Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Програмування. Частина 2.

Лабораторна робота №2

«Масиви в мові програмування Java (Python)»

Виконав:
студент гр. ІО-41
Давидчук А. М.
Залікова книжка № 4106

Перевірив
Коренко Д.В.

Київ-2025

Тема: «Масиви в мові програмування Java (Python)».

Мета: Ознайомлення з масивами та використання основних методів їх обробки в мові програмування Java (Python). Здобуття навичок у використанні масивів в мові програмування Java (Python).

Через те, що є можливість використовувати мову Python, для виконання лабораторних робіт з ООП, то я надалі буду його використовувати як основний засіб.

Моя залікова книжка 4106, значить:

$C_5 = 4106 \bmod 5 = 1$
$C_7 = 4106 \bmod 7 = 4$
$C_{11} = 4106 \bmod 11 = 3$

Згідно з таблицями варіантів, мій варіант:

C_5	Дія з матрицею
0	$C = a \cdot B,\ a - const$
1	$C = B^T$
2	$C = A + B$
3	$C = A \oplus B$
4	$C = A \times B$

C_7	Тип елементів матриці
0	double
1	byte
2	short
3	int
4	long
5	char
6	float

C_{11}	Дія з матрицею C
0	Обчислити суму найменших елементів кожного стовпця матриці
1	Обчислити суму найменших елементів кожного рядка матриці
2	Обчислити суму найбільших елементів кожного стовпця матриці
3	Обчислити суму найбільших елементів кожного рядка матриці
4	Обчислити суму найбільших елементів в рядках матриці з парними номерами та найменших елементів в рядках матриці з непарними номерами
5	Обчислити суму найбільших елементів в рядках матриці з непарними номерами та найменших елементів в рядках матриці з парними номерами

Код (Python):

```
from sys import exit
# sys.exit для завершення програми

class long:

    # Клас long -- шаблон цілочисельного 64-бітного числа

    def __init__(self, value: int):
        """
        Ініціалізує екземпляр класу long.
        Перевіряє, чи є введене значення цілим числом. Якщо ні, програма завершується.
        Обчислює і зберігає значення в межах допустимого діапазону 64-бітного числа.
        """
        try:

            # Перевірка, чи є value цілим числом
            if not float(value).is_integer(): print("Error: value must be integer!"); exit()

        except:

            # Якщо виникла помилка при перетворенні в float
            print("Error: value must be integer!"); exit()

        # Мінімальне та максимальне значення для 64-бітного числа
        RANGE = 1 << 64
        MIN_LONG = -(RANGE // 2)

        # Обертання значення в межах допустимого діапазону
        value = (value - MIN_LONG) % RANGE + MIN_LONG

        self.value = value

    def __gt__(self, other):

        #Операція порівняння: перевіряє, чи більше поточне значення за інше.
        return self.value > other.value

    def __radd__(self, other):

        #Операція додавання: реалізує додавання до числа `long` з іншими типами
(наприклад, числами).
        return self.value + other

    def __str__(self) -> str:
```

```
        # Метод для переведення об'єкта в рядок.
        return str(self.value)

class Matrix_Calc:
    # Клас Matrix_Calc для обчислення та маніпуляцій з матрицями

    def __init__(self, matrix: list[list], rows: int, cols: int):

        #Ініціалізація матриці та її перетворення в матрицю з елементами типу `long`.
        self.matrix = matrix

        # Перетворення всіх елементів матриці на об'єкти класу long
        for i in range(rows):
            for j in range(cols):
                self.matrix[i][j] = long(matrix[i][j])

        self.rows = rows
        self.cols = cols
        self.C_matrix = None

    def calculate_C_matrix(self):

        # Обчислює C матрицю шляхом транспонування початкової матриці

        self.C_matrix = [[0] * self.rows for _ in range(self.cols)]

        # Транспонування матриці: елемент на позиції [i][j] переміщується на [j][i]
        for i in range(self.rows):
            for j in range(self.cols):
                self.C_matrix[j][i] = self.matrix[i][j]

    def sum_the_matrix(self) -> int:

        # Підраховує суму максимальних елементів кожного рядка в транспонованій матриці.

        sum = 0
        for row in self.C_matrix: sum = sum + max(row)
        return sum

    def print_C_matrix(self):

        #Виводить транспоновану матрицю в зручному для читання форматі.
        for row in self.C_matrix: print(" ".join(map(str, row)))

# Приклад використання
matrix = [
```

```
    [-1, -2, 0],
    [-1, 0, -3],
    [0, -1, -3]
]

result = Matrix_Calc(matrix, 3, 3) # Створення об'єкта для матриці
result.calculate_C_matrix() # Обчислення матриці C: транспонування матриці matrix
result.print_C_matrix() # Виведення матриці C
print(f"\nsum: {result.sum_the_matrix()}") # Підрахунок та виведення суми максимальних
елементів кожного рядка
```

Висновок

У цій лабораторній роботі було розроблено клас `long`, який реалізує 64-бітове ціле число з перевіркою на вхідні значення та підтримкою операцій порівняння та додавання. Також створено клас `Matrix_Calc`, який здійснює транспонування матриці та обчислює суму максимальних елементів кожного рядка транспонованої матриці. Це дозволяє працювати з матрицями та цілими числами у межах 64-бітового діапазону, а також зручно здійснювати математичні операції.