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

**Лабораторна робота №5**

з дисципліни

«Програмне забезпечення високопродуктивних комп’ютерних систем»

**Тема: «Бібліотека МРІ»**

| Виконав:  студент групи ІП-05  Гапій Денис Едуардович  номер у списку групи: 4 | Перевірив:  Корочкін О. В. |
| --- | --- |

Київ 2023

**Мета**: Розробка програми для паралельної комп’ютерної системи з локальною пам’яттю, мовою C++. Використати повідомлення (Send | Recv) бібліотеки MPI для організації взаємодії між потоками.

**Варіант**: 9

**Технічне Завдання:**

a = min(B\*MZ + Z\*(MR\*MX))

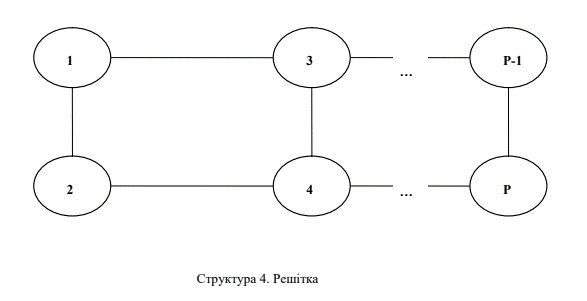
**Введення по потоках:**

* 1: MX, B, Z , MZ, MR
* 2: -
* 3: -
* 4: -

**Виведення в потоці:**

* 1: a

**Структура ПКС ЛП:**



**Виконання Роботи**

**[Етап 1] Розробка паралельного алгоритму:**

| Обчислення | Формула |
| --- | --- |
| 1 | Ah= B\*MZh + Z\*(MR\*MXh) |
| 2 | ai = min(Ah) |
| 3 | a = min(a, ai) i: 1..8 |

**[Етап 2] Розроблення алгоритмів роботи кожного процесу:**

Задача T1

| 1 | Введення **MX, B, Z , MZ, MR** |
| --- | --- |
| 2 | Передати **MX4h, B4h, Z4h, MZ4h, MR4h** в **T2** |
| 3 | Передати **MX3h, B3h, Z3h, MZ3h, MR3h** в **T3** |
| 4 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 5 | **Обчислення 2 [**a\_1= min(Ah)] |
| 6 | Прийняти **a2** з **T2** |
| 7 | **Обчислення** a= min(a1, a2) |
| 8 | Прийняти **a3** з **T3** |
| 9 | **Обчислення** a= min(а,а 3) |
| 10 | Виведення **a** |

Задача T2

| 1 | Прийняти **MX4h, B4h, Z4h, MZ4h, MR4h** з **T1** |
| --- | --- |
| 2 | Передати **MX3h, B3h, Z3h, MZ3h, MR3h** в **T4** |
| 3 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 4 | **Обчислення 2 [**a\_2= min(Ah)] |
| 5 | Прийняти **a** з **T4** |
| 6 | **Обчислення 3 [**a= min(a, a\_2)] |
| 7 | Передати **а** в **T1** |

Задача T3

| 1 | Прийняти **MX3h, B3h, Z3h, MZ3h, MR3h** з **T1** |
| --- | --- |
| 2 | Передати **MX2h, B2h, Z2h, MZ2h, MR2h** в **T5** |
| 3 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 4 | **Обчислення 2 [**a\_3= min(Ah)] |
| 5 | Прийняти **a** з **T5** |
| 6 | **Обчислення 3 [**a= min(a, a\_3)] |
| 7 | Передати **а** в **T1** |

Задача T4

| 1 | Прийняти **MX3h, B3h, Z3h, MZ3h, MR3h** з **T2** |
| --- | --- |
| 2 | Передати **MX2h, B2h, Z2h, MZ2h, MR2h** в **T6** |
| 3 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 4 | **Обчислення 2 [**a\_4= min(Ah)] |
| 5 | Прийняти **a** з **T6** |
| 6 | **Обчислення 3 [**a= min(a, a\_4)] |
| 7 | Передати **а** в **T2** |

