

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА  
ШЕВЧЕНКА  
Кафедра інформаційних та інтелектуальних систем

Лабораторна робота №1  
з дисципліни  
«Методи та системи паралельного  
програмування»

з теми «Паралельне програмування за допомогою стандартних засобів  
сучасних операційних систем»

Виконала студентка  
групи КН-31  
Шпирук Є. С.

Київ – 2020

**Мета роботи:** освоїти реалізацію паралельних обчислень за допомогою стандартних засобів багатопроцесорності та багатопоточності сучасних ОС та виконати порівняльне оцінювання часу виконання програми з послідовним обчисленням.

### **Завдання лабораторної роботи**

**Завдання №1.** На базі однієї із операційних систем ПК (за вибором студента) на одній із мов програмування (C / C++ / Java / Python / Pascal) створити програмний застосунок, який виконує операцію множення двох матриць, елементи яких задаються випадковим чином. Створити послідовний алгоритм розрахунку множення двох матриць. Відлагодити та запустити застосунок, який виводить на екран монітора час затрачений на виконання обчислень.

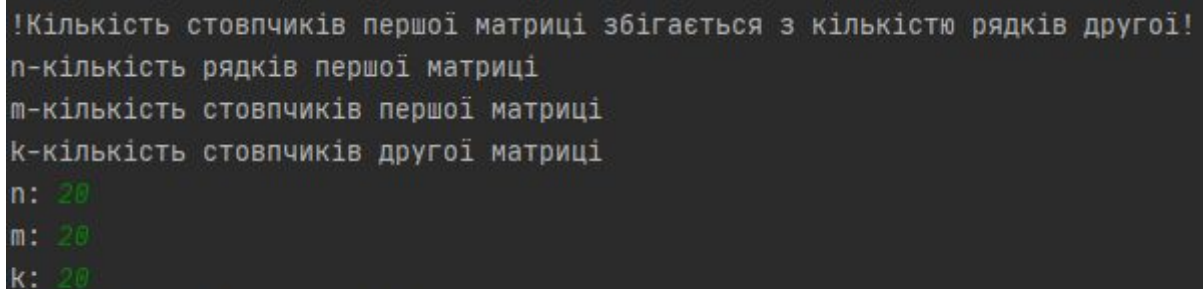
**Вхідні дані:**  $n$ ,  $m$ ,  $k$  – розмірності матриць  $n \times m$  та  $m \times k$ , уведені із клавіатури ПК.

**Вихідні дані:** Згенеровані вхідні та результуюча матриці, записані у файл для використання в завданні №2.

**Обрана мова програмування:** Python.

#### **Приклад роботи коду:**

Введемо наступні вхідні дані:  $n=20$ ,  $m=20$ ,  $k=20$ .



```
!Кількість стовпчиків першої матриці збігається з кількістю рядків другої!  
n-кількість рядків першої матриці  
m-кількість стовпчиків першої матриці  
k-кількість стовпчиків другої матриці  
n: 20  
m: 20  
k: 20
```

