Міністерство освіти і науки України

Національний технічний університет України «Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №1

з дисципліни:

«Мультипарадигменне програмування»

Виконав:

студент групи ІК-21

Бераудо Лоренцо Раффаелович

Київ 2025

ЛАБОРАТОРНА РОБОТА №1

Завдання:

На імперативній мові програмування (Fortran) 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: Fortran (процедурне програмування, без використання ООП).

Варіант 1: Дискретний рівномірний розподіл (рівноймовірний)

Хід розв'язання задачі

Вхідні дані:

- Генерується чисельний ряд випадкових чисел у діапазоні [1, 1000]:

```
call random_seed()
do i = 1, n
    call random_number(step)
    num_list(i) = int(step * 999) + 1
end do
```

- Визначається потужність алфавіту (10 символів).
- Використовується список англійського алфавіту для відображення чисел у символи:

```
alphabet = (/'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J'/)
```

Сортування чисельного ряду

Вхідний чисельний ряд сортується у порядку зростання для визначення мінімального та максимального значення. Вони використовуються для побудови інтервалів:

```
sorted_list = num_list
call sort(sorted_list, n)
```

Розбиття на інтервали (Дискретний рівномірний розподіл)

Функція визначає межі інтервалів, розподіляючи значення рівномірно по заданій кількості класів:

```
step = real(max_val - min_val) / alphabet_power
do i = 0, alphabet_power
    segment_ends(i + 1) = min_val + int(i * step)
end do
segment_ends(alphabet_power + 1) = max_val
```

Присвоєння букв відповідно до інтервалів

Функція визначає, до якого інтервалу належить кожне число у числовому ряді, потім замінює числові значення на відповідні літери:

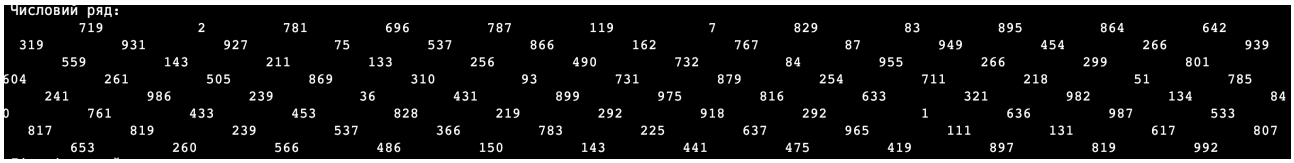
```
do i = 1, n
    classified_indices(i) = find_segment_index(num_list(i),
 segment_ends, alphabet_power)
end do
```

Побудова матриці передування

Функція створює матрицю переходів між символами:

```
transition_matrix = 0
do i = 1, n - 1
    transition_matrix(classified_indices(i),
classified_indices(i + 1)) = &
        transition_matrix(classified_indices(i),
classified_indices(i + 1)) + 1
end do
```

Результати розрахунку



Лінгвістичний ряд:

Матриця передування:

```
Матриця передування:
        0       0       0       0       1       1       1       3       1       3
        1       2       2       0       1       0       1       1       1       0
        3       1       1       1       1       3       1       1       0       3
        1       0       0       0       0       0       0       1       1       2
        0       1       1       0       3       0       0       1       1       2
        0       1       0       1       1       0       0       0       3       0
        0       0       2       2       0       0       0       0       1       2
        3       1       3       0       1       0       0       2       1       0
        1       1       3       1       0       0       4       1       1       1
        1       2       3       0       1       2       0       0       3       2
```

Висновок

Під час виконання лабораторної роботи було реалізовано алгоритм перетворення чисельного ряду у лінгвістичний ланцюжок відповідно до дискретного рівномірного розподілу. Числовий ряд був рівномірно розбитий на інтервали, кожному інтервалу присвоєно символ. Було побудовано матрицю передування, що демонструє ймовірності переходів між символами.

В ході виконання роботи здобуто практичні навички роботи з масивами, функціями, сортуванням, обробкою чисел та реалізацією алгоритмів на імперативній мові програмування Fortran.

Код програми

```
program uniform_distribution
    implicit none
    integer, parameter :: n = 100, alphabet_power = 10
    integer :: i, min_val, max_val, index
    real :: step
    integer :: num_list(n), sorted_list(n),
segment_ends(alphabet_power + 1), classified_indices(n)
    character(len=1) :: alphabet(10)
    integer :: transition_matrix(alphabet_power, alphabet_power)

    ! Ініціалізація алфавіту
    alphabet = (/'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I',
'J'/)

    ! Генерація випадкових чисел
    call random_seed()
    do i = 1, n
        call random_number(step)
        num_list(i) = int(step * 999) + 1
    end do

    ! Копіювання і сортування списку
    sorted_list = num_list
    call sort(sorted_list, n)

    ! Визначення меж інтервалів
    min_val = sorted_list(1)
    max_val = sorted_list(n)
    step = real(max_val - min_val) / alphabet_power
    do i = 0, alphabet_power
        segment_ends(i + 1) = min_val + int(i * step)
    end do
    segment_ends(alphabet_power + 1) = max_val

    ! Класифікація чисел за інтервалами
    do i = 1, n
        classified_indices(i) = find_segment_index(num_list(i),
segment_ends, alphabet_power)
    end do

    ! Обчислення матриці передування
    transition_matrix = 0
    do i = 1, n - 1
        transition_matrix(classified_indices(i),
classified_indices(i + 1)) = &
            transition_matrix(classified_indices(i),
classified_indices(i + 1)) + 1
    end do

    ! Вивід результатів
    print *, "Числовий ряд:"
```

```
    print *, num_list

    print *, "Лінгвістичний ряд:"
    do i = 1, n
        write(*, '(A1)', advance="no")
alphabet(classified_indices(i))
    end do
    print *

    print *, "Матриця передування:"
    do i = 1, alphabet_power
        print *, transition_matrix(i, :)
    end do

contains
    ! Функція сортування масиву (простий алгоритм бульбашкового
сортування)
    subroutine sort(arr, size)
        implicit none
        integer, intent(inout) :: arr(size)
        integer, intent(in) :: size
        integer :: i, j, temp
        do i = 1, size - 1
            do j = 1, size - i
                if (arr(j) > arr(j + 1)) then
                    temp = arr(j)
                    arr(j) = arr(j + 1)
                    arr(j + 1) = temp
                end if
            end do
        end do
    end subroutine sort

    ! Функція визначення індексу інтервалу
    integer function find_segment_index(num, segments, count)
        implicit none
        integer, intent(in) :: num, segments(count + 1), count
        integer :: i
        do i = 1, count
            if (num < segments(i + 1)) then
                find_segment_index = i
                return
            end if
        end do
        find_segment_index = count
    end function find_segment_index

end program uniform_distribution
```