Задача T5

| 1 | Прийняти **MX2h, B2h, Z2h, MZ2h, MR2h** з **T3** |
| --- | --- |
| 2 | Передати **MXh, Bh, Zh, MZh, MRh** в **T7** |
| 3 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 4 | **Обчислення 2 [**a\_5= min(Ah)] |
| 5 | Прийняти **a** з **T7** |
| 6 | **Обчислення 3 [**a= min(a, a\_5)] |
| 7 | Передати **а** в **T3** |

Задача T6

| 1 | Прийняти **MX2h, B2h, Z2h, MZ2h, MR2h** з **T4** |
| --- | --- |
| 2 | Передати **MXh, Bh, Zh, MZh, MRh** в **T8** |
| 3 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 4 | **Обчислення 2 [**a\_6= min(Ah)] |
| 5 | Прийняти **a** з **T8** |
| 6 | **Обчислення 3 [**a= min(a, a\_6)] |
| 7 | Передати **а** в **T4** |

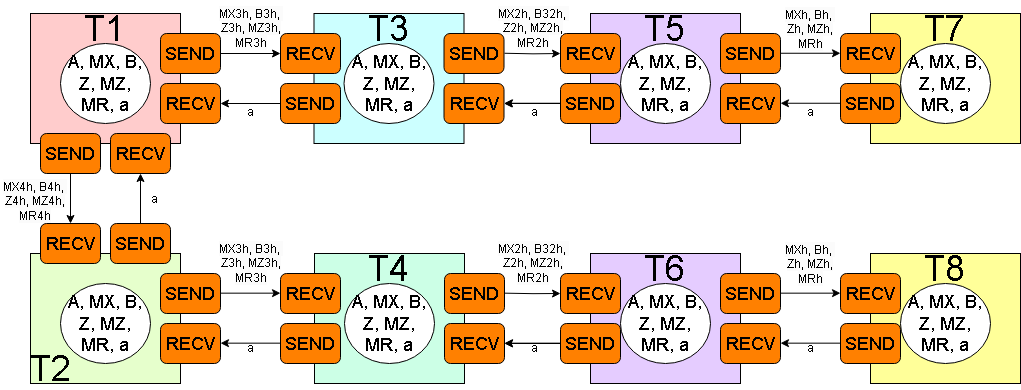
Задача T7

| 1 | Прийняти **MXh, Bh, Zh, MZh, MRh** з **T5** |
| --- | --- |
| 2 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 3 | **Обчислення 2 [**a\_7= min(Ah)] |
| 4 | Передати **а\_7** в **T5** |

Задача T8

| 1 | Прийняти **MXh, Bh, Zh, MZh, MRh** з **T6** |
| --- | --- |
| 2 | **Обчислення 1 [**Ah= B\*MZh + Z\*(MR\*MXh)] |
| 3 | **Обчислення 2 [**a\_8= min(Ah)] |
| 4 | Передати **а\_8** в **T6** |

**[Етап 3] Розроблення структурної схеми взаємодії задач:**

****

**Призначення взаємодії потоків та захисту спільних ресурсів:**

SEND – метод для відправлення повідомлення наступному потоку;

RECV – отримання повідомлення;

**Структура графу (комунікатора):**

ПРОЦЕСС== СУСІДИ

0 == 1, 2, 3

1 == 0, 3

2 == 0, 4

3 == 1, 5

4 == 2, 6

5 == 3, 7

6 == 4

7 == 5

Вершин = 8

Індекси = 3, 5, 7, 9, 11, 13, 14, 15

Ребра = 1, 2, 3, 0, 3, 0, 4, 1, 5, 2, 6, 3, 7, 4 ,5

**[Етап 4] Розробка програми:**

**Лістинг Lab.cpp**

/\*\*

\* Паралельне програмування

\* Лабораторна робота: №5

\* Оцінка: A

\* Варіант: 9

\* Завдання: a = min(B\*MZ + Z\*(MR\*MX))

\* • ПВВ1 – a, MX, B, Z , MZ, MR

\* • ПВВ2 – -

\* • ПВВ3 – -

\* • ПВВ4 – -

\* Виконав: Гапій Денис Едуардович ІП-05

\* Дата: 07.05.2023

\*\*/

/\*\*

\* Паралельне програмування

\* Лабораторна робота: №5

\* Оцінка: A

\* Варіант: 9

\* Завдання: a = min(B\*MZ + Z\*(MR\*MX))

