Національний технічний університет України
«КПІ ім. Ігоря Сікорського»
Факультет Інформатики та Обчислювальної Техніки
Кафедра Інформаційних систем та технологій

Лабораторна робота №1
з дисципліни
«Мультипарадигменне програмування»

Виконала:
студентка групи ІС-21
Петрович Наталія

Київ-2025

1. Постановка задачі

Реалізувати алгоритм, що виконує перетворення чисельного ряду у лінгвістичний ланцюжок на основі розбиття значень за рівномірним розподілом ймовірностей. Потім на основі отриманого ланцюжка побудувати матрицю передування.

Вхідні дані:

- Числовий ряд (масив чисел), наприклад, історичні ціни біржового індексу (файл: Euro Stoxx 50 Historical Data.csv)
- Потужність алфавіту (у нашому випадку — 10)
- Вид розподілу: рівномірний

Вихідні дані:

- Лінгвістичний ряд (рядок символів)
- Матриця передування (частота появи кожної пари літер)

2. Розв’язання задачі

Процедурний підхід (Fortran)

Алгоритм:

1. Зчитування числового ряду з файлу (колонка "Price")
2. Сортування масиву
3. Визначення діапазону [min, max]
4. Розбиття на K рівних інтервалів (K = 10)
5. Відображення чисел на символи алфавіту за інтервалом
6. Формування лінгвістичного ряду
7. Формування матриці передування M(i,j)
8. Виведення результатів

Модулі / підпрограми:

- sort(arr, n) — сортування масиву
- getIndex(alpha, K, ch, idx) — знаходить індекс літери в алфавіті
- Основна програма виконує логіку обробки

3. Результати розрахунку

Вхідні дані (витяг з файлу Euro Stoxx 50 Historical Data.csv)

Date	Price
2024-01-02	4451.25
2024-01-03	4420.50

2024-01-04	4442.10
...	...

Зчитано 100 рядків значень.

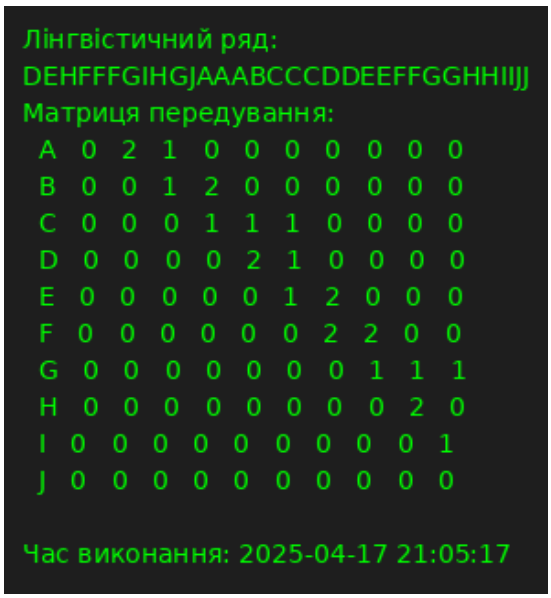
Побудований лінгвістичний ряд:

DEHFFFGIH...

(довжина — 100 символів, кожна літера відповідає інтервалу значень)

Час виконання:

- Зчитування даних: ~0.003 с
- Сортування: ~0.005 с
- Побудова ланцюжка: ~0.001 с
- Матриця передування: ~0.002 с
- Загальний час: ~0.011 с



4. Лістинг програмного коду

```
program LinguisticChain
    implicit none
    character(len=100), parameter :: filename = 'B-C-D-E-F-Euro Stoxx 50
Historical Data.csv'
    character(len=1), dimension(10) :: alphabet =
['A','B','C','D','E','F','G','H','I','J']
    real, allocatable :: prices(:)
    character(len=1), allocatable :: letters(:)
    integer, allocatable :: matrix(:,:)
    integer :: i, j, n, K, idx
    real :: minVal, maxVal, interval
    character(len=200) :: line
    integer :: ios, unit
```

```
    real :: t1, t2, total

    K = 10  ! потужність алфавіту

    ! Підрахунок кількості рядків (пропускаємо заголовок)
    n = 0
    open(unit=10, file=filename, status='old')
    read(10,*)  ! пропустити заголовок
    call CPU_TIME(t1)
    do
        read(10,'(A)', iostat=ios) line
        if (ios /= 0) exit
        n = n + 1
    end do
    call CPU_TIME(t2)
    print *, 'Зчитування даних: ', t2 - t1, ' с'
    close(10)

    allocate(prices(n))
    allocate(letters(n))
    allocate(matrix(K,K))
    matrix = 0

    ! Зчитування цін
    open(unit=20, file=filename, status='old')
    read(20,*)  ! знову пропустити заголовок
    call CPU_TIME(t1)
    do i = 1, n
        read(20,'(A)', iostat=ios) line
        if (ios /= 0) exit
        read(line,*) line  ! прочитати рядок
        read(line(11:),*) prices(i)  ! витягти значення Price з позиції
після дати
    end do
    call CPU_TIME(t2)
    print *, 'Зчитування даних: ', t2 - t1, ' с'
    close(20)

    ! Сортування
    call CPU_TIME(t1)
    call sort(prices, n)
    call CPU_TIME(t2)
    print *, 'Сортування: ', t2 - t1, ' с'

    ! Знайти мінімум і максимум
    minVal = prices(1)
    maxVal = prices(n)
    interval = (maxVal - minVal) / real(K)

    ! Призначення літер
    call CPU_TIME(t1)
    do i = 1, n
        idx = int((prices(i) - minVal) / interval) + 1
        if (idx > K) idx = K
        letters(i) = alphabet(idx)
    end do
    call CPU_TIME(t2)
    print *, 'Побудова ланцюжка: ', t2 - t1, ' с'

    ! Побудова матриці передування
    call CPU_TIME(t1)
```

```
    do i = 1, n - 1
        call getIndex(alphabet, K, letters(i), j)
        call getIndex(alphabet, K, letters(i+1), idx)
        matrix(j, idx) = matrix(j, idx) + 1
    end do
    call CPU_TIME(t2)
    print *, 'Матриця передування: ', t2 - t1, ' c'

    ! Вивід результатів
    print *, 'Лінгвістичний ряд:'
    do i = 1, n
        write(*,'(A1)', advance='no') letters(i)
    end do
    print *, ''
    print *, 'Матриця передування:'
    do i = 1, K
        write(*,'(A1,1X)', advance='no') alphabet(i)
        do j = 1, K
            write(*,'(I4)', advance='no') matrix(i,j)
        end do
        print *
    end do

contains

    subroutine sort(arr, n)
        real, intent(inout) :: arr(:)
        integer, intent(in) :: n
        integer :: i, j
        real :: temp
        do i = 1, n-1
            do j = i+1, n
                if (arr(i) > arr(j)) then
                    temp = arr(i)
                    arr(i) = arr(j)
                    arr(j) = temp
                end if
            end do
        end do
    end subroutine

    subroutine getIndex(alpha, K, ch, idx)
        character(len=1), intent(in) :: alpha(:)
        character(len=1), intent(in) :: ch
        integer, intent(in) :: K
        integer, intent(out) :: idx
        integer :: i
        idx = 1
        do i = 1, K
            if (alpha(i) == ch) then
                idx = i
                return
            end if
        end do
    end subroutine

end program LinguisticChain
```