Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

САМОСТІЙНА РОБОТА
З ДИСЦИПЛІНИ “ОСНОВИ ПАРАЛЕЛЬНОГО ПРОГРАМУВАННЯ ”
НА ТЕМУ: “ПУЛИ ПОТОКІВ В JAVA”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 25

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2020 р.

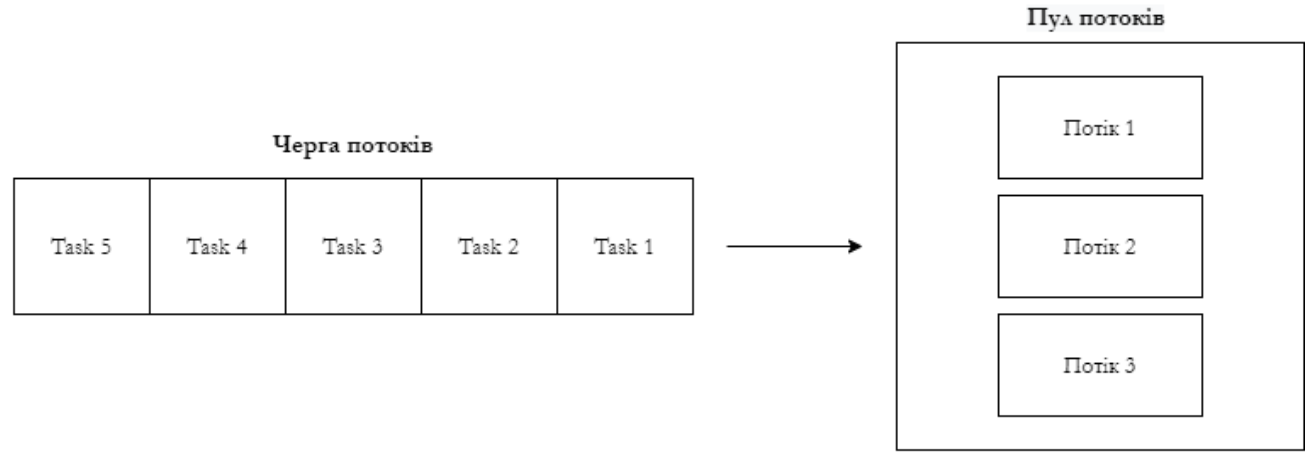
Виконання роботи

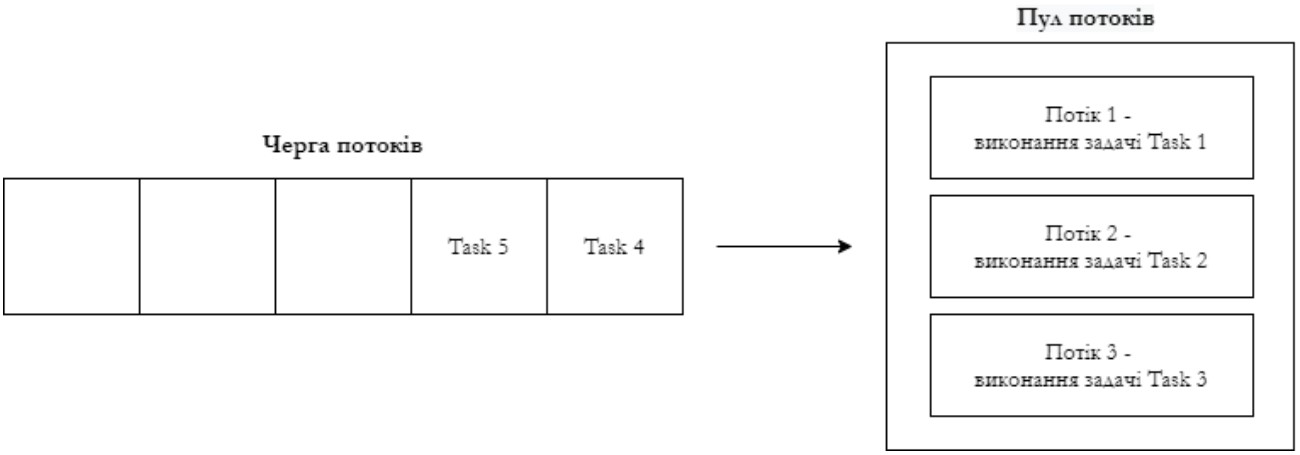
Для створення пулів потоків Java надає структуру Executor, яка зосереджена навколо інтерфейсу Executor, його підінтерфейсу – ExecutorService і класу ThreadPoolExecutor, який реалізує обидва цих інтерфейси. При використанні executor потрібно тільки реалізувати об'єкти Runnable і відправити їх на виконання до executor.