\* • ПВВ1 – a, MX, B, Z , MZ, MR

\* • ПВВ2 – -

\* • ПВВ3 – -

\* • ПВВ4 – -

\* Виконав: Гапій Денис Едуардович ІП-05

\* Дата: 07.05.2023

\*\*/

using namespace std;

#pragma region Includes

#include <iostream>

#include <string>

#include <chrono>

#include "mpi.h"

#include "matrix.h"

#pragma endregion

#pragma region Constants

const int N = 16;

const int P = 8;

const int H = N / P;

const int DEFAULT = 1;

#pragma endregion

struct arrH

{

int arr[H];

};

struct arrH calculation1(int Bh[], int Zh[], Matrix <int> MXh, Matrix <int> MZh, Matrix <int> MRh)

{

arrH Ah;

for (int i = 0; i < H; i++) {

int part1 = 0;

int part2 = 0;

for (int j = 0; j < N; j++) {

part1 += Bh[i] \* MZh(i, j);

for (int k = 0; k < N; k++) {

part2 += Zh[i] \* MRh(i, j) \* MXh(i, j);

}

}

Ah.arr[i] = part1 + part2;

}

return Ah;

}

int calculation2(int Ah[]) {

int min = INT\_MAX;

for (int i = 0; i < H; i++)

if (Ah[i] < min)

min = Ah[i];

return min;

}

int calculation3(int a1, int a2) {

return a1 < a2 ? a1 : a2;

}

int main(int argc, char\* argv[])

{

// Declare new datatype for matrix

MPI\_Datatype row\_type;

// Initialize the MPI environment

MPI\_Init(&argc, &argv);

//Create row\_type

MPI\_Type\_contiguous(N, MPI\_INT, &row\_type);

MPI\_Type\_commit(&row\_type);

int a;

arrH A;

int B[N];

int Z[N];

Matrix <int> MX = Matrix <int>(N, N);

Matrix <int> MZ = Matrix <int>(N, N);

Matrix <int> MR = Matrix <int>(N, N);

MPI\_Comm graph\_comm;

int nnodes = 8; /\* number of nodes \*/

int index[8] = { 3, 5, 7, 9, 11, 13, 14, 15 }; /\* index definition \*/

int edges[15] = { 1, 2, 3, 0, 3, 0, 4, 1, 5, 2, 6, 3, 7, 4 ,5 }; /\* edges definition \*/

int reorder = 1; /\* allows processes reordered for efficiency \*/

MPI\_Graph\_create(MPI\_COMM\_WORLD, nnodes, index, edges, reorder, &graph\_comm);

// Get the rank of the calling process

int t\_id;

MPI\_Comm\_rank(graph\_comm, &t\_id);

if (t\_id == 0) { // T1

printf("Lab started\n");

printf("T1 start\n");

auto start = chrono::high\_resolution\_clock::now();

//Введення MX, B, Z, MZ, MR

for (int i = 0; i < N; i++) {

for (int j = 0; j < N; j++) {

MX(i,j) = DEFAULT;

MZ(i,j) = DEFAULT;

MR(i,j) = DEFAULT;

}

B[i] = DEFAULT;

Z[i] = DEFAULT;

}

//Передати MX4h, B4h, Z4h, MZ4h, MR4h в T2

MPI\_Send(B + H, H \* 4, MPI\_INT, 1, 0, graph\_comm);

MPI\_Send(Z + H, H \* 4, MPI\_INT, 1, 0, graph\_comm);

MPI\_Send(&MX(H, 0), H \* 4 \* N, MPI\_INT, 1, 0, graph\_comm);

MPI\_Send(&MZ(H, 0), H \* 4 \* N, MPI\_INT, 1, 0, graph\_comm);

MPI\_Send(&MR(H, 0), H \* 4 \* N, MPI\_INT, 1, 0, graph\_comm);

////Передати MX3h, B3h, Z3h, MZ3h, MR3h в T3

MPI\_Send(B + 5 \* H, H \* 3, MPI\_INT, 2, 1, graph\_comm);

