



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

ЛАБОРАТОРНА РОБОТА №2
з дисципліни «Паралельні та розподілені обчислення
Паралельне програмування-2»
Тема: «WinAPI. Семафори, мютекси, події, критичні секції»

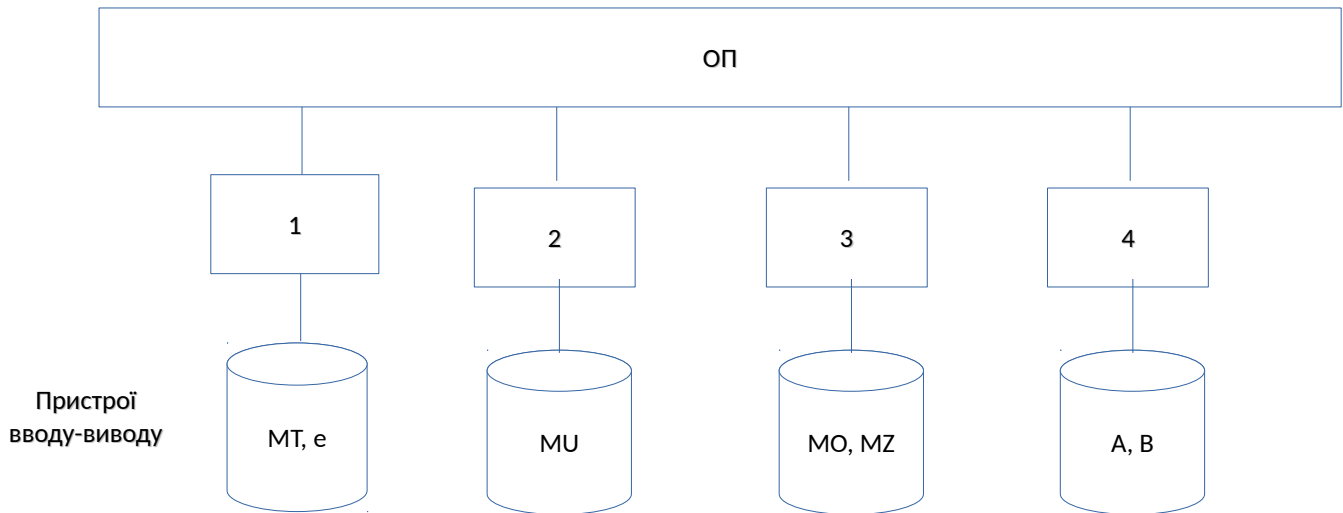
Виконав:
студент 3-го курсу
групи ІП-42
з номер заліковки 4206
Дзюба Влад

Завдання

Мета роботи: розробка програми для ПКС зі СП

Мова програмування: C++

Засоби організації взаємодії процесів: семафори, мютекси, події, критичні секції
бібліотеки Win32



Варіант

$$A = (B \cdot MO) (MZ \cdot MT + e \cdot MU)$$

Математичний паралельний алгоритм:

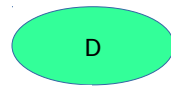
1. $D_h = B \cdot MO_h$
 2. $A_h = D \cdot (MZ \cdot MT_h + e \cdot MU_h)$
- CP: B, MZ, e, D.

Алгоритм для кожного процесу:

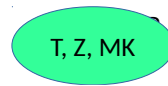
T_1 : 1. Ввести MT, e . 2. Чекати сигнал про завершення вводу з T_2 . 3. Чекати сигнал про завершення вводу з T_3 . 4. Чекати сигнал про завершення вводу з T_4 . 5. Сигнал про початок обрахунку D до T_2 . 6. Сигнал про початок обрахунку D до T_3 . 7. Сигнал про початок обрахунку D до T_4 . 8. Копіювати $B_1 = B$. 9. Обрахувати $D_h = B_1 \cdot MO_h$. 10. Чекати сигнал про завершення обрахунку D_h з T_2 . 11. Чекати сигнал про завершення обрахунку D_h з T_3 . 12. Чекати сигнал про завершення обрахунку D_h з T_4 . 13. Сигнал про початок обрахунку A до T_2 . 14. Сигнал про початок обрахунку A до T_3 . 15. Сигнал про початок обрахунку A до T_4 . 16. Копіювати $D_1 = D, MZ_1 = MZ, e_1 = e$. 17. Обрахувати $A_h = D_1 \cdot (MZ_1 \cdot MT_h + e_1 \cdot MU_h)$. 18. Чекати сигнал про завершення обрахунку A_h з T_2 . 19. Чекати сигнал про завершення обрахунку з A_h T_3 . 20. Чекати сигнал про завершення обрахунку з A_h T_4 . 21. Вивести A .	 $W_{2,1}$ $W_{3,1}$ $W_{4,1}$ $S_{2,1}$ $S_{3,1}$ $S_{4,1}$ КУ1 $W_{2,2}$ $W_{3,2}$ $W_{4,2}$ $S_{2,2}$ $S_{3,2}$ $S_{4,2}$ КУ2 $W_{2,3}$ $W_{3,3}$ $W_{4,3}$	T_2 : 1. Ввести MU . 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку D від T_1 . 4. Копіювати $B_2 = B$. 5. Обрахувати $D_h = B_2 \cdot MO_h$. 6. Сигнал про завершення обрахунку D_h до T_1 . 7. Чекати сигнал про початок обрахунку A від T_1 . 8. Копіювати $D_2 = D, MZ_2 = MZ, e_2 = e$. 9. Обрахувати $A_h = D_2 \cdot (MZ_2 \cdot MT_h + e_2 \cdot MU_h)$. 10. Сигнал про завершення обрахунку A_h до T_1 .	 $S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$
---	---	---	---