Пули потоків доволі зручний механізм, який дозволяє нам використовувати переваги багатопотоковості, але зосереджувати увагу на задачах, які ми хочемо, щоб потік виконував, а не на механіці потоків.

Щоб використовувати пули потоків, ми спочатку створюємо об'єкт ExecutorService і передаємо йому набір задач. Клас ThreadPoolExecutor дозволяє встановити розмір ядра і максимальний пул. Runnables, які запускаються певним потоком, виконуються послідовно.

На першій схемі внизу зображено ініціалізація пулу потоків із розміром три потоки та черга завдань з п’яти запущених об’єктів. На другій схемі зображено виконання пулом потоків перших трьох завдань.





Висновки

Для реалізації пулів потоків в Java:

1. Було створено 5 задач (Runnable Object) для виконання (рядки 40-44 файлу *PoolThread.java*), реалізація кожної окремої задачі наведена відповідно у файлах *Func1.java*, *Func2.java*, *Func3.java* і т.д з використанням інтерфейсу Runnable.

2. Створено пул потоків фіксованого розміру (розмір в даній реалізації задається змінною T_MAX – 23 рядок файлу *PoolThread.java*) за допомогою методу *newFixedThreadPool(int)* з інтерфейсу Executors. У разі фіксованого пулу потоків, якщо всі потоки в даний час виконуються executor, відкладені задачі поміщаються в чергу і виконуються, коли потік з пулу завершує своє виконання.

3. Передано об'єкти задач (T1, T2, T3 …) до пулу для виконання (рядки 48-52 файлу *PoolThread.java*).

4. Було завершено роботу пулу потоків за допомогою методу **shutdown()** з ExecutorService (рядок 54 файлу *PoolThread.java*).

Лістинг програми

PoolThread.java

```
1. /*-------------------------------------------------
2.   |                    Thread Pools                 |
3.   -------------------------------------------------
4.   |   Author   |    Jack (Yevhenii) Shendrikov      |
5.   |   Group    |                IO-82               |
6.   |   Variant  |                 #25                |
7.   |    Date    |             17.12.2020             |
8.   -------------------------------------------------
9.   | Function 1 |    e = ((A + B)*(C + D*(MA*ME)))   |
10.   | Function 1 |    f = MAX(MG*MK) - MIN(ML + MH)   |
11.   | Function 3 |       O = (SORT(MP*MR)*S)          |
12.   | Function 4 |      MF = (MG*MH)*TRANS(MK)        |
13.   | Function 5 |     S = (MO*MP)*V+t*MR*(O+P)       |
14.   -------------------------------------------------
15. */
16.
17. import java.util.Scanner;
18. import java.util.concurrent.ExecutorService;
19. import java.util.concurrent.Executors;
20.
21. public class PoolThread{
22.
23.     private static final int T_MAX = 3;
24.
25.     public static void main(String[] args) {
26.
27.         Scanner scanner = new Scanner(System.in);
28.         int N;
29.
30.         System.out.print("!!! Note that if the value of N > 10 -> the result will not be
displayed !!!\n" +
31.                 "!!! If you enter N <= 0 - execution will be terminated !!!\n\n" +
"Enter N: ");
```

```
32.         N = scanner.nextInt();
33.
34.         if (N <= 0) throw new ArithmeticException("Restart the program and enter N >
0.");
35.
36.         System.out.println("\nProgram started!\n");
37.
38.         Data data = new Data(N);
39.
40.         Runnable T1 = new Func1("Task 1", data);
41.         Runnable T2 = new Func2("Task 2", data);
42.         Runnable T3 = new Func3("Task 3", data);
43.         Runnable T4 = new Func4("Task 4", data);
44.         Runnable T5 = new Func5("Task 5", data);
45.
46.         ExecutorService pool = Executors.newFixedThreadPool(T_MAX);
47.
48.         pool.execute(T1);
49.         pool.execute(T2);
50.         pool.execute(T3);
51.         pool.execute(T4);
52.         pool.execute(T5);
53.
54.         pool.shutdown();
55.
56.         System.out.println("\nProgram finished!\n");
57.     }
58. }
```

Func1.java

```
1. public class Func1 implements Runnable {
2.
3.     private Data data;
4.     private String t_name;
5.
6.     Func1(String name, Data data){
7.         t_name = name;
8.         this.data = data;
9.     }
10.
11.     // F1 -> D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)
12.     public void run(){
13.         System.out.println(t_name + " started.");
14.
15.         try {
16.             int[] A, B, C;
17.             int[][] MA, ME;
18.
19.             // Input
20.             A = data.allOnesVector(); B = data.allOnesVector(); C =
data.allOnesVector();
21.             MA = data.allOnesMatrix(); ME = data.allOnesMatrix();
22.
23.             Thread.sleep(50);
24.             if (data.getN() < 10) {
25.                 System.out.println("\n----- " + t_name + " Input Data -----");
26.                 data.vectorOutput(A, 'A'); data.vectorOutput(A, 'B');
data.vectorOutput(A, 'C');
27.                 data.matrixOutput(MA, "MA"); data.matrixOutput(MA, "ME");
28.                 System.out.println("----------------------------\n");
29.             }
30.
31.             // Calculation
32.             int[] result = data.func1(A, B, C, MA, ME);
```

```
33.             Thread.sleep(100);
34.
35.             // Output
36.             if (data.getN() < 10) {
37.                 System.out.println("\n----- " + t_name + " Result Data ----");
38.                 data.vectorOutput(result, 'D');
39.                 System.out.println("----------------------------\n");
40.             }
41.         } catch (InterruptedException e) {
42.             e.printStackTrace();
43.         }
44.
45.
46.         System.out.println(t_name + " finished.\n");
47.     }
48. }
```

Func2.java

```
1. public class Func2 implements Runnable {
2.
3.     private Data data;
4.     private String t_name;
5.
6.     Func2(String name, Data data){
7.         t_name = name;
8.         this.data = data;
9.     }
10.
11.     // F2 -> MF = (MG*MH)*TRANS(MK)
12.     public void run(){
13.         System.out.println(t_name + " started.");
14.         try {
15.             int[][] MG, MH, MK;
16.
17.             // Input
18.             Thread.sleep(100);
19.             MG = data.allOnesMatrix(); MH = data.allOnesMatrix(); MK =
data.allOnesMatrix();
20.
21.             if (data.getN() < 10) {
22.                 System.out.println("\n----- " + t_name + " Input Data -----");
23.                 data.matrixOutput(MG, "MG"); data.matrixOutput(MH, "MH");
data.matrixOutput(MK, "MK");
24.                 System.out.println("----------------------------\n");
25.             }
26.
27.             // Calculation
28.             int result[][] = data.func2(MG, MH, MK);
29.             Thread.sleep(100);
30.
31.             // Output
32.             if (data.getN() < 10) {
33.                 System.out.println("\n----- " + t_name + " Result Data ----");
34.                 data.matrixOutput(result, "MF");
35.                 System.out.println("----------------------------\n");
36.             }
37.         } catch (InterruptedException e) {
38.             e.printStackTrace();
39.         }
40.
41.
42.         System.out.println(t_name + " finished.\n");
43.     }
44. }
```

Func3.java

```
1. public class Func3 implements Runnable {
2.
3.     private Data data;
4.     private String t_name;
5.
6.     Func3(String name, Data data){
7.         t_name = name;
8.         this.data = data;
9.     }
10.
11.     // O = (SORT(MP*MR)*S)
12.     public void run(){
13.         System.out.println(t_name + " started.");
14.         try {
15.             int[] S;
16.             int[][] MP, MR;
17.
18.             // Input
19.             Thread.sleep(70);
20.             S = data.allOnesVector();
21.             MP = data.allOnesMatrix(); MR = data.allOnesMatrix();
22.
23.             if (data.getN() < 10) {
24.                 System.out.println("\n----- " + t_name + " Input Data -----");
25.                 data.vectorOutput(S, 'S');
26.                 data.matrixOutput(MP, "MP"); data.matrixOutput(MR, "MR");
27.                 System.out.println("-----------------------------\n");
28.             }
29.
30.             // Calculation
31.             int[] result = data.func3(S, MP, MR);
32.             Thread. sleep(100);
33.
34.             // Output
35.             if (data.getN() < 10) {
36.                 System.out.println("\n----- " + t_name + " Result Data ----");
37.                 data.vectorOutput(result, 'O');
38.                 System.out.println("-----------------------------\n");
39.             }
40.         } catch (InterruptedException e) {
41.             e.printStackTrace();
42.         }
43.
44.
45.         System.out.println(t_name + " finished.\n");
46.     }
47. }
```

