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

ЛАБОРАТОРНА РОБОТА №1

З ДИСЦИПЛІНИ “ ПАРАЛЕЛЬНІ ТА РОЗПОДІЛЕНІ ОБЧИСЛЕННЯ”

НА ТЕМУ: “ Семафори, мютекси, події, критичні секції у WinAPI”

Виконав:
Студент III курсу ФІОТ
групи ІО-82
Шендріков Євгеній
Номер у списку - 24

Перевірив:
Доцент
Корочкін О. В.

м. Київ – 2021 р.

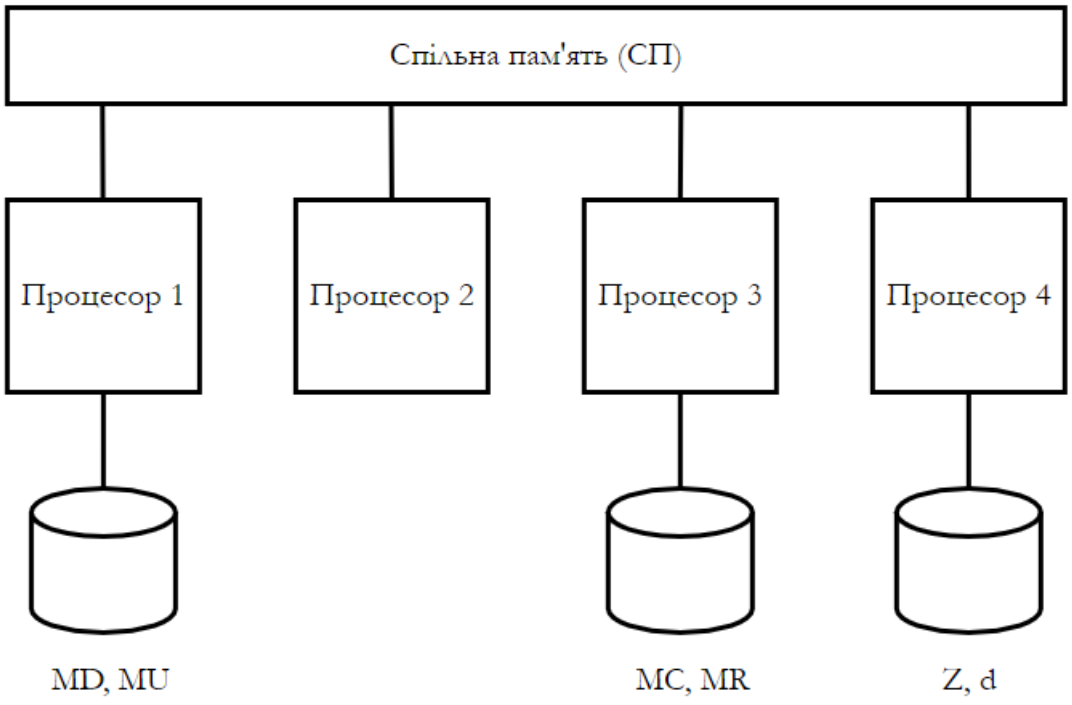
Технічне завдання

1. Розробити паралельний алгоритм рішення математичної задачі $MU = MD * MC * d + max(Z) * MR$ з використанням бібліотеки WinAPI на мові C++;
2. Виявити спільні ресурси;
3. Описати алгоритм кожного потоку (T1 – Tp) з визначенням критичних ділянок (КД) і точок синхронізації (W_{ij} , S_{ij});
4. Розробити структурну схему взаємодії задач, де застосувати всі вказані засоби взаємодії процесів;
5. Розробити програму (обов'язкові "шапка", коментарі);
6. Виконати налагодження програми;
7. Отримати правильні результати обчислень;
8. За допомогою Диспетчеру задач Windows проконтролювати завантаження ядер процесору.

Засоби організації взаємодії: семафори, мютекси, критичні секції, події;

Засоби взаємодії: семафори.