MPI\_Send(Z + 5 \* H, H \* 3, MPI\_INT, 2, 1, graph\_comm);

MPI\_Send(&MX(5 \* H, 0), H \* 3 \* N, MPI\_INT, 2, 1, graph\_comm);

MPI\_Send(&MZ(5 \* H, 0), H \* 3 \* N, MPI\_INT, 2, 1, graph\_comm);

MPI\_Send(&MR(5 \* H, 0), H \* 3 \* N, MPI\_INT, 2, 1, graph\_comm);

//Обчислення 1[Ah = Bh \* MZh + Zh \* (MRh \* MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2[a\_1 = min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T2

int a2;

MPI\_Recv(&a2, 1, MPI\_INT, 1, 12, graph\_comm, MPI\_STATUS\_IGNORE);

//Прийняти a з T3

int a3;

MPI\_Recv(&a3, 1, MPI\_INT, 2, 13, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3[a = min(a\_1, a(2), a(3))]

a = calculation3(a, a2);

a = calculation3(a, a3);

//Виведення a

printf("Answer is: %d\n", a);

printf("T1 end\n");

auto stop = chrono::high\_resolution\_clock::now();

auto duration = chrono::duration\_cast<chrono::milliseconds>(stop - start);

printf("Lab finished in %d ms\n", duration.count());

}

else if (t\_id == 1) { // T2

printf("T2 start\n");

//Прийняти MX4h, B4h, Z4h, MZ4h, MR4h з T1

int size\_h = H \* 4;

MPI\_Recv(B, size\_h, MPI\_INT, 0, 0, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 0, 0, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h \* N, MPI\_INT, 0, 0, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h \* N, MPI\_INT, 0, 0, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h \* N, MPI\_INT, 0, 0, graph\_comm, MPI\_STATUS\_IGNORE);

//Передати MX3h, B3h, Z3h, MZ3h, MR3h в T4

MPI\_Send(B + H, H \* 3, MPI\_INT, 3, 2, graph\_comm);

MPI\_Send(Z + H, H \* 3, MPI\_INT, 3, 2, graph\_comm);

MPI\_Send(&MX(H, 0), H \* 3 \* N, MPI\_INT, 3, 2, graph\_comm);

MPI\_Send(&MZ(H, 0), H \* 3 \* N, MPI\_INT, 3, 2, graph\_comm);

MPI\_Send(&MR(H, 0), H \* 3 \* N, MPI\_INT, 3, 2, graph\_comm);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_2= min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T4

int a4;

MPI\_Recv(&a4, 1, MPI\_INT, 3, 3, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3 [a= min(a, a\_2)]

a = calculation3(a, a4);

//Передати а в T1

MPI\_Send(&a, 1, MPI\_INT, 0, 12, graph\_comm);

printf("T2 end\n");

}

else if (t\_id == 2) { // T3

printf("T3 start\n");

//Прийняти MX3h, B3h, Z3h, MZ3h, MR3h з T1

int size\_h = H \* 3;

MPI\_Recv(B, size\_h, MPI\_INT, 0, 1, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 0, 1, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h \* N, MPI\_INT, 0, 1, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h \* N, MPI\_INT, 0, 1, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h \* N, MPI\_INT, 0, 1, graph\_comm, MPI\_STATUS\_IGNORE);

//Передати MX2h, B2h, Z2h, MZ2h, MR2h в T5

MPI\_Send(B + H, H \* 2, MPI\_INT, 4, 4, graph\_comm);

MPI\_Send(Z + H, H \* 2, MPI\_INT, 4, 4, graph\_comm);

MPI\_Send(&MX(H, 0), H \* 2 \* N, MPI\_INT, 4, 4, graph\_comm);

MPI\_Send(&MZ(H, 0), H \* 2 \* N, MPI\_INT, 4, 4, graph\_comm);

MPI\_Send(&MR(H, 0), H \* 2 \* N, MPI\_INT, 4, 4, graph\_comm);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_3= min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T5

int a5;