Func4.java

```
1. public class Func4 implements Runnable {
2.
3.     private Data data;
4.     private String t_name;
5.
6.     Func4(String name, Data data){
7.         t_name = name;
8.         this.data = data;
9.     }
10.
11.     // e = ((A + B)*(C + D*(MA*ME)))
12.     public void run(){
13.         System.out.println(t_name + " started.");
14.         try {
15.             int[] A, B, C, D;
16.             int[][] MA, ME;
17.
```

```
18.             // Input
19.             A = data.allOnesVector(); B = data.allOnesVector(); C =
data.allOnesVector(); D = data.allOnesVector();
20.             MA = data.allOnesMatrix(); ME = data.allOnesMatrix();
21.
22.             Thread.sleep(30);
23.             if (data.getN() < 15) {
24.                 System.out.println("\n----- " + t_name + " Input Data -----");
25.                 data.vectorOutput(A, 'A'); data.vectorOutput(A, 'B');
data.vectorOutput(A, 'C'); data.vectorOutput(A, 'D');
26.                 data.matrixOutput(MA, "MA"); data.matrixOutput(MA, "ME");
27.                 System.out.println("----------------------------\n");
28.             }
29.
30.             // Calculation
31.             int result = data.func4(A, B, C, D, MA, ME);
32.             Thread.sleep(100);
33.
34.             // Output
35.             if (data.getN() < 10) {
36.                 System.out.println("\n----- " + t_name + " Result Data ----");
37.                 data.numOutput(result, 'e');
38.                 System.out.println("----------------------------\n");
39.             }
40.         } catch (InterruptedException e) {
41.             e.printStackTrace();
42.         }
43.
44.         System.out.println(t_name + " finished.\n");
45.     }
46. }
```

Func5.java

```
1. public class Func5 implements Runnable {
2.
3.     private Data data;
4.     private String t_name;
5.
6.     Func5(String name, Data data){
7.         t_name = name;
8.         this.data = data;
9.     }
10.
11.     // f = MAX(MG*MK) - MIN(ML + MH)
12.     public void run(){
13.         System.out.println(t_name + " started.");
14.         try {
15.             int[][] MG, MK, ML, MH;
16.
17.             // Input
18.             Thread.sleep(60);
19.             MG = data.allOnesMatrix(); MK = data.allOnesMatrix();
20.             ML = data.allOnesMatrix(); MH = data.allOnesMatrix();
21.
22.             if (data.getN() < 10) {
23.                 System.out.println("\n----- " + t_name + " Input Data -----");
24.                 data.matrixOutput(MG, "MG"); data.matrixOutput(MK, "MK");
25.                 data.matrixOutput(ML, "ML"); data.matrixOutput(MH, "MH");
26.                 System.out.println("----------------------------\n");
27.             }
28.
29.             // Calculation
30.             int result = data.func5(MG, MK, ML, MH);
31.             Thread.sleep(100);
32.
33.             // Output
34.             if (data.getN() < 10) {
35.                 System.out.println("\n----- " + t_name + " Result Data ----");
```

```
                data.numOutput(result, 'f');
                System.out.println("----------------------------\n");
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        }


        System.out.println(t_name + " finished.\n");
    }
}
```