Структурна схема ПКС



Виконання роботи

Крок 1. Побудова паралельного алгоритму

1) $m_i = \max(Z_H)$, $i = \overline{1, P}$
2) $m = \max(m_i, m)$
3) $MU_H = MD_H \cdot MC \cdot d + m \cdot MR_H$

Спільний ресурс: m, d, MC

Крок 2. Розроблення алгоритмів роботи кожного процесу

Задача T1		ТС та КД
1	Введення MD	
2	Сигнал задачам T2, T3, T4 про введення MD	$S_{2,3,4-1}$
3	Чекати на введення MR, MC у задачі T3	W_{3-1}
4	Чекати на введення Z, d у задачі T4	W_{4-2}
5	Копіювати MC1 := MC	КД
6	Копіювати d1:= d	КД
7	Обчислення m1 := $\max(Z_H)$	
8	Обчислення m: = max (m, m1)	КД
9	Сигнал T3 про завершення обчислень m	S_{3-2}
10	Чекати сигналу T3 про завершення обчислень m	W_{3-3}
10	Копіювання m1: = m	КД
12	Обчислення $MU_H = MD_H \cdot MC1 \cdot d1 + m1 \cdot MK_H$	
13	Сигнал T3 про завершення обчислень MU	S_{3-3}

Задача T2		ТС та КД
1	Чекати на введення MD у задачі T1	W_{1-1}
2	Чекати на введення MR, MC у задачі T3	W_{3-2}
3	Чекати на введення Z, d у задачі T4	W_{4-3}
4	Копіювати MC2:= MC	
5	Копіювати d2:= d	КД
6	Обчислення m2 := $\max(Z_H)$	
7	Обчислення m: = max(m,m2)	КД
8	Сигнал T3 про завершення обчислень m	S_{3-1}
9	Чекати сигналу T3 про завершення обчислень m	W_{3-4}
10	Копіювання m2: = m	КД
11	Обчислення $MU_H = MD_H \cdot MC2 \cdot d2 + m2 \cdot MK_H$	
12	Сигнал T3 про завершення обчислень MU	S_{3-3}

Задача T3		ТС та КД
1	Введення MR, MC	
2	Сигнал задачам T1, T2, T4 про введення MR, MC	$S_{1,3,4-1}$
3	Чекати на введення MD у задачі T1	W_{1-1}
4	Чекати на введення Z, d у задачі T4	W_{4-2}

