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

ЗВІТ

з лабораторної роботи №1

з дисципліни «Мультипарадигменне програмування»

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

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.

Варіант: 20

Розподіл ймовірностей: Розбиття на рівні інтервали

Хід розв'язання задачі:

Програма реалізує перетворення числового ряду у **лінгвістичний ланцюжок** з наступним побудуванням **матриці передування**. У цій реалізації використовується **мовний стандарт Fortran 90** із використанням **динамічних масивів** для підвищення гнучкості.

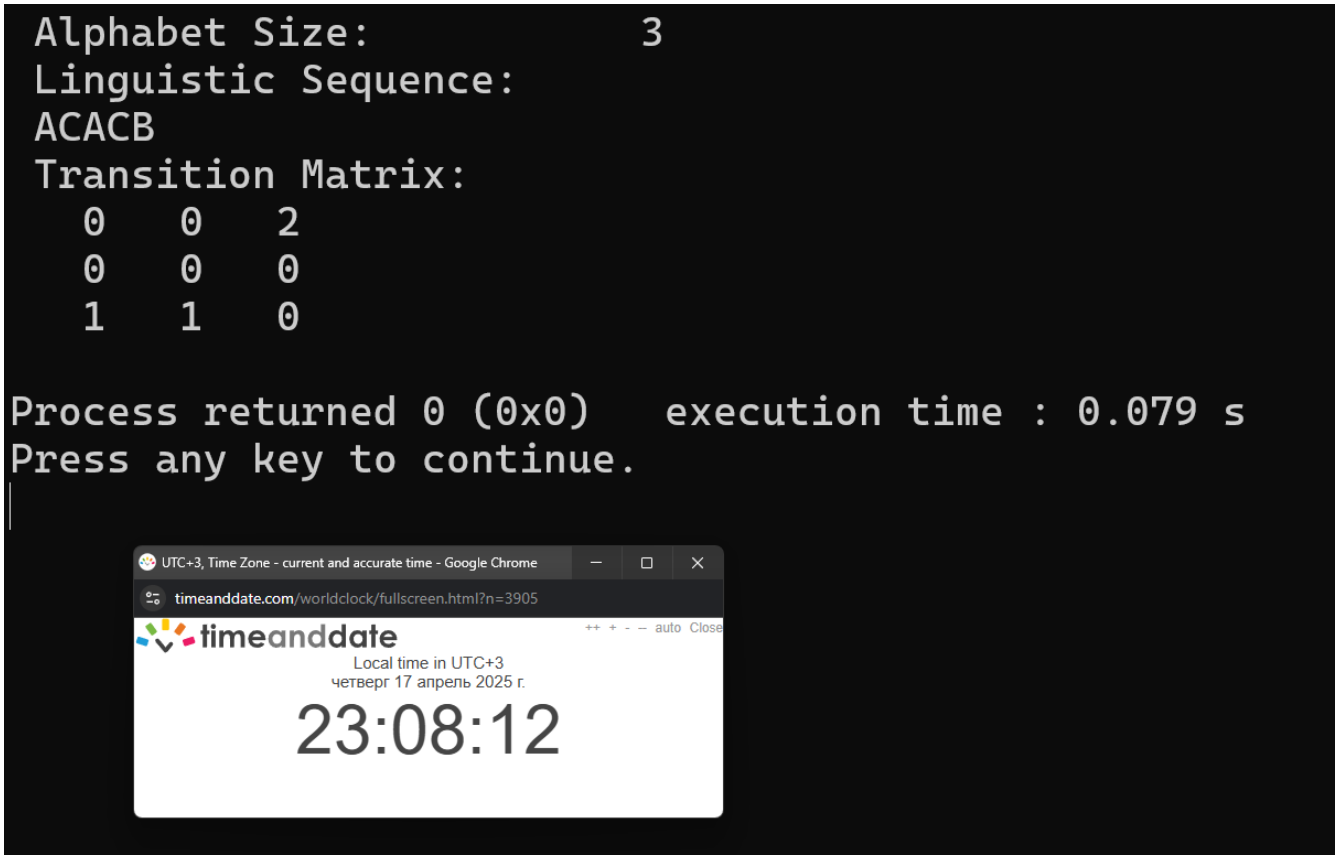
Алгоритм можна поділити на 6 основних етапів:

1. **Читання числового ряду з файлу**
    - Визначаємо кількість чисел у файлі (n).
    - Динамічно виділяємо пам'ять під масив numbers(n).
    - Читаємо числа у масив.
2. **Сортування чисел у порядку зростання**
3. **Розбиття чисел на інтервали за рівномірним розподілом**
    - Визначаємо мінімальне (min_val) та максимальне (max_val) значення.
    - Обчислюємо ширину інтервалу: $\text{step} = \frac{\max_{val} - \min_{val}}{alphabet\_size}$
    - Кожне число переводиться у символ алфавіту залежно від його інтервалу.