MPI\_Recv(&a5, 1, MPI\_INT, 4, 5, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3 [a= min(a, a\_3)]

a = calculation3(a, a5);

//Передати а в T1

MPI\_Send(&a, 1, MPI\_INT, 0, 13, graph\_comm);

printf("T3 end\n");

}

else if (t\_id == 3) { // T4

printf("T4 start\n");

//Прийняти MX3h, B3h, Z3h, MZ3h, MR3h з T2

int size\_h = H \* 3;

MPI\_Recv(B, size\_h, MPI\_INT, 1, 2, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 1, 2, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h\* N, MPI\_INT, 1, 2, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h\* N, MPI\_INT, 1, 2, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h\* N, MPI\_INT, 1, 2, graph\_comm, MPI\_STATUS\_IGNORE);

//Передати MX2h, B2h, Z2h, MZ2h, MR2h в T6

MPI\_Send(B + H, H \* 2, MPI\_INT, 5, 6, graph\_comm);

MPI\_Send(Z + H, H \* 2, MPI\_INT, 5, 6, graph\_comm);

MPI\_Send(&MX(H, 0), H \* 2 \* N, MPI\_INT, 5, 6, graph\_comm);

MPI\_Send(&MZ(H, 0), H \* 2 \* N, MPI\_INT, 5, 6, graph\_comm);

MPI\_Send(&MR(H, 0), H \* 2 \* N, MPI\_INT, 5, 6, graph\_comm);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_4= min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T6

int a6;

MPI\_Recv(&a6, 1, MPI\_INT, 5, 9, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3 [a= min(a, a\_4)]

a = calculation3(a, a6);

//Передати а в T2

MPI\_Send(&a, 1, MPI\_INT, 1, 3, graph\_comm);

printf("T4 end\n");

}

else if (t\_id == 4) { // T5

printf("T5 start\n");

//Прийняти MX2h, B2h, Z2h, MZ2h, MR2h з T3

int size\_h = H \* 2;

MPI\_Recv(B, size\_h, MPI\_INT, 2, 4, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 2, 4, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h\* N, MPI\_INT, 2, 4, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h\* N, MPI\_INT, 2, 4, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h\* N, MPI\_INT, 2, 4, graph\_comm, MPI\_STATUS\_IGNORE);

//Передати MXh, Bh, Zh, MZh, MRh в T7

MPI\_Send(B + H, H, MPI\_INT, 6, 7, graph\_comm);

MPI\_Send(Z + H, H, MPI\_INT, 6, 7, graph\_comm);

MPI\_Send(&MX(H, 0), H\* N, MPI\_INT, 6, 7, graph\_comm);

MPI\_Send(&MZ(H, 0), H\* N, MPI\_INT, 6, 7, graph\_comm);

MPI\_Send(&MR(H, 0), H\* N, MPI\_INT, 6, 7, graph\_comm);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_5= min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T7

int a7;

MPI\_Recv(&a7, 1, MPI\_INT, 6, 8, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3 [a= min(a, a\_5)]

int a\_mins[2]{ a, a7 };

a = calculation3(a, a7);

//Передати а в T3

MPI\_Send(&a, 1, MPI\_INT, 2, 5, graph\_comm);

printf("T5 end\n");

}

else if (t\_id == 5) { // T6

printf("T6 start\n");

//Прийняти MX2h, B2h, Z2h, MZ2h, MR2h з T4

int size\_h = H \* 2;

MPI\_Recv(B, size\_h, MPI\_INT, 3, 6, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 3, 6, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h\* N, MPI\_INT, 3, 6, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h\* N, MPI\_INT, 3, 6, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h\* N, MPI\_INT, 3, 6, graph\_comm, MPI\_STATUS\_IGNORE);

//Передати MXh, Bh, Zh, MZh, MRh в T8

MPI\_Send(B + H, H, MPI\_INT, 7, 10, graph\_comm);

MPI\_Send(Z + H, H, MPI\_INT, 7, 10, graph\_comm);

MPI\_Send(&MX(H, 0), H\* N, MPI\_INT, 7, 10, graph\_comm);

MPI\_Send(&MZ(H, 0), H\* N, MPI\_INT, 7, 10, graph\_comm);