Рисунок 1. Вхідні дані для завдання №1

output1 – Блокнот

Файл	Правка	Формат	Вид	Справка															
44	29	3	15	87	60	39	27	17	15	76	90	91	55	23	33	25	60	56	92
46	8	25	29	91	10	29	68	89	21	67	74	37	45	6	8	54	43	62	29
76	100	76	35	49	78	29	34	29	91	15	83	47	30	30	90	1	33	31	69
11	8	32	1	94	18	38	1	61	15	73	47	93	31	98	51	89	35	25	10
12	44	12	92	38	94	1	95	36	8	81	64	89	33	62	72	38	51	57	32
51	71	99	51	78	87	28	30	29	89	10	73	2	88	29	10	31	58	55	31
47	23	77	61	61	46	71	22	3	60	21	69	67	98	87	99	47	96	55	28
8	9	96	100	42	44	67	76	51	86	4	41	56	35	60	6	57	97	63	1
14	51	17	96	96	19	26	2	18	12	84	21	56	30	5	8	89	7	26	32
48	61	31	7	57	47	37	11	72	96	47	60	75	37	18	80	29	16	93	79
96	100	78	75	51	1	6	48	29	14	20	75	81	39	50	39	66	58	33	46
84	58	11	21	100	97	40	36	23	77	64	52	75	100	32	76	71	21	14	20
61	49	25	46	41	79	73	85	10	26	84	23	59	95	78	28	45	37	5	16
85	45	95	86	43	48	72	53	49	25	41	18	7	72	38	87	41	23	22	29
89	48	68	81	84	58	61	29	73	14	94	58	22	31	98	39	30	17	24	49
1	73	55	67	60	68	37	63	63	37	70	80	3	83	76	46	28	27	58	8
57	7	35	85	66	32	63	48	57	77	72	25	49	82	71	14	31	24	67	94
66	28	39	72	59	66	49	49	22	21	22	29	7	56	2	91	29	41	82	56
76	47	5	90	99	50	32	9	76	21	98	44	48	17	41	90	92	15	1	37
93	76	82	15	7	61	87	87	87	25	90	63	75	81	82	57	89	19	19	26

Рисунок 2. Перша матриця

output2 – Блокнот

Файл	Правка	Формат	Вид	Справка															
64	13	8	77	13	34	27	90	68	28	43	2	95	62	82	53	73	79	22	12
11	57	27	60	85	25	64	58	97	57	42	89	36	20	67	39	46	27	94	46
20	58	54	16	49	45	31	44	27	56	75	65	26	85	83	4	54	22	79	19
28	75	82	46	88	77	40	13	64	63	78	44	82	26	66	2	96	75	59	9
16	88	9	36	87	50	48	16	69	29	25	49	50	13	90	49	96	6	65	51
29	62	68	62	76	35	72	95	40	98	62	85	20	26	79	31	73	40	14	46
83	37	57	57	55	46	64	24	20	91	55	20	13	1	77	14	92	12	63	8
37	30	56	84	76	2	64	49	60	37	49	100	7	61	33	63	36	78	4	98
6	70	88	48	75	56	18	91	38	37	79	52	10	14	64	37	52	93	9	41
93	93	60	80	4	83	27	49	61	45	22	34	21	85	93	5	94	17	64	33
92	83	93	91	77	66	78	21	25	81	78	88	86	23	58	43	64	21	51	67
52	25	45	36	30	78	64	88	21	58	31	59	2	94	97	49	13	28	23	43
46	99	85	90	41	90	99	86	21	62	27	2	71	11	38	75	93	1	33	69
70	81	32	84	52	83	84	62	80	68	84	89	69	40	54	43	77	57	23	22
34	38	35	11	55	50	54	89	40	33	31	50	30	31	24	70	38	9	70	31
62	51	99	2	66	92	84	24	80	76	55	15	4	44	14	84	20	21	20	27
78	37	33	63	27	80	58	77	16	57	60	40	6	78	5	85	73	22	86	2
32	100	29	45	28	12	15	48	22	43	45	1	33	82	51	25	60	80	37	97
13	21	43	44	49	3	36	42	9	50	34	31	36	6	88	69	73	74	30	21
16	12	79	85	57	21	64	96	37	2	98	97	10	11	7	87	58	57	23	46