T_3 : <ol style="list-style-type: none"> 1. Ввести MU. 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку D від T_1 . 4. Копіювати $B_3=B$. 5. Обрахувати $D_h=B_3 \cdot MO_h$. 6. Сигнал про завершення обрахунку D_h до T_1 . 7. Чекати сигнал про початок обрахунку A від T_1 . 8. Копіювати $D_3=D, MZ_3=MZ, e_3=e$. 9. Обрахувати $A_h=D_3 \cdot (MZ_3 \cdot MT_h + e_3 \cdot MU_h)$ 10. Сигнал про завершення обрахунку A_h до T_1 . 	$S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$	T_4 : <ol style="list-style-type: none"> 1. Ввести MU. 2. Сигнал про завершення вводу до T_1 . 3. Чекати сигнал про початок обрахунку D від T_1 . 4. Копіювати $B_4=B$. 5. Обрахувати $D_h=B_4 \cdot MO_h$. 6. Сигнал про завершення обрахунку D_h до T_1 . 7. Чекати сигнал про початок обрахунку A від T_1 . 8. Копіювати $D_4=D, MZ_4=MZ, e_4=e$. 9. Обрахувати $A_h=D_4 \cdot (MZ_4 \cdot MT_h + e_4 \cdot MU_h)$. 10. Сигнал про завершення обрахунку A_h до T_1 . 	$S_{1,1}$ $W_{1,1}$ КУ1 $S_{1,2}$ $W_{1,2}$ КУ2 $S_{1,3}$
--	---	--	---