Data.java

```
import java.util.Arrays;
import java.util.Random;

class Data {
    private int N;

    Data(int N){
        this.N = N;
    }

    int getN(){
        return N;
    }

    // ----------------------------------------
    int[] allOnesVector(){
        int[] A = new int[N];
        for (int i = 0; i < A.length; i++) {
            A[i] = 1;
        }
        return A;
    }

    int[][] allOnesMatrix(){
        int[][] MA = new int[N][N];
        for (int i = 0; i < MA.length; i++) {
            for (int j = 0; j < MA[i].length; j++) {
                MA[i][j] = 1;
            }
        }
        return MA;
    }

    // ----------------------------------------
    int randomNum(){
        Random random = new Random();
        return random.nextInt(10);
    }

    int[] randomVector(){
        int[] A = new int[N];
        Random random = new Random();
        for (int i = 0; i < A.length; i++) {
            A[i] = random.nextInt(10);
        }
        return A;
    }

    int[][] randomMatrix(){
        int[][] MA = new int[N][N];
        Random random = new Random();
        for (int i = 0; i < MA.length; i++) {
            for (int j = 0; j < MA[i].length; j++) {
                MA[i][j] = random.nextInt(10);
```

```
55.             }
56.         }
57.         return MA;
58.     }
59.
60.     // ----------------------------------------
61.     void matrixOutput(int[][] MA, String name){
62.         System.out.println("\tMatrix " + name + ":");
63.         for (int[] i : MA) {
64.             System.out.print("\t\t");
65.             for (int j : i) {
66.                 System.out.print(j + " ");
67.             }
68.             System.out.println();
69.         }
70.     }
71.
72.     void vectorOutput(int[] A, char name){
73.         System.out.print("\tVector " + name + ": ");
74.         for (int i : A) {
75.             System.out.print(i + " ");
76.         }
77.         System.out.println();
78.     }
79.
80.     void numOutput(int a, char name){
81.         System.out.print("\tNumber " + name + ": " + a + "\n");
82.     }
83.
84.     // ----------------------------------------
85.     private int maxMatrix(int[][] MA){
86.         int max = MA[0][0];
87.         for (int i = 0; i < N ; i++){
88.             for (int j = 0; j < N ; j++) {
89.                 if (MA[i][j] > max) {
90.                     max = MA[i][j];
91.                 }
92.             }
93.         }
94.         return max;
95.     }
96.
97.     private int minMatrix(int[][] MA){
98.         int min = MA[0][0];
99.         for (int i = 0; i < N ; i++){
100.             for (int j = 0; j < N ; j++) {
101.                 if (MA[i][j] < min) {
102.                     min = MA[i][j];
103.                 }
104.             }
105.         }
106.         return min;
107.     }
108.
109.     private int[][] matrixTransp(int[][] MA){
110.         int buf;
111.         for (int i = 0; i < MA.length ; i++){
112.             for (int j = 0; j <=i; j++){
113.                 buf = MA[i][j];
114.                 MA[i][j] = MA[j][i];
115.                 MA[j][i] = buf;
116.             }
117.         }
118.         return MA;
119.     }
120.
121.     // ----------------------------------------
122.     private int[] sortVector(int[] A){
```

```
123.         Arrays.sort(A);
124.         return A;
125.     }
126.
127.     private int[][] sortMatrix(int[][] MA){
128.         int temp[] = new int[N * N];
129.         int k = 0;
130.
131.         for (int i = 0; i < N; i++)
132.             for (int j = 0; j < N; j++)
133.                 temp[k++] = MA[i][j];
134.
135.         // sort temp[]
136.         Arrays.sort(temp);
137.
138.         k = 0;
139.         for (int i = 0; i < N; i++)
140.             for (int j = 0; j < N; j++)
141.                 MA[i][j] = temp[k++];
142.
143.         return MA;
144.     }
145.
146.     // ----------------------------------------
147.     private int[] sumVectors(int[] A, int[] B){
148.         int[] C = new int[N];
149.         for (int i = 0; i < N ; i++){
150.             C[i] = A[i] + B[i];
151.         }
152.         return C;
153.     }
154.
155.     private int[][] sumMatrix(int[][] MA, int[][] MB){
156.         int[][] MC = new int[N][N];
157.         for (int i = 0; i < N ; i++)
158.             for (int j = 0; j < N ; j++)
159.                 MC[i][j] = MA[i][j] + MB[i][j];
160.         return MC;
161.     }
162.
163.     // ----------------------------------------
164.     private int vectorMult(int[] A, int[] B){
165.         int c = 0;
166.         for (int i = 0; i < N ; i++){
167.             c += A[i] * B[i];
168.         }
169.         return c;
170.     }
171.
172.     private int[][] matrixMult(int[][] MA, int[][] MB){
173.         int[][] MC = new int[N][N];
174.         for (int i = 0; i < N ; i++){
175.             for (int j = 0; j < N ; j++){
176.                 for (int k = 0; k < N ; k++){
177.                     MC[i][j] += MA[i][k] * MB[k][j];
178.                 }
179.             }
180.         }
181.         return MC;
182.     }
183.
184.     private int[] vectorMatrixMult(int[] A, int[][] MA){
185.         int[] B = new int[N];
186.         for (int i = 0; i < N ; i++){
187.             for (int j = 0; j < N ; j++){
188.                 B[i] += A[j] * MA[j][i];
189.             }
190.         }
```

```
191.         return B;
192.     }
193.
194.     // ----------------------------------------
195.     // F1 -> D = SORT(A)+SORT(B)+SORT(C)*(MA*ME)
196.     int[] func1(int[] A, int[] B, int[] C, int[][] MA, int[][] ME){
197.         return sumVectors(sumVectors(sortVector(A), sortVector(B)),
vectorMatrixMult(sortVector(C), matrixMult(MA, ME)));
198.     }
199.
200.     // F2 -> MF = (MG*MH)*TRANS(MK)
201.     int[][] func2(int[][] MG, int[][] MH, int[][] MK){
202.         return matrixMult(matrixMult(MG, MH), matrixTransp(MK));
203.     }
204.
205.     // F3 ->  O = (SORT(MP*MR)*S)
206.     int[] func3(int[] S,int[][] MP, int[][] MR){
207.         return vectorMatrixMult(S, sortMatrix(matrixMult(MP, MR)));
208.     }
209.
210.     // F4 -> e = ((A + B)*(C + D*(MA*ME)))
211.     int func4(int[] A, int[] B, int[] C, int[] D, int[][] MA, int[][] ME){
212.         return vectorMult(sumVectors(A, B), sumVectors(C, vectorMatrixMult(D,
matrixMult(MA, ME))));
213.     }
214.
215.     // F5 -> f = MAX(MG*MK) - MIN(ML + MH)
216.     int func5(int[][] MG, int[][] MK, int[][] ML, int[][] MH){
217.         return maxMatrix(matrixMult(MG, MK)) - minMatrix(sumMatrix(ML, MH));
218.     }
219. }
```