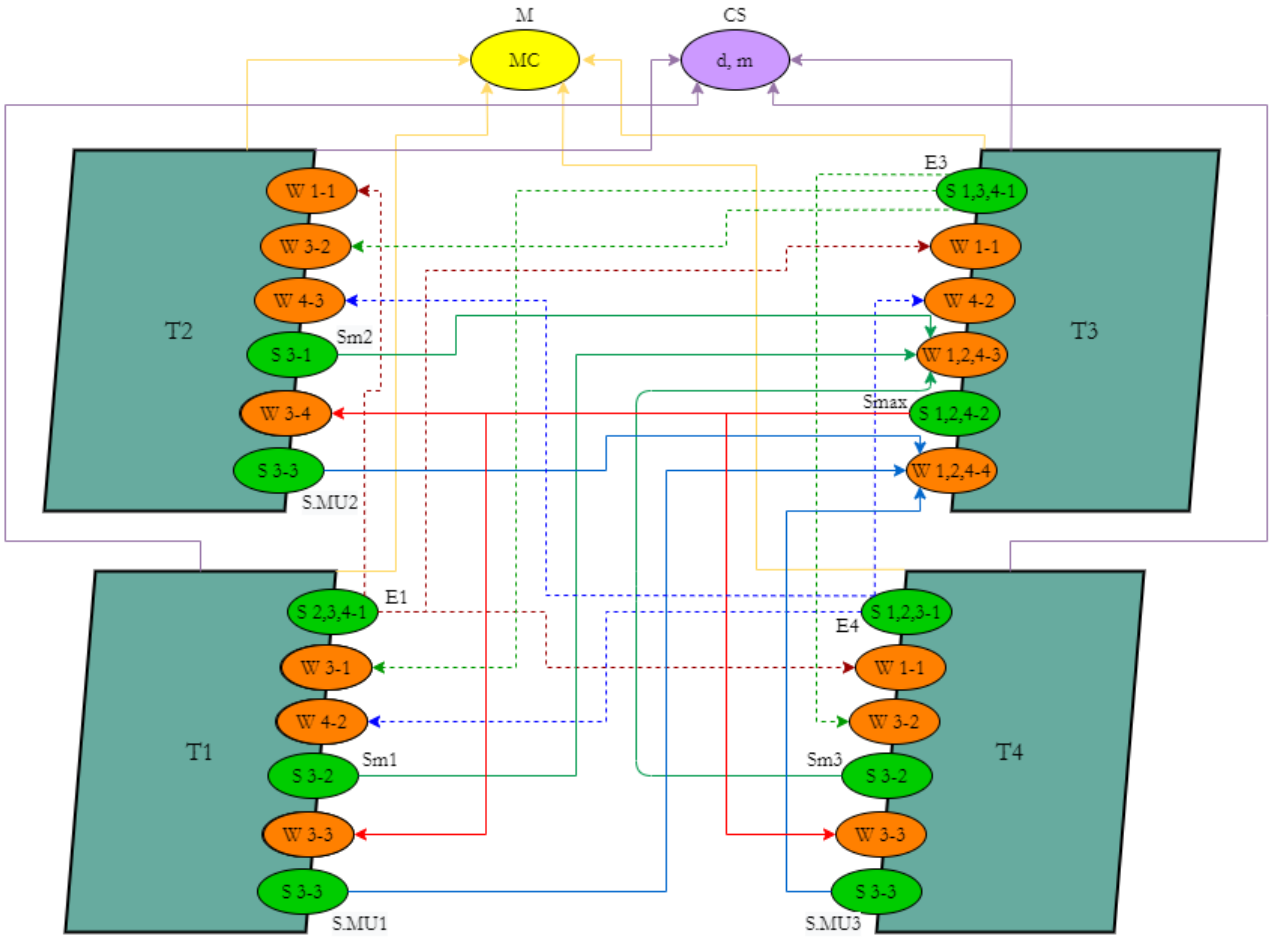
5	Копіювати MC3:= MC	КД
6	Копіювати d3:= d	КД
7	Обчислення m3 := max(Z_H)	
8	Обчислення m: = max(m,m3)	КД
9	Чекати на завершення обчислень m в T1, T2, T4	$W_{1,2,4-3}$
11	Копіювання m3: = m	КД
12	Сигнал T1, T2, T4 про завершення обчислень m	$S_{1,2,4-2}$
12	Обчислення $MU_H = MD_H \cdot MC3 \cdot d3 + m3 \cdot MK_H$	
13	Чекати на завершення обчислень MU в T1, T2, T4	$W_{1,2,4-4}$
14	Виведення MU	

Задача T4		ТС та КД
1	Введення Z, d	
2	Сигнал задачам T1, T2, T3 про введення Z, d	$S_{1,2,3-1}$
3	Чекати на введення MD у задачі T1	W_{1-1}
4	Чекати на введення MR, MC у задачі T3	W_{3-2}
5	Копіювати MC4:= MC	КД
6	Копіювати d4:= d	КД
7	Обчислення m4 := max(Z_H)	
8	Обчислення m: = max(m,m4)	КД
9	Сигнал T3 про завершення обчислень m	S_{3-2}
10	Чекати сигналу T3 про завершення обчислень m	W_{3-3}
11	Копіювання m4: = m	КД
12	Обчислення $MA_H = MB_H \cdot MC4 \cdot d4 + m4 \cdot MK_H$	
13	Сигнал T3 про завершення обчислень MA	S_{3-3}

Крок 3. Розроблення структурної схеми взаємодії задач

Умовні позначення на структурній схемі:

- CS – для доступу до спільного ресурсу d, m;
- M – для доступу до спільного ресурсу MC;
- E1 – для синхронізації із завершенням вводу в T1;
- E3 – для синхронізації із завершенням вводу в T3;
- E4 – для синхронізації із завершенням вводу в T4;
- Sm1, Sm2, Sm3, Smax – для синхронізації обчислень максимуму Z;
- S.MA1, S.MA2, S.MA3 – для синхронізації решти обчислень і виведення результату.