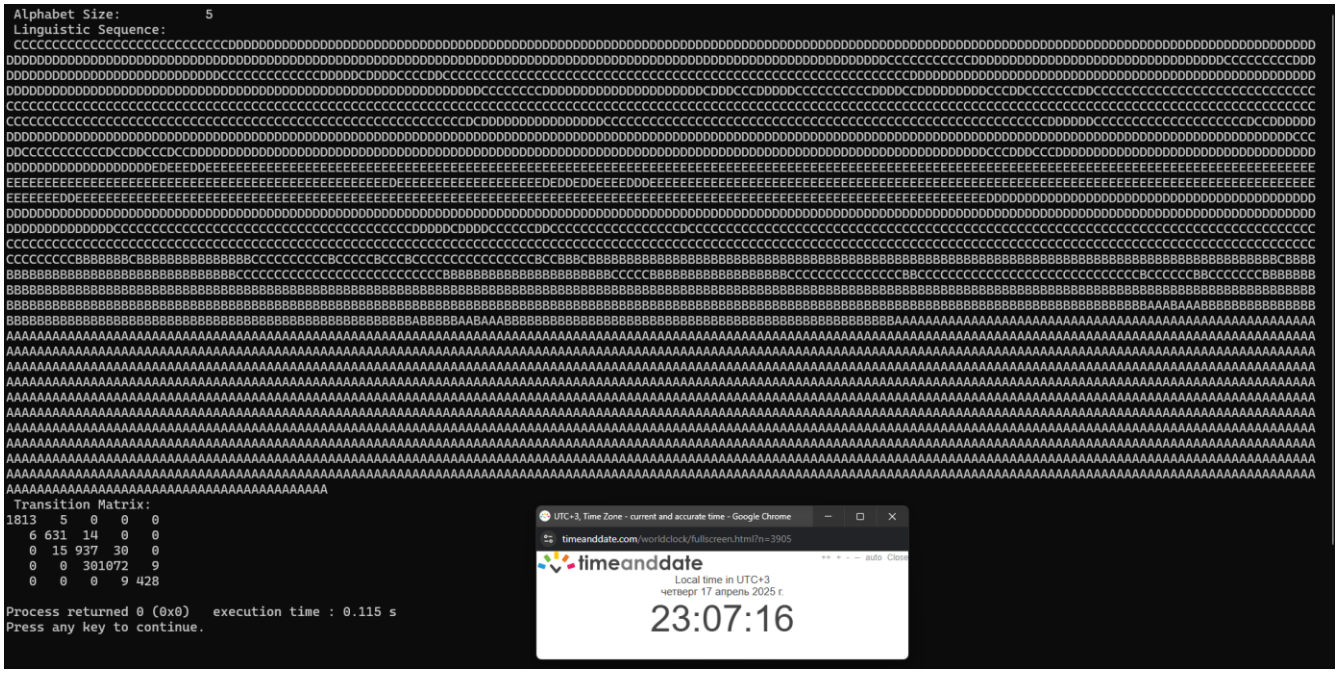
4. **Генерація лінгвістичного ряду**
    - Кожному числу присвоюється буква *(A, B, C, ...)*.
    - Літера визначається за індексом інтервалу.
5. **Побудова матриці передування**
    - Створюється квадратна матриця *transition_matrix(alphabet_size, alphabet_size)*, де кожна комірка містить кількість випадків, коли одна літера слідує за іншою.
6. **Виведення результатів**
    - Виводиться **лінгвістичний ряд**.
    - Виводиться **матриця передування**.