Рисунок 3. Друга матриця



41086	53522	50447	57069	51094	47803	56753	57182	38499	49267	48984	48201	36123	33635	55074	49322	60849	35762	35639	42734
35922	47906	43498	49452	46701	42145	43420	47870	34364	41947	44816	43435	30739	34213	52681	40540	54318	38979	32466	36526
43011	56404	55415	54849	54326	52940	56107	62031	52413	54396	52286	52703	33516	45474	65002	45436	61942	39104	44105	39979
39087	51734	44771	42384	45216	50676	48964	49414	31745	45007	40895	36683	29680	31220	43867	45045	53512	23450	40245	33121
41565	59505	61279	56789	61792	52544	61254	58360	44799	57750	53956	54925	37687	38062	54703	51054	61507	42685	38521	47148
41168	58845	46364	54014	52582	49395	49755	57922	48284	53698	52497	55044	35050	47246	69189	37677	66009	41685	46439	36809
53583	67484	58624	56817	57649	63508	63305	62781	50989	63674	58151	49552	40813	50904	66291	51579	72971	41944	50996	42060
42279	61902	53057	52368	51750	50278	47601	53754	39618	54171	51080	45286	32941	46728	62546	37118	68392	43118	46864	39892
32206	45291	38563	42751	44028	42229	41578	34781	31895	39636	39804	37539	32959	23244	40992	32971	52821	24336	38435	25310
43975	56182	57650	57316	52109	53623	55578	60397	44937	52291	51191	48425	32960	36957	61293	49978	64471	38494	40838	38230
39982	54730	48671	54718	53000	51618	53747	61037	47996	49339	51889	48036	40392	45586	57748	48633	61939	41971	47620	39141
54740	65246	54635	64356	57320	62720	64776	63311	54381	61463	54342	54053	42555	43859	64273	51886	72309	37112	45823	41304
48551	56498	50476	59144	55194	50431	58826	55645	45988	56454	52508	53406	40098	37221	54214	44219	63715	36830	41823	40254
44319	53285	53334	50853	56347	51830	52459	51765	49331	55130	57448	49914	37251	40537	57887	41027	62621	42296	43431	31898
45330	57995	57278	55962	63214	55341	56872	61295	48865	56601	59502	56568	43065	39273	65232	47286	67730	42420	48668	38584
41910	58109	54076	51811	61148	52257	55839	54994	47666	57405	54619	59800	34147	40091	62694	42966	60343	40829	44541	39502
48058	59847	58697	63688	57947	54929	56704	60143	46756	53083	59651	55774	42724	37347	62094	47934	73759	45409	44758	39144
37171	46240	47959	47183	50807	42528	48036	46509	42935	48690	49854	43851	31881	33800	53097	41623	57451	41615	34024	31591
46370	58126	56594	54005	58764	59384	56105	54763	47564	54358	54687	47471	40192	36082	54684	49008	64778	37211	45013	35482
59227	65022	66696	70736	66001	65137	70914	76612	54027	69166	67807	64923	43211	51067	68015	58734	73867	47803	52856	46475

Рисунок 4. Результирующая матрица

Час: 0.0024595260620117188 секунд

Рисунок 5. Час витрачений на виконання обчислень

**Код:**

```
import random
import time
def arr(r, c):
    tmp = []
    res = []
    for i in range(0, r):
        for j in range(0, c):
            tmp.append(random.randint(1, 100))
        res.append(tmp)
        tmp = []
    return res
def multiply(arr1, arr2):
    new_el = 0
    tmp = []
    mult = []
    arr1r = len(arr1)
    arr1c = len(arr1[0])
    arr2c = len(arr2[0])
    for z in range(0, arr1r):
        for j in range(0, arr2c):
            for i in range(0, arr1c):
                new_el = new_el + arr1[z][i] * arr2[i][j]
```

```

        tmp.append(new_el)
        new_el = 0
    mult.append(tmp)
    tmp = []
    return mult
def tofile (file, matrix):
    for line in matrix:
        for number in line:
            file.write(str(number) + ' ')
        file.write('\n')
print("!Кількість стовпчиків першої матриці збігається з кількістю рядків
другої!")
print("n-кількість рядків першої матриці")
print("m-кількість стовпчиків першої матриці")
print("k-кількість стовпчиків другої матриці")
n= int(input('n:'))
m = int(input('m:'))
k = int(input('k:'))
arr1 = arr(n, m)
arr2 = arr(m, k)
start_time = time.time()
result = multiply(arr1, arr2)
print("Час: %s секунд " % (time.time() - start_time))
output1 = open('output1.txt', 'w')
output2 = open('output2.txt', 'w')
output3 = open('output3.txt', 'w')
tofile (output1, arr1)
tofile (output2, arr2)
tofile (output3, result)

```

**Завдання №2.** На основі тих самих програмних засобів, що використовувались в завданні №1. Створити алгоритм паралельного обчислення множення двох матриць, записаних в файлі при виконанні застосунка завдання №1. Результуюча матриця має поелементно порівнюватись з записаною в файл застосунком завдання №1. Час затрачений на виконання корисних обчислень відобразити на моніторі ПК.

Час: 0.10204267501831055 секунд

Рисунок 6. Час витрачений на виконання обчислень

## Код:

```
import time
from multiprocessing.pool import Pool
def get_arr(file):
    rl = file.readlines()
    result = [line.rstrip('\n').split(' ') for line in rl]
    for line in result:
        i = 0
        for string in line:
            line[i] = int(string)
            i = i + 1
    return result
def multiply(arrs):
    el = 0
    tmp = []
    result = []
    arr1, arr2=arrs
    arr1r = len(arr1)
    arr1c = len(arr1[0])
    arr2c = len(arr2[0])
    for z in range(0, arr1r):
        for j in range(0, arr2c):
            for i in range(0, arr1c):
                el = el + arr1[z][i] * arr2[i][j]
            tmp.append(el)
            el = 0
        result.append(tmp)
        tmp = []
    return result

def connection (arr1, arr2):
    i = 0
    result = []
    for row in arr1:
        new_row = row + arr2[i]
        i = i + 1
        result.append(new_row)
    return result

arr1 = get_arr(open('output1.txt', 'r'))
arr2 = get_arr(open('output2.txt', 'r'))
result_1 = get_arr(open('output3.txt', 'r'))
start_time = time.time()
arr1_top = arr1[:int(len(arr1) / 2):]
arr1_bottom = arr1[int(len(arr1) / 2)::]
```

```

arr2_left = [item[:int(len(arr1) / 2):] for item in arr2]
arr2_right = [item[int(len(arr1) / 2)::] for item in arr2]
pool = Pool(processes=4)
pieces = pool.map(multiply, [(arr1_top, arr2_left),
                             (arr1_top, arr2_right),
                             (arr1_bottom, arr2_left),
                             (arr1_bottom, arr2_right)])
pool.close()
pool.join()
result_2_top = connection (pieces[0], pieces[1])
result_2_bottom = connection (pieces[2], pieces[3])
result_2 = result_2_top + result_2_bottom
if (result_2==result_1):
    print("Результуючі матриці співпадають")
print("Час: %s секунд " % (time.time() - start_time))

```

Побудуємо графіки залежностей часу виконання від меншої з розмірностей вхідних матриць для обох застосунків.

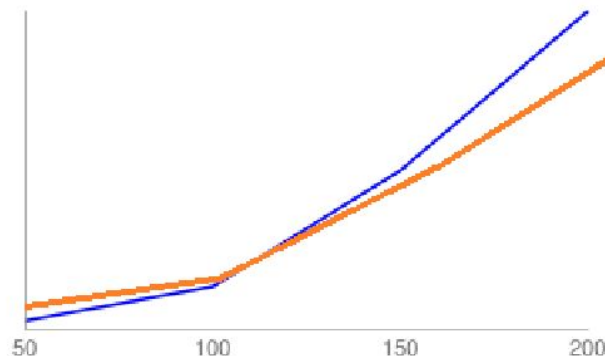


Рисунок 7. Графік

Вісь абсцис відповідає за значення вхідних матриць, вісь ординат за час на виконання обчислень (в с).

**Синім** кольором зображений графік для коду з послідовним виконанням.

**Оранжевим** кольором зображений графік для коду з паралельним виконанням.

Можна зробити висновок, що для матриць невеликого розміру ефективнішим є код з послідовним виконанням (оскільки він займатиме менше часу), а для більших матриць - з паралельним виконанням.