Крок 4. Розробка програми

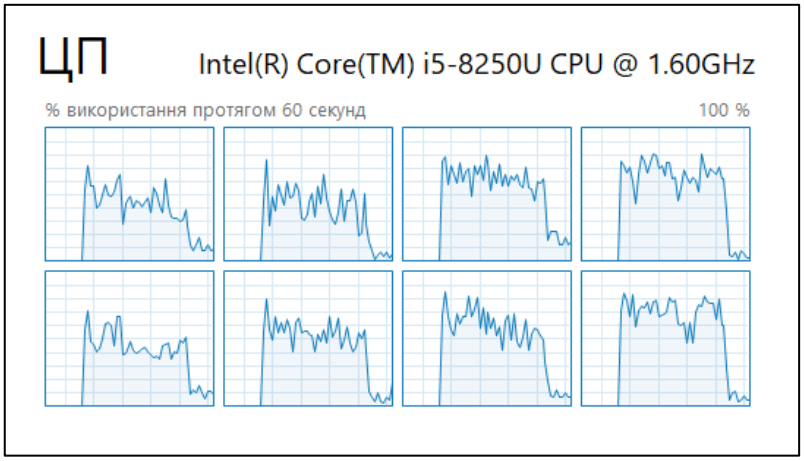
Результат роботи

```
Lab 1 started!

T4 started.
T2 started.
T1 started.
T3 started.

T4 finished.
T2 finished.
T1 finished.
T3 finished.

Lab 1 finished!
```



Висновки

1. На основі засобів бібліотеки MPI на C++ було розроблено програму та паралельний алгоритм для рішення математичної задачі заданої за варіантом.

2. Було описано алгоритм кожного потоку (T1 – T4) з визначенням критичних ділянок (КД) та точок синхронізації (Wij , Sij);

3. Розроблено структурну схему взаємодії задач, де було застосовано всі вказані в завданні засоби взаємодії процесів. Для доступу до спільного ресурсу використовувались критичні ділянки та мютекс, для синхронізації обчислень – бінарні семафори, а для синхронізації потоків – події.

4. Було перевірено працездатність програми, а також проконтрольовано завантаження ядер процесору за допомогою Диспетчеру задач. Програма забезпечує 80% завантаженості.