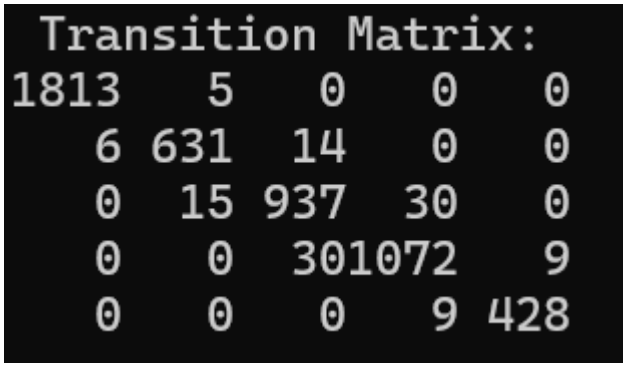
Результати виконання

Перший числовий ряд (5 значень – 3 символи): 3.2, 7.8, 1.5, 9.0, 4.6

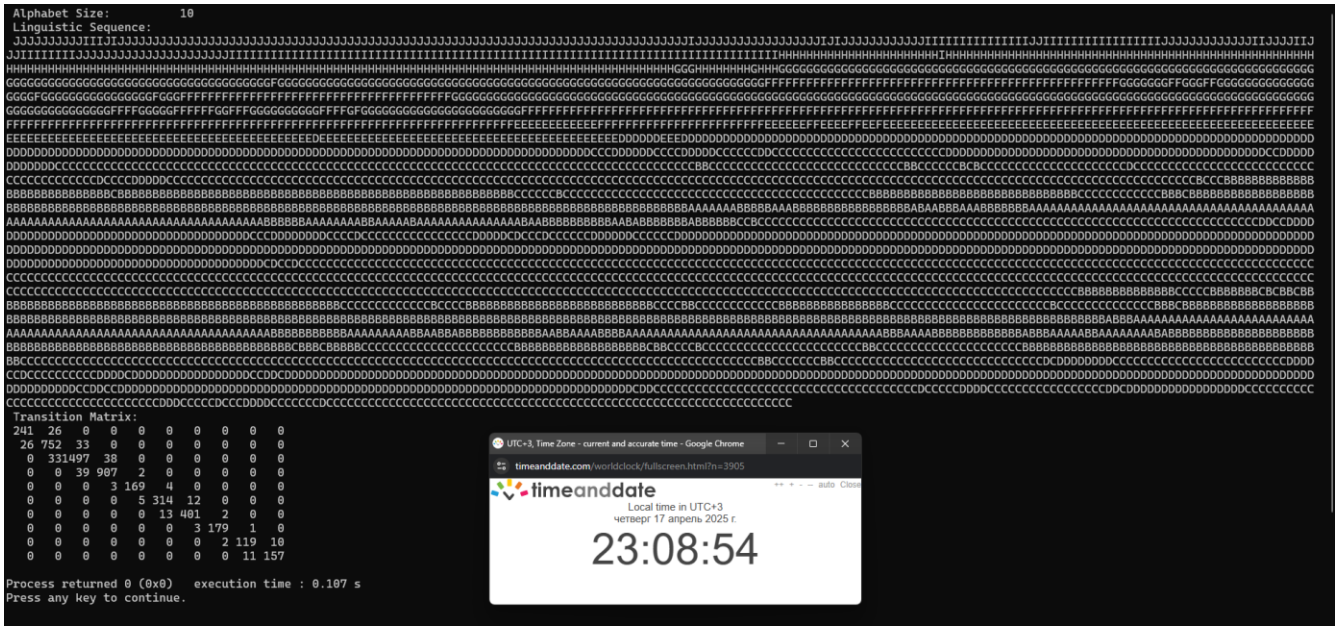


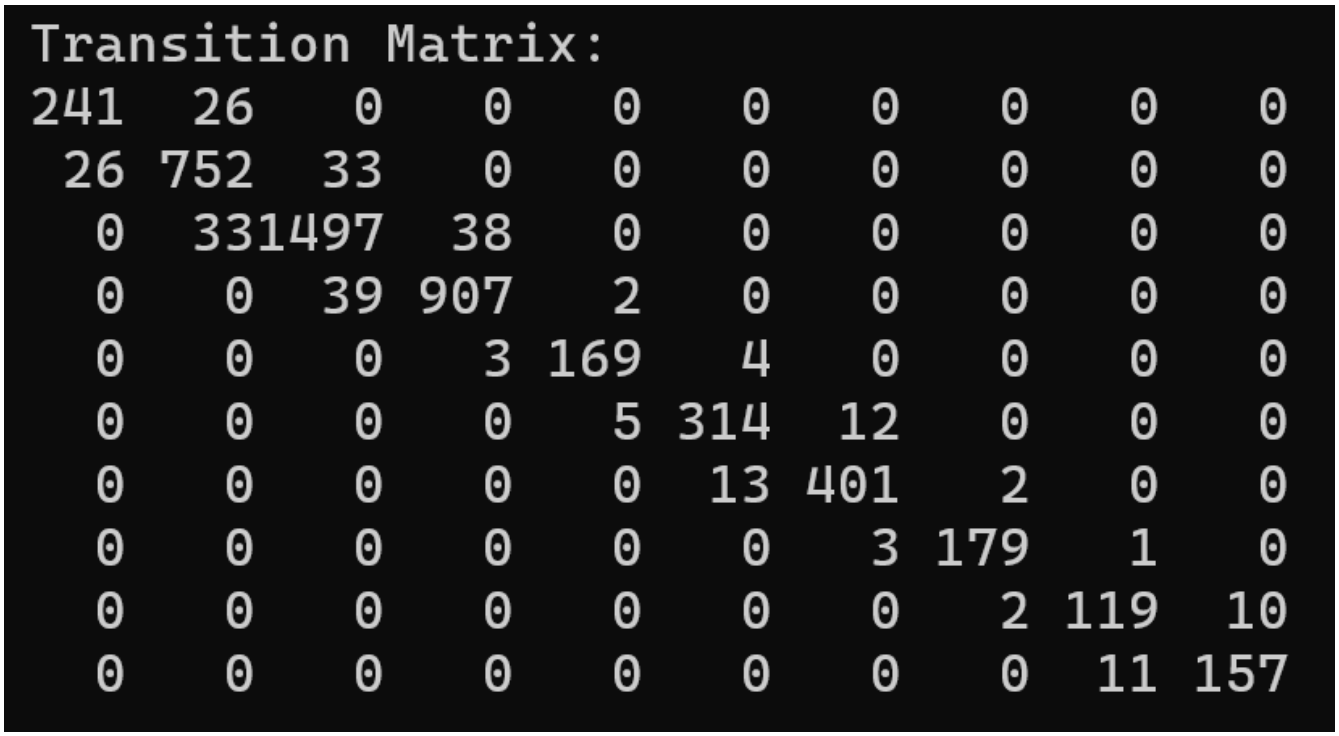
Другий числовий ряд (5000 значень - 5 символів): B-C-D-E-Gold Futures Historical Data (Price)