MPI\_Send(&MR(H, 0), H\* N, MPI\_INT, 7, 10, graph\_comm);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_6= min(Ah)]

a = calculation2(A.arr);

//Прийняти a з T8

int a8;

MPI\_Recv(&a8, 1, MPI\_INT, 7, 11, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 3 [a= min(a, a\_6)]

a = calculation3(a, a8);

//Передати а в T4

MPI\_Send(&a, 1, MPI\_INT, 3, 9, graph\_comm);

printf("T6 end\n");

}

else if (t\_id == 6) { // T7

printf("T7 start\n");

//Прийняти MXh, Bh, Zh, MZh, MRh з T5

int size\_h = H;

MPI\_Recv(B, size\_h, MPI\_INT, 4, 7, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 4, 7, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0, 0), size\_h\* N, MPI\_INT, 4, 7, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0, 0), size\_h\* N, MPI\_INT, 4, 7, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0, 0), size\_h\* N, MPI\_INT, 4, 7, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_7= min(Ah)]

a = calculation2(A.arr);

//Передати а в T5

MPI\_Send(&a, 1, MPI\_INT, 4, 8, graph\_comm);

printf("T7 end\n");

}

else if (t\_id == 7) { // T8

printf("T8 start\n");

//Прийняти MXh, Bh, Zh, MZh, MRh з T6

int size\_h = H;

MPI\_Recv(B, size\_h, MPI\_INT, 5, 10, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(Z, size\_h, MPI\_INT, 5, 10, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MX(0,0), size\_h\*N, MPI\_INT, 5, 10, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MZ(0,0), size\_h\*N, MPI\_INT, 5, 10, graph\_comm, MPI\_STATUS\_IGNORE);

MPI\_Recv(&MR(0,0), size\_h\*N, MPI\_INT, 5, 10, graph\_comm, MPI\_STATUS\_IGNORE);

//Обчислення 1 [Ah= Bh\*MZh + Zh\*(MRh\*MXh)]

A = calculation1(B, Z, MX, MZ, MR);

//Обчислення 2 [a\_8= min(Ah)]

a = calculation2(A.arr);

//Передати а в T6

MPI\_Send(&a, 1, MPI\_INT, 5, 11, graph\_comm);

printf("T8 end\n");

}

MPI\_Type\_free(&row\_type);

// Finalize: Any resources allocated for MPI can be freed

MPI\_Finalize();

}

**Лістинг matrix.h**

#pragma once

#ifndef MATRIX\_H\_

#define MATRIX\_H\_

#include <vector>

#include <complex>

using std::vector;

using std::complex;

// simple wrapper of a vector

template <class T>

class Matrix {

public:

    // create an empty matrix

    Matrix(int numrows, int numcols)

        :Nrow(numrows), Ncol(numcols), elements(Nrow\* Ncol) {}

    // construct it from existing data

    Matrix(int numrows, int numcols, T\* data)

        :Nrow(numrows), Ncol(numcols), elements(data, data + numrows \* numcols) {}

    int rows() { return Nrow; }

    int cols() { return Ncol; }

    // access to elements, col is the fast axis

    T operator() (int row, int col) const { return elements[Ncol \* row + col]; }

    T& operator() (int row, int col) { return elements[Ncol \* row + col]; }

    // get raw pointer to elements[0]

    T\* data() { return elements.data(); }

    const vector<T>& elem() { return elements; }

private:

    int Nrow, Ncol;

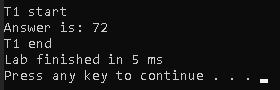
    vector<T> elements;

};

