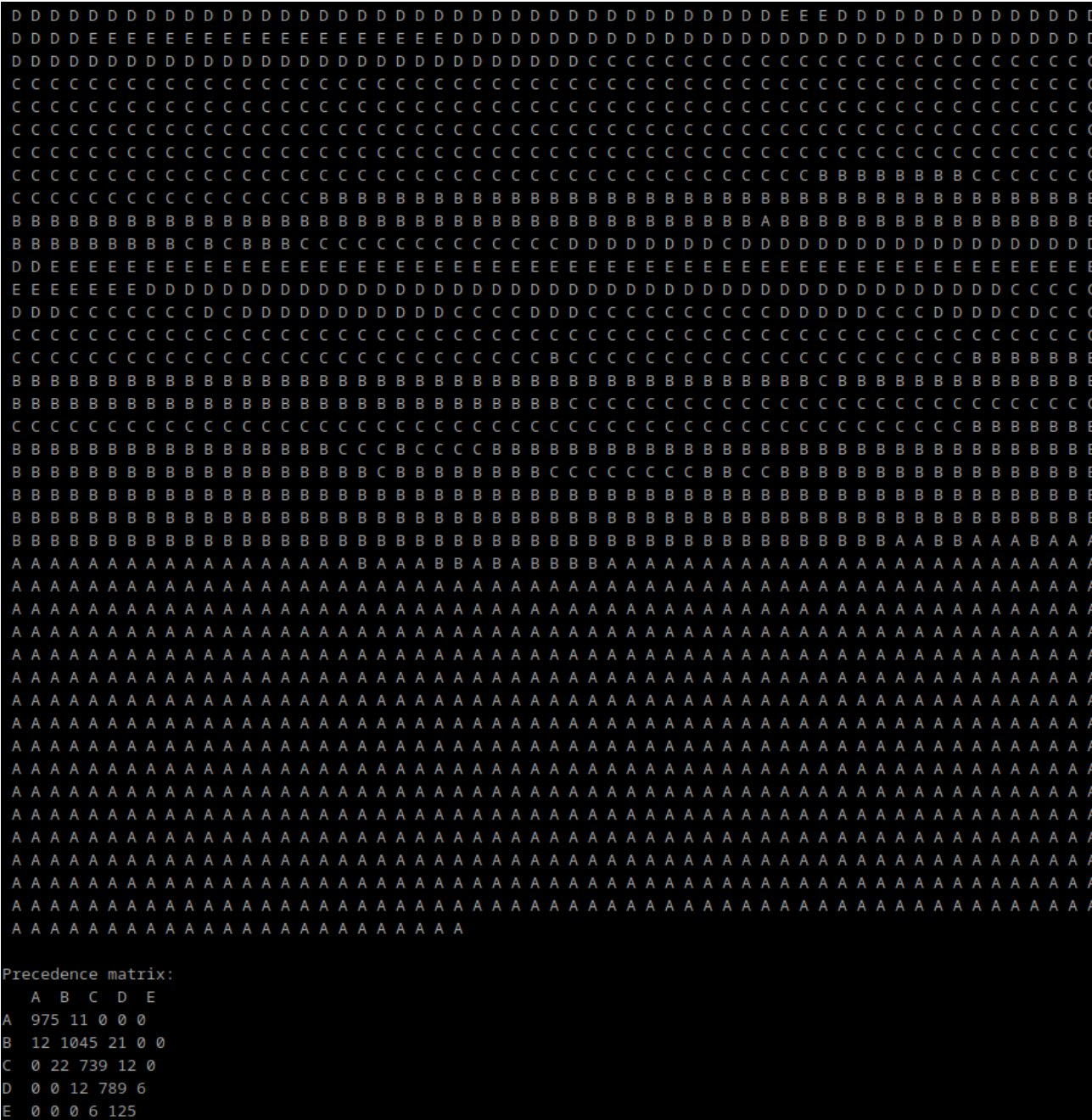


Рисунок 1: Перший числовий ряд В-С-D-E-F-Brent Oil Futures Historical Data (5000 значень - 5 символів)

```

E E E E E F F F F G F G F F F G G G G G G H H I I I I I J J J K K K J J I J J J K K K K L L L L L L L L L L L
L L M M M M M M M N N N N N O O O O N N O O O O O O N N N N N N N N N N N N N N N N N N N N N M M M M M
M M M M M M M L L L L L L L L L L K K K K J J J K K K K J J J K K K K K K K J J J K J J J J J J J J J I I I I
I J J J I I I I I I I J J J J J J J J J J J I I I I J J J I I I I I I I J J J J I I I J J J I J J I
I I I I I I I I I I H H H H H H H H H H H H H H H H H H H G G G G G G G G G G G G H H H H H H H H H H
H H H H G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G F F F F F F F F
F F F F F F F F F F F F F F F F F F F F F F E E F E E E E E E F F F F F F F F F F F F F F F F F F
F F F F F F F F F F F F F F F F F F F F F F F F G G G G G G G G G G G G G H H H H H H H H H H G G H H
H H H H H H H G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G G F F F F F F
F F F F F F F F F F F F F F F F G G G F G G G F F F F F F F F F F F F F F F F F F E E E E F E E E E
F F F F F F F F F F F F F F F F F F G F F F F F F F F G G G G G G G F F G G F F F F F F F F F F F F
F F F F F F E E E E E E E E E E E E E E E E E E E E E E E E E E E E F E E E E E E E E E E E E E E E
E E E D D D D D D D D D D E D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D D
D E E E E E E E E E E D D E D D D D D D D D D D D D D D D D D D D D D D D D D D D D D C C C C C C C C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C C C C C C B B C C C C C C C C C C B B B B B B C C C C C C C C C C C C C C C C B B B B B B B B B B B B