Лістинг коду

```
/*-------------------------------------------------------
 |                      Labwork #1                      |
 |                   PKS SP in WinAPI                   |
 -------------------------------------------------------
 |  Author  |       Jack (Yevhenii) Shendrikov          |
 |  Group   |                 IO-82                     |
 |  Variant |                  #24                      |
 |  Date    |             10.02.2021                    |
 -------------------------------------------------------
  | Function |       MU = MD*MC*d + max(Z)*MR           |
  ---------------------------------------------------------
*/


#include <iostream>
#include <windows.h>

using namespace std;

typedef int* vector;
typedef int** matrix;

const int N = 2000;
const int P = 4;
const int H = N / P;

int d, m;
vector Z = new int[N];
```

```
29. matrix MU = new vector[N],
30.
31. MD = new vector[N],
32. MC = new vector[N],
33. MR = new vector[N];
34. //------- Визначення засобів взаємодії задач -------
35. HANDLE E1, E3, E4;
36. MTX, S_m[3], S_MU[3], S_max;
37. CRITICAL_SECTION CS;
38.
39. //----------------------------------------T1-----------------------------
--------------
40. void T1() {
41.   int d1, m1, s;
42.   matrix MC1 = new vector[N];
43.   for (int i = 0; i < N; i++)
44.         MC1[i] = new int[N];
45.
46.   cout << "T1 started." << endl;
47.
48.   // 1 - Введення MD
49.   for (int i = 0; i < N; i++)
50.         MD[i] = new int[N];
51.
52.   for (int i = 0; i < N; i++)
53.         for (int j = 0; j < N; j++)
54.               MD[i][j] = 1;
55.
56.   // 2 - Сигнал задачам T2, T3, T4 про введення MB
57.   SetEvent(E1);
58.
59.   // 3 - Чекати на введення MR, MC у задачі T3
60.   WaitForSingleObject(E3, INFINITE);
61.
62.   // 4 - Чекати на введення Z, d у задачі T4
63.   WaitForSingleObject(E4, INFINITE);
64.
65.   // 5 - Копіювати MC1 := MC
66.   WaitForSingleObject(MTX, INFINITE);
67.   for (int i = 0; i < N; i++)
68.         for (int j = 0; j < N; j++)
69.               MC1[i][j] = MC[i][j];
70.
71.   ReleaseMutex(MTX);
72.
73.   // 6 - Копіювання d1 := d
74.   EnterCriticalSection(&CS);
75.   d1 = d;
76.   LeaveCriticalSection(&CS);
77.
78.   // 7 - Обчислення m1 := max(Zн)
79.   m1 = 0;
80.   for (int i = 0; i < H; i++)
81.         if (Z[i] > m1) m1 = Z[i];
```

```
82.
83.   // 8 - Обчислення m: = max(m, m1)
84.   EnterCriticalSection(&CS);
85.   m = max(m, m1);
86.   LeaveCriticalSection(&CS);
87.
88.   // 9 - Сигнал T3 про про завершення обчислень m
89.   ReleaseSemaphore(S_m[0], 1, NULL);
90.
91.   // 10 - Чекати сигналу від T3 про завершення обчислень m
92.   WaitForSingleObject(S_max, INFINITE);
93.
94.   // 11 - Копіювання m1 := m
95.   EnterCriticalSection(&CS);
96.   m1 = m;
97.   LeaveCriticalSection(&CS);
98.
99.   // 12 - Обчислення MUн = MDн·MC1·d1 + m1·MRн
100.  for (int i = H; i < 2 * H; i++) {
101.        for (int j = 0; j < N; j++) {
102.              s = 0;
103.              for (int k = 0; k < N; k++) {
104.                    s += MD[i][k] * MC1[k][j];
105.              }
106.              MU[i][j] = s * d1 + m1 * MD[i][j];
107.        }
108.  }
109.
110.  // 13 - Сигнал T3 про завершення обчислень MU
111.  ReleaseSemaphore(S_MU[0], 1, NULL);
112.
113.  cout << "T1 finished." << endl;
114.
115. }
116.
117.
118. //-------------------------------------------T2---------------------------
----------------
119. void T2() {
120.  int d2, s, m2;
121.  matrix MC2 = new vector[N];
122.  for (int i = 0; i < N; i++)
123.        MC2[i] = new int[N];
124.
125.  cout << "T2 started." << endl;
126.
127.  // 1 - Чекати на введення MD у задачі T1
128.  WaitForSingleObject(E1, INFINITE);
129.
130.  // 2 - Чекати на введення MR, MC у задачі T3
131.  WaitForSingleObject(E3, INFINITE);
132.
133.  // 3 - Чекати на введення Z, d у задачі T4