Приклад роботи програми

```
!!! Note that if the value of N > 10 -> the result will not be displayed !!!
!!! If you enter N <= 0 - execution will be terminated !!!

Enter N: 3

Program started!

Task 2 started.
Task 3 started.
Task 1 started.

Program finished!


----- Task 1 Input Data -----
      Vector A: 1 1 1
      Vector B: 1 1 1
      Vector C: 1 1 1
      Matrix MA:
            1 1 1
            1 1 1
            1 1 1
      Matrix ME:
            1 1 1
            1 1 1
            1 1 1
----------------------------
```

```
----- Task 3 Input Data -----
      Vector S: 1 1 1
      Matrix MP:
            1 1 1
            1 1 1
            1 1 1
      Matrix MR:
            1 1 1
            1 1 1
            1 1 1
----------------------------


----- Task 2 Input Data -----
      Matrix MG:
            1 1 1
            1 1 1
            1 1 1
      Matrix MH:
            1 1 1
            1 1 1
            1 1 1
      Matrix MK:
            1 1 1
            1 1 1
            1 1 1
----------------------------


----- Task 1 Result Data ----
      Vector D: 11 11 11
----------------------------

Task 1 finished.

Task 4 started.

----- Task 3 Result Data ----
      Vector O: 9 9 9
----------------------------

Task 3 finished.

Task 5 started.

----- Task 4 Input Data -----
      Vector A: 1 1 1
      Vector B: 1 1 1
      Vector C: 1 1 1
      Vector D: 1 1 1
      Matrix MA:
            1 1 1
```

```
            1 1 1
            1 1 1
      Matrix ME:
            1 1 1
            1 1 1
            1 1 1
----------------------------


----- Task 2 Result Data ----
      Matrix MF:
            9 9 9
            9 9 9
            9 9 9
----------------------------

Task 2 finished.


----- Task 5 Input Data -----
      Matrix MG:
            1 1 1
            1 1 1
            1 1 1
      Matrix MK:
            1 1 1
            1 1 1
            1 1 1
      Matrix ML:
            1 1 1
            1 1 1
            1 1 1
      Matrix MH:
            1 1 1
            1 1 1
            1 1 1
----------------------------


----- Task 4 Result Data ----
      Number e: 60
----------------------------

Task 4 finished.


----- Task 5 Result Data ----
      Number f: 1
----------------------------

Task 5 finished.


Process finished with exit code 0
```