B C C C C C C C C C C C C C C C B B B B B B C C C C C C B B B C C C B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
C C C C C C C C C C C C C C C C C C C C C C C C C C C C B C B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B C B C C C C C B C B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B A B B B B B A A A A B B B B B B B B B B B B B B A A A A A A A B B B A A A B
A A A A B B B B A B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B C C C C C C C C C C C C B B B B B B B B B B B B B B B B B B B B B B B C C B B
B B B B B B B B B B B B B B B B C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C C
C C C C C C C C C C C C C C C B B B B C C C C C C C C C C C C C C C C B B C C B B C C B B B B B B B B B B
B B B B B C C B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B B
B B B B B B B B B B B B B B B B B A A A B B A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A A

```

```

Precedence matrix:
  A B C D E F G H I J K L M N O
A 131 7 0 0 0 0 0 0 0 0 0 0 0 0 0
B 8 451 18 0 0 0 0 0 0 0 0 0 0 0 0
C 0 19 341 11 0 0 0 0 0 0 0 0 0 0 0
D 0 0 12 220 23 0 0 0 0 0 0 0 0 0 0
E 0 0 0 24 372 21 0 0 0 0 0 0 0 0 0
F 0 0 0 0 22 363 21 0 0 0 0 0 0 0 0
G 0 0 0 0 0 22 305 21 0 0 0 0 0 0 0
H 0 0 0 0 0 0 22 248 7 0 0 0 0 0 0
I 0 0 0 0 0 0 0 7 129 12 0 0 0 0 0
J 0 0 0 0 0 0 0 0 12 120 13 0 0 0 0
K 0 0 0 0 0 0 0 0 0 13 306 28 1 0 0
L 0 0 0 0 0 0 0 0 0 0 29 280 5 0 0
M 0 0 0 0 0 0 0 0 0 0 0 6 84 3 0
N 0 0 0 0 0 0 0 0 0 0 0 2 22 3
O 0 0 0 0 0 0 0 0 0 0 0 0 1 2 9
CLIPS>

```

Рисунок 2: Перший числовий ряд B-C-D-E-F-Brent Oil Futures Historical Data (5000 значень - 15 символів)