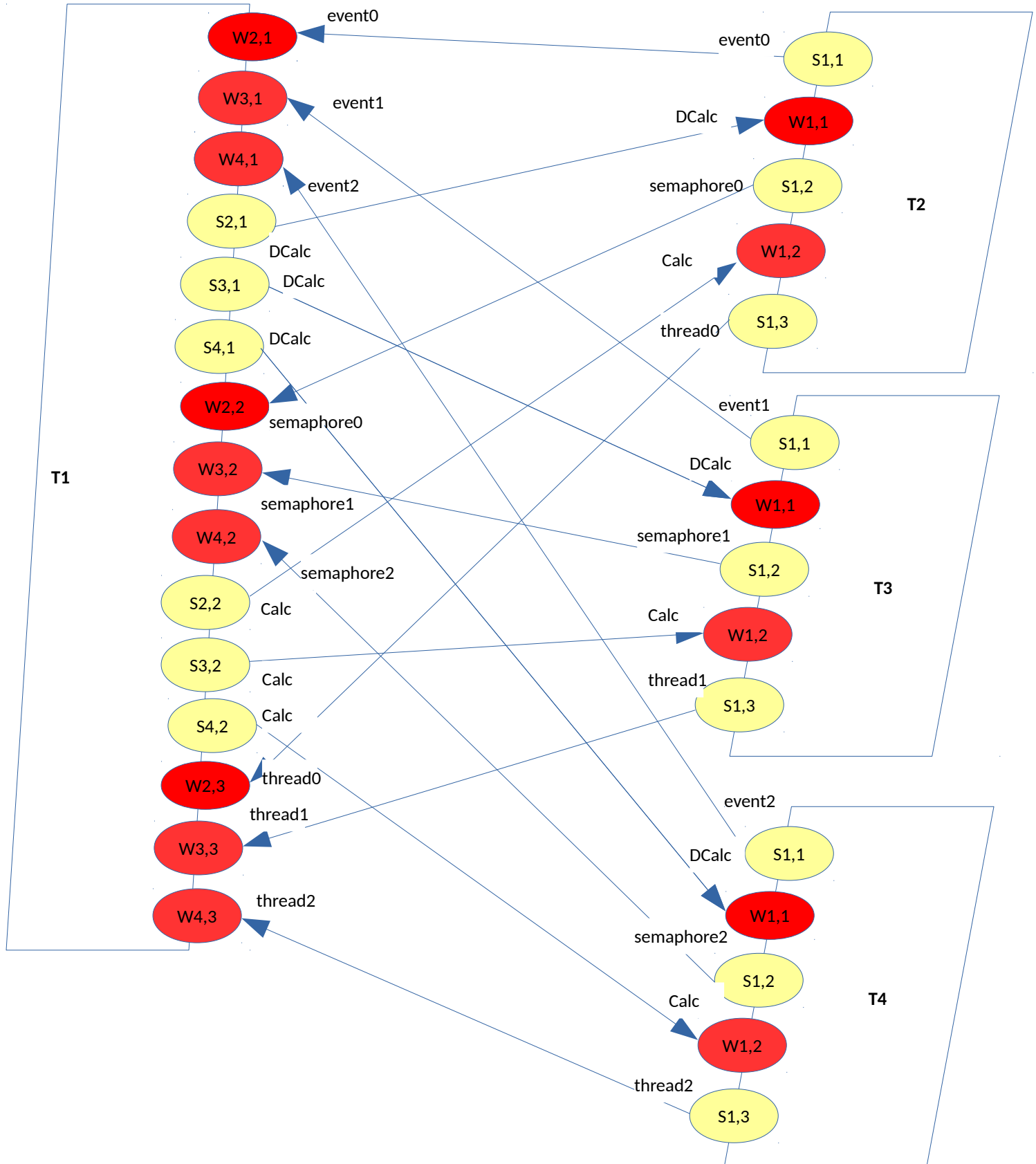
Структурна схема взаємодії процесів



CriticalSection



ghMutex



Лістинг програми:

main.cpp

```
/*
// main
// Author:
//     Dzyuba Vlad, IP-42
*/
#include <windows.h>
#include <iostream>

#define THREADCOUNT 4
#define N 4000
#define H N/4

void T1Read();
void T2Read();
void T3Read();
void T4Read();
void T1DCalc();
void T2DCalc();
void T3DCalc();
void T4DCalc();
void T1Calc();
void T2Calc();
void T3Calc();
void T4Calc();
DWORD WINAPI T1Proc( LPVOID lpParam );
DWORD WINAPI T2Proc( LPVOID lpParam );
DWORD WINAPI T3Proc( LPVOID lpParam );
DWORD WINAPI T4Proc( LPVOID lpParam );

// input
int *MT[N], e, *MU[N], *MO[N], *MZ[N], B[N];
//output
int A[N];
//intermediate
int D[N];

HANDLE events[THREADCOUNT-1];
HANDLE semaphores[THREADCOUNT-1];
CRITICAL_SECTION CriticalSection;
HANDLE ghMutex;
HANDLE DCalc, Calc;

using namespace std;

int main()
{
    InitializeCriticalSectionAndSpinCount(&CriticalSection,
0x000000400);
    for (int i = 0; i < N; i++)
    {
        MT[i] = new int[N];
```

```

        MU[i] = new int[N];
        MO[i] = new int[N];
        MZ[i] = new int[N];
    }
    HANDLE threads[THREADCOUNT-1];
    DWORD ThreadId;

    DCalc = CreateEvent(NULL, TRUE, FALSE, NULL);
    Calc = CreateEvent(NULL, TRUE, FALSE, NULL);
    ghMutex = CreateMutex(NULL, FALSE, NULL);
    for (int i = 0; i < THREADCOUNT-1; i++)
    {
        events[i] = CreateEvent(NULL, TRUE, FALSE, NULL);
        semaphores[i] = CreateSemaphore(NULL, 0, 1, NULL);
    }
    threads[0] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)
T2Proc, NULL, 0, &ThreadId);
    threads[1] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)
T3Proc, NULL, 0, &ThreadId);
    threads[2] = CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE)
T4Proc, NULL, 0, &ThreadId);

    T1Read();
    WaitForMultipleObjects(THREADCOUNT - 1, events, TRUE,
INFINITE);

    SetEvent(DCalc);
    T1DCalc();
    WaitForMultipleObjects(THREADCOUNT-1, semaphores, TRUE,
INFINITE);

    SetEvent(Calc);
    T1Calