

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

Лабораторна робота № 1  
з дисципліни «Мультипарадигмене програмування»

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

Київ - 2025

## Завдання

На процедурній мові 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

Fortran.

## 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. Читання числового ряду з файлу
  - Визначення кількості чисел у файлі ( $n$ ).
  - Динамічна алокація пам'яті масиву `numbers(n)`
  - Читання чисел у масив.
2. Сортвання чисел у порядку зростання
3. Розбиття чисел на інтервали на рівномірні інтервали
  - Визначення мінімального ( $\text{min\_number}$ ) та максимального ( $\text{max\_number}$ ) чисел.
  - Обчислення довжини інтервалу:  $\text{interval\_length} = \frac{\text{max\_number} - \text{min\_number}}{\text{alphabet\_size}}$
4. Генерування лінгвістичного ряду
  - Перетворення кожного числа на літеру за індексом інтервалу.
5. Побудова матриці передування
  - Створення квадратної матриці `transition_matrix(alphabet_size, alphabet_size)`, де кожна комірка містить кількість випадків слідування літери стовпчика за літерою рядка.
6. Виведення результатів
  - Виведення лінгвістичного ряду.
  - Виведення матриці передування.

```
rhinemann@rhinemann-PC ~/D/K/S/w/w/Lab_1 (main)> ./bin/Release/lab_1
Enter alphabet size
```

[illegible]

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

[illegible]

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

## Лістинг коду

main.f90

```
1  module procedures
2      implicit none
3
4      private
5      public quicksort, print_matrix
6
7  contains
8      ! Array quicksort subroutine
9      recursive subroutine quicksort(a)
10         real, intent(inout) :: a(:)
11         real x, t
12         integer :: first = 1, last
13         integer i, j
14
15         last = size(a, 1)
16         x = a( (first+last) / 2 )
17         i = first
18         j = last
19
20         do
21             do while (a(i) < x)
22                 i=i+1
23             end do
24             do while (x < a(j))
25                 j=j-1
26             end do
27             if (i >= j) exit
28             t = a(i); a(i) = a(j); a(j) = t
29             i=i+1
30             j=j-1
31         end do
32
33         if (first < i - 1) call quicksort(a(first : i - 1))
34         if (j + 1 < last) call quicksort(a(j + 1 : last))
35     end subroutine quicksort
36
37     ! Matrix Printing subroutine
38     subroutine print_matrix(matrix, alphabet)
39         integer, intent(in) :: matrix(:, :)
40         character(len=1), intent(in) :: alphabet(:)
41
42         integer :: i
43
44         print '(X, *(A8))', alphabet
45         do i = 1, size(matrix, 1)
46             print '(A, *(I8))', alphabet(i), matrix(i, :)
47         end do
48     end subroutine print_matrix
49 end module procedures
50
51 program lab_1
52     use procedures
53     implicit none
54
55     real, allocatable :: numbers(:), original_numbers(:)
56     character(len=1), allocatable :: alphabet(:), linguistic_sequence(:)
57     integer, allocatable :: precedence_matrix(:, :)
58
59     integer :: n, i, index, alphabet_size, io_status, row, col
60     real :: min_number, max_number, interval_length
61     character(len=8) :: filename = "data.txt"
62
63     ! Input alphabet size
64     print *, "Enter alphabet size"
65     do
66         read *, alphabet_size
67         if (alphabet_size <= 26) exit
```

```

68     print *, "Alphabet size too large, try again"
69 end do
70
71
72 ! Allocate alphabet and precedence matrix
73 allocate(alphabet(alphabet_size))
74 allocate(precedence_matrix(alphabet_size, alphabet_size))
75
76 ! Fill the alphabet with values
77 do i = 1, alphabet_size
78     alphabet(i) = achar(64 + i)
79 end do
80
81 ! Reading a number file and counting the amount of numbers
82 open(unit=10, file=filename, status="old", action="read", iostat=io_status)
83 if (io_status /= 0) then
84     print *, "Error opening file."
85     stop
86 end if
87
88 n = 0
89 do
90     read(10, *, iostat=io_status)
91     if (io_status /= 0) exit
92     n = n + 1
93 end do
94 close(10)
95
96 ! Allocating memory for arrays
97 allocate(numbers(n))
98 allocate(original_numbers(n))
99 allocate(linguistic_sequence(n))
100
101 ! Reading numbers from file to original_numbers
102 open(unit=10, file=filename, status="old", action="read", iostat=io_status)
103 if (io_status /= 0) then
104     print *, "Error opening file."
105     stop
106 end if
107
108 do i = 1, n
109     read(10, *) original_numbers(i)
110 end do
111 close(10)
112
113 ! Copying original_numbers to numbers for sorting
114 numbers = original_numbers
115 call quicksort(numbers)
116
117 ! Calculating step intervals
118 min_number = numbers(1)
119 max_number = numbers(n)
120 interval_length = (max_number - min_number) / alphabet_size
121
122 ! Converting numbers to a lexical string
123 do i = 1, n
124     index = ceiling((original_numbers(i) - min_number) / interval_length)
125     if (index < 1) index = 1
126     if (index > alphabet_size) index = alphabet_size
127     linguistic_sequence(i) = alphabet(index)
128 end do
129
130 ! Constructing a precedence matrix
131 do i = 1, n - 1
132     row = findloc(alphabet, linguistic_sequence(i), 1)
133     col = findloc(alphabet, linguistic_sequence(i + 1), 1)
134     precedence_matrix(row, col) = precedence_matrix(row, col) + 1
135 end do
136
137 ! Printing program results
138 print *, "Alphabet Size:", alphabet_size

```

```
139     print *, "Linguistic Sequence:", " ", linguistic_sequence
140     print *, "Transition Matrix:"
141     call print_matrix(precedence_matrix, alphabet)
142
143     ! Freeing allocated memory
144     deallocate(numbers, original_numbers, alphabet, linguistic_sequence, precedence_matrix)
145 end program lab_1
```