Лістинг коду

main.clp

```
1  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
2  ;; Templates
3  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
4  (deftemplate numbers (slot value)) ;; Template for a number series
5  (deftemplate symbol-mapped (slot value) (slot symbol)) ;; Template for mapping numbers to
   symbols
6  (deftemplate linguistic-pair
7    (slot from) ;; Template for linguistic pairs
8    (slot to)
9    (slot pair-id))
10 (deftemplate status (slot stage)) ;; Template to save statuses for every step
11 (deftemplate sorted-list (multislot values)) ;; Template for a sorted list
12
13
14 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
15 ;; Globals
16 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
17
18 ;; Alphabet definition
19 (defglobal
20   ?*alphabet* = (create$ A B C D E F G H I J K L M N O))
21
22
23 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
24 ;; Functions
25 ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
26
27 ;; File reading function
28 (deffunction load-numbers (?filename)
29   (bind ?opened (open ?filename filein))
30   (if (eq ?opened FALSE) then
31     (printout t "Could not open " ?filename crlf)
32     (return))
33   (loop-for-count (?i 1 10000)
34     (bind ?val (read filein))
35     (if (eq ?val EOF) then (return))
36     (assert (numbers (value ?val)))) ;; Reading the numbers as facts
37   (close filein)
38   (printout t "Data read from " ?filename crlf)
39 )
40
41 ;; Insertion into sorted sequence function
42 (deffunction insert-ordered (?val ?sorted)
43   ;; Функція для вставки значення в відсортований ряд
44   (bind ?result (create$))
45   (bind ?inserted FALSE)
46   (foreach ?x ?sorted
47     (if (and (not ?inserted) (< ?val ?x)) then
48       (bind ?result (create$ ?result ?val))
49       (bind ?inserted TRUE))
50     (bind ?result (create$ ?result ?x)))
51   (if (not ?inserted) then
52     (bind ?result (create$ ?result ?val)))
53   ?result)
54
55 ;; Function to reflect numbers into symbols
56 (deffunction assign-symbols (?vals)
57   (bind ?count (length$ ?*alphabet*)) ;; Length of the alphabet
58   (bind ?min (nth$ 1 ?vals)) ;; Minimal value
59   (bind ?max (nth$ (length$ ?vals) ?vals)) ;; Maximal value
60   (bind ?step (/ (- ?max ?min) ?count)) ;; Interval width
61
62   ;; Creating the intervals
63   (bind ?intervals (create$))
64   (bind ?i 1)
65   (while (<= ?i ?count)
66     (bind ?start (+ ?min (* (- ?i 1) ?step)))
```

```

67      (bind ?end (+ ?start ?step))
68      (bind ?sym (nth$ ?i ?*alphabet*)) ;; Determining the number
69      (bind ?intervals (create$ ?intervals ?start ?end ?sym))
70      (bind ?i (+ ?i 1)))
71
72  ;; Binding numbers to symbols
73  (do-for-all-facts ((?n numbers)) TRUE
74    (bind ?v ?n:value)
75    (bind ?j 0)
76    (while (< ?j (* ?count 3))
77      (bind ?a (nth$ (+ ?j 1) ?intervals))
78      (bind ?b (nth$ (+ ?j 2) ?intervals))
79      (bind ?s (nth$ (+ ?j 3) ?intervals))
80      (if (or (and (>= ?v ?a) (< ?v ?b))
81          (and (= ?v ?max) (= ?b ?max))) then
82        (assert (symbol-mapped (value ?v) (symbol ?s))) ;; Reflecting the symbol
83        (bind ?j (* ?count 3))) ; вихід з циклу
84      (bind ?j (+ ?j 3))))
85
86  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
87  ;; Rules
88  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
89  ;; Number-sorting rule
90  (defrule sort-values
91    =>
92    (bind ?raw (create$))
93    (do-for-all-facts ((?n numbers)) TRUE
94      (bind ?raw (create$ ?raw ?n:value)))
95    (bind ?sorted (create$))
96    (foreach ?val ?raw
97      (bind ?sorted (insert-ordered ?val ?sorted))) ;; Using a sorting function
98    (assert (sorted-list (values ?sorted))) ;; Saving a sorted sequence
99    (assert (status (stage ready))) ;; Ready status
100  )
101
102  ;; Symbol-number mapping rule
103  (defrule map-values-to-symbols
104    ?sorted <- (sorted-list (values $?vals)) ;; Checking if a number sequence is sorted
105    (status (stage ready)) ;; Checking the ready status
106    =>
107    (assign-symbols ?vals) ;; Calling a reflection function
108  )
109
110  ;; Rule to build linguistic pairs
111  (defrule build-pairs
112    =>
113    (bind ?all (find-all-facts ((?s symbol-mapped)) TRUE)) ;; Finding all facts
114    (bind ?len (length$ ?all)) ;; Taking a number of facts
115    (loop-for-count (?i 1 (- ?len 1))
116      (bind ?from (fact-slot-value (nth$ ?i ?all) symbol)) ;; Taking a symbol from a
117      current fact
118      (bind ?to (fact-slot-value (nth$ (+ ?i 1) ?all) symbol)) ;; Taking a symbol
119      from the next fact
120      (assert (linguistic-pair (from ?from) (to ?to) (pair-id ?i)))) ;; Creating a
121      linguistic pair
122    )
123
124  ;; Rule to print the linguistic sequence
125  (defrule print-linguistic-sequence
126    =>
127    (printout t crlf "Linguistic sequence: ")
128    (do-for-all-facts ((?s symbol-mapped)) TRUE
129      (printout t ?s:symbol " ") ;; Print every symbol in a sequence
130    )
131    (printout t crlf)
132  )
133
134  ;; Rule to print the precedence matrix
135  (defrule print-precedence-matrix
136    =>
137    (printout t crlf "Precedence matrix:" crlf)

```

```
134     (printout t " ")
135     (foreach ?col ?*alphabet*
136       (printout t " " ?col)) ;; Printing row header
137     (printout t crlf)
138     (foreach ?row ?*alphabet*
139       (printout t ?row " ") ;; Printing column header
140       (foreach ?col ?*alphabet*
141         (bind ?count (length$ (find-all-facts ((?t linguistic-pair)
142           (and (eq ?t:from ?row) (eq ?t:to ?col)))))
143         (printout t " " ?count)) ;; Printing the amount of pairs for every symbol
143 combination
144       (printout t crlf))
145   )
```