134.  WaitForSingleObject(E4, INFINITE);
```

```
135.
136.  // 4 - Копіювати MC2 := MC
137.  WaitForSingleObject(MTX, INFINITE);
138.  for (int i = 0; i < N; i++)
139.        for (int j = 0; j < N; j++)
140.              MC2[i][j] = MC[i][j];
141.
142.  ReleaseMutex(MTX);
143.
144.  // 5 - Копіювання d
145.  EnterCriticalSection(&CS);
146.  d2 = d;
147.  LeaveCriticalSection(&CS);
148.
149.  // 6 - Обчислення m2 := max(Zн)
150.  m2 = 0;
151.  for (int i = H; i < 2 * H; i++)
152.        if (Z[i] > m2) m2 = Z[i];
153.
154.  // 7- Обчислення m: = max(m, m2)
155.  EnterCriticalSection(&CS);
156.  m = max(m, m2);
157.  LeaveCriticalSection(&CS);
158.
159.  // 8 - Сигнал Т3 про про завершення обчислень m
160.  ReleaseSemaphore(S_m[1], 1, NULL);
161.
162.  // 9 - Чекати сигналу від Т3 про завершення обчислень m
163.  WaitForSingleObject(S_max, INFINITE);
164.
165.  // 10 - Копіювання m2 := m
166.  EnterCriticalSection(&CS);
167.  m2 = m;
168.  LeaveCriticalSection(&CS);
169.
170.  // 11 - Обчислення MUн = MDн·MC2·d2 + m2·MRн
171.  for (int i = H; i < 2 * H; i++) {
172.        for (int j = 0; j < N; j++) {
173.              s = 0;
174.              for (int k = 0; k < N; k++) {
175.                    s += MD[i][k] * MC2[k][j];
176.              }
177.              MU[i][j] = s * d2 + m2 * MD[i][j];
178.        }
179.  }
180.
181.  // 12 - Сигнал Т3 про завершення обчислень MU
182.  ReleaseSemaphore(S_MU[1], 1, NULL);
183.
184.  cout << "T2 finished." << endl;
185.
186. }
187.
188.
```

```
189. //----------------------------------------T3-------------------------------
----------------
190. void T3() {
191.  int d3, s, m3;
192.  matrix MC3 = new vector[N];
193.  for (int i = 0; i < N; i++) {
194.        MC3[i] = new int[N];
195.        MU[i] = new int[N];
196.  };
197.
198.  cout << "T3 started.\n" << endl;
199.
200.  // 1 - Введення MR, MC
201.  for (int i = 0; i < N; i++) {
202.        MD[i] = new int[N];
203.        MC[i] = new int[N];
204.  };
205.
206.  for (int i = 0; i < N; i++) {
207.        for (int j = 0; j < N; j++) {
208.              MD[i][j] = 1;
209.              MC[i][j] = 1;
210.        }
211.  }
212.
213.  // 2 - Сигнал задачам T1, T3, T4 про введення MR, MC
214.  SetEvent(E3);
215.
216.  // 3 - Чекати на введення MD у задачі T1
217.  WaitForSingleObject(E3, INFINITE);
218.
219.  // 4 - Чекати на введення Z, d у задачі T4
220.  WaitForSingleObject(E4, INFINITE);
221.
222.  // 5 - Копіювати MC3 := MC
223.  WaitForSingleObject(MTX, INFINITE);
224.  for (int i = 0; i < N; i++)
225.        for (int j = 0; j < N; j++)
226.              MC3[i][j] = MC[i][j];
227.
228.  ReleaseMutex(MTX);
229.
230.  // 6 - Копіювання d
231.  EnterCriticalSection(&CS);
232.  d3 = d;
233.  LeaveCriticalSection(&CS);
234.
235.  // 7 - Обчислення m3 := max(Zн)
236.  m3 = 0;
237.  for (int i = 2 * H; i < 3 * H; i++)
238.        if (Z[i] > m3) m3 = Z[i];
239.
240.  // 8 - Обчислення m: = max(m, m3)
241.  EnterCriticalSection(&CS);
```

```
242.  m = max(m, m3);
243.  LeaveCriticalSection(&CS);
244.
245.  // 9 - Чекати на завершення обчислень m в T1, T2, T4
246.  WaitForMultipleObjects(3, S_m, TRUE, INFINITE);
247.
248.  // 10 - Копіювання m1: = m
249.  EnterCriticalSection(&CS);
250.  m3 = m;
251.  LeaveCriticalSection(&CS);
252.
253.  // 11 - Сигнал T1, T2, T4 про завершення обчислень m
254.  ReleaseSemaphore(S_max, 1, NULL);
255.
256.  // 12 - Обчислення MUн = MDн·MC3·d + m3·MRн
257.  for (int i = 0; i < H; i++) {
258.        for (int j = 0; j < N; j++) {
259.              s = 0;
260.              for (int k = 0; k < N; k++) {
261.                    s += MD[i][k] * MC3[k][j];
262.              }
263.              MU[i][j] = s * d3 + m3 * MD[i][j];
264.        }
265.  }
266.
267.  // 13 - Чекати на завершення обчислень MU в T1, T2, T4
268.  WaitForMultipleObjects(3, S_MU, TRUE, INFINITE);
269.
270.  // 14 - Виведення MU