Третій числовий ряд (5000 значень – 10 символів): B-C-D-E-S&P 500 Historical Data (Price)





Лістінг програмного тексту

```
program lab1
  implicit none

  real, allocatable :: numbers(:), original_numbers(:)
  integer :: n, i, j, index, alphabet_size, io_status
  character(len=1), allocatable :: alphabet(:)
  integer, allocatable :: transition_matrix(:,:)
  real :: min_val, max_val, step
  character(len=1), allocatable :: sequence(:)
  integer :: row, col

  ! === 1. Задаємо потужність алфавіту ===
  alphabet_size = 10

  ! === 2. Створюємо алфавіт і матрицю ===
  allocate(alphabet(alphabet_size))
  allocate(transition_matrix(alphabet_size, alphabet_size))
  transition_matrix = 0
  do i = 1, alphabet_size
    alphabet(i) = achar(64 + i)  ! 'A', 'B', ...
  end do

  ! === 3. Підрахунок кількості чисел у файлі ===
  open(unit=10, file="3.txt", status="old", action="read")
  n = 0
  do
    read(10, *, iostat=io_status)
    if (io_status /= 0) exit
    n = n + 1
  end do
  close(10)

  ! === 4. Виділення пам'яті ===
  allocate(numbers(n))
  allocate(original_numbers(n))
  allocate(sequence(n))

  ! === 5. Зчитування чисел у original_numbers ===
  open(unit=10, file="3.txt", status="old", action="read")
  do i = 1, n
    read(10, *) original_numbers(i)
  end do
  close(10)
```

```
  ! === 6. Копія для сортування — numbers ===
  numbers = original_numbers
  call sort(numbers, n)

  ! === 7. Побудова інтервалів ===
  min_val = numbers(1)
  max_val = numbers(n)
  step = (max_val - min_val) / real(alphabet_size)

  ! === 8. Перетворення чисел у літери (по оригінальному порядку) ===
  do i = 1, n
    index = ceiling((original_numbers(i) - min_val) / step)
    if (index < 1) index = 1
    if (index > alphabet_size) index = alphabet_size
    sequence(i) = alphabet(index)
  end do

  ! === 9. Побудова матриці передування ===
  do i = 1, n - 1
    row = index_in_alphabet(sequence(i), alphabet, alphabet_size)
    col = index_in_alphabet(sequence(i + 1), alphabet, alphabet_size)
    transition_matrix(row, col) = transition_matrix(row, col) + 1
  end do

  ! === 10. Виведення результатів ===
  print *, "Alphabet Size:", alphabet_size
  print *, "Linguistic Sequence:"
  print *, sequence(1:n)
  print *, "Transition Matrix:"
  do i = 1, alphabet_size
    write(*, '(100I4)') (transition_matrix(i, j), j = 1,
alphabet_size)
  end do

  ! === 11. Звільнення пам’яті ===
  deallocate(numbers, original_numbers, sequence, alphabet,
transition_matrix)

contains

  ! === Функція сортування масиву ===
  subroutine sort(arr, size)
    implicit none
    integer, intent(in) :: size
```

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

  ! === Функція знаходження індексу символу у алфавіті ===
  function index_in_alphabet(c, alph, size) result(indx)
    implicit none
    character(len=1), intent(in) :: c
    character(len=1), intent(in) :: alph(size)
    integer, intent(in) :: size
    integer :: indx, i
    indx = 0
    do i = 1, size
      if (c == alph(i)) then
        indx = i
        exit
      end if
    end do
  end function index_in_alphabet

end program lab1
```