#endif // MATRIX\_H\_

**[Етап 5] Тестування програми:**

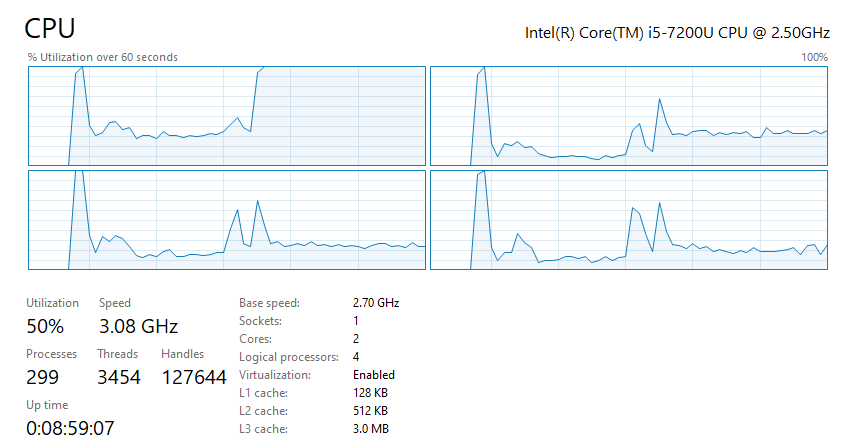
**Результат роботи програми при N=8:**

****

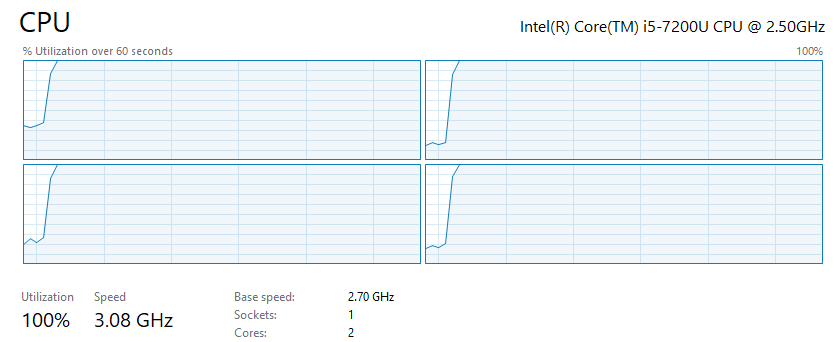
**Коефіцієнт прискорення:**

| **N** | **T8** | **T1** | **Кп = T1/T4** |
| --- | --- | --- | --- |
| **160** |  |  | **1.85** |
| **800** |  |  | **2.46** |
| **1600** |  |  | **3.44** |

**Навантаження ядер для T1:**

****

**Навантаження ядер для T1-T8:**

****

Апаратні характеристики можна спостерігати на зображені вище. ОС: Windows. X64

**Висновок:**

- Виконано розробку паралельного алгоритму задачі, яка потребує використання паралельних алгоритмів для множення матриць між собою, скаляру на матричну велечину та пошук мінімального значення у векторі. Визначено перелік спільних ресурсів, для яких потрібно вирішити завдання взаємного виключення.

- Проведено розробку алгоритмів потоків Т1-Т8 для паралельно розподіленого обчислення своєї частини результату рівняння. Окрім взаємних виключень та обчислень, було передбачено синхронізацію для вводу / виводу значень.

- Розроблено схему роботи усіх восьми потоків. Визначено засоби організації взаємодії потоків, а саме – надсилання повідомлень та їх отримання для синхронізації потоків.

- Виконано розробку програми мовою C++ та за допомоги функціоналу бібліотеки MPI (використано версію від Microsoft). Для створення потоків використано прапорець `-n` з аргументом 8 при запуску mpiexec.exe з командного рядка. Кожний потік отримав власний ідентифікатор і має власний алгоритм процесу описаний у Етапі 2.

- Під час розробки програми виникла проблема функціонування програми при N >= 280. Відлагодження показали, що програма падає через двовимірний масивв для передавання реалізації матриці, проте остаточну причину не знайдено, тому натомість двохвимірного масив було створено та використано векторну структуру (matrix.h).

- З тестування можемо винести, що:

* Програмні результати співпали з обчисленнями у онлайн калькуляторі (WolframApha) 64 + 8 = 72.
* При відключені інших ядер, потоки продовжують працювати через єдиний логічний процесор. І скомпільований с++ застосунок доводить обчислення до кінця, але за більшу кількість часу.
* Підраховано коефіцієнт прискорення для N : 160, 800, 1600. Для аналізу витраченого часу виділено час початку та закінчення роботи усього застосунку за допомогою функції бібліотеки `chrono`: chrono::high\_resolution\_clock::now().