271.  if (N < 10) {
272.        for (int i = 0; i < N; i++) {
273.              for (int j = 0; j < N; j++) {
274.                    cout << MU[i][j];
275.              }
276.              cout << endl;
277.        }
278.  }
279.  cout << "T3 finished." << endl;
280.
281. }
282.
283.
284. //--------------------------------------T4--------------------------------
------------------
285. void T4() {
286.  int d4, s, m4;
287.  matrix MC4 = new vector[N];
288.  for (int i = 0; i < N; i++)
289.        MC4[i] = new int[N];
290.
291.  cout << "T4 started." << endl;
292.
293.  // 1 - Введення Z, d
294.  for (int i = 0; i < N; i++)
```

```
295.       Z[i] = 1;
296.
297.  d = 1;
298.
299.  //2. Сигнал задачам T1, T2, T3 про введення Z, d
300.  SetEvent(E4);
301.
302.  //3. Чекати на введення MD у задачі T1
303.  WaitForSingleObject(E1, INFINITE);
304.
305.  //4. Чекати на введення MR, MC у задачі T3
306.  WaitForSingleObject(E3, INFINITE);
307.
308.  //5. Копіювати MC4 := MC
309.  WaitForSingleObject(MTX, INFINITE);
310.  for (int i = 0; i < N; i++)
311.       for (int j = 0; j < N; j++)
312.            MC4[i][j] = MC[i][j];
313.  ReleaseMutex(MTX);
314.
315.  // 6 - Копіювання d
316.  EnterCriticalSection(&CS);
317.  d4 = d;
318.  LeaveCriticalSection(&CS);
319.
320.  // 7 - Обчислення m4 := max(Zн)
321.  m4 = 0;
322.  for (int i = 3 * H; i < N; i++)
323.       if (Z[i] > m4) m4 = Z[i];
324.
325.  // 8 - Обчислення m: = max(m, m4)
326.  EnterCriticalSection(&CS);
327.  m = max(m, m4);
328.  LeaveCriticalSection(&CS);
329.
330.  // 9 - Сигнал T3 про про завершення обчислень m
331.  ReleaseSemaphore(S_m[2], 1, NULL);
332.
333.  // 10 - Чекати сигналу від T3 про завершення обчислень m
334.  WaitForSingleObject(S_max, INFINITE);
335.
336.  // 11 - Копіювання m4: = m
337.  EnterCriticalSection(&CS);
338.  m4 = m;
339.  LeaveCriticalSection(&CS);
340.
341.  // 12 - Обчислення MUн = MDн·MC4·d4 + m4·MRн
342.  for (int i = 3 * H; i < N; i++) {
343.       for (int j = 0; j < N; j++) {
344.            s = 0;
345.            for (int k = 0; k < N; k++) {
346.                 s += MD[i][k] * MC4[k][j];
347.            }
348.            MU[i][j] = s * d4 + m4 * MD[i][j];
```

```
349.        }
350.  }
351.
352.  // 13 - Сигнал T3 про завершення обчислень MU
353.  ReleaseSemaphore(S_MU[2], 1, NULL);
354.
355.  cout << "T4 finished." << endl;
356. }
358.
359.
360. int main(int argc, char* argv[]) {
361.  cout << "Lab 1 started!\n" << endl;
362.
363.  E4 = CreateEvent(NULL, TRUE, FALSE, NULL);
364.  E3 = CreateEvent(NULL, TRUE, FALSE, NULL);
365.  E1 = CreateEvent(NULL, TRUE, FALSE, NULL);
366.
367.  InitializeCriticalSection(&CS);
368.
369.  MTX = CreateMutex(NULL, FALSE, NULL);
370.
371.  S_m[0] = CreateSemaphore(NULL, 0, 1, NULL);
372.  S_m[1] = CreateSemaphore(NULL, 0, 1, NULL);
373.  S_m[2] = CreateSemaphore(NULL, 0, 1, NULL);
374.
375.  DWORD tid1, tid2, tid3, tid4;
376.  HANDLE threads[] = {
377.         CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)T1, NULL, NULL,
&tid1),
378.         CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)T2, NULL, NULL,
&tid2),
379.         CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)T3, NULL, NULL,
&tid3),
380.         CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)T4, NULL, NULL,
&tid4)
381.  };
382.
383.  WaitForMultipleObjects(4, threads, true, INFINITE);
384.
385.  SetThreadPriority(threads[0], THREAD_PRIORITY_NORMAL);
386.  SetThreadPriority(threads[1], THREAD_PRIORITY_NORMAL);
387.  SetThreadPriority(threads[2], THREAD_PRIORITY_HIGHEST);
388.  SetThreadPriority(threads[3], THREAD_PRIORITY_LOWEST);
389.
390.  CloseHandle(threads[0]);
391.  CloseHandle(threads[1]);
392.  CloseHandle(threads[2]);
393.  CloseHandle(threads[3]);
394.
395.  cout << "\nLab 1 finished!\n" << endl;
396.  char key;
397.  cin >> key;
399.  return 0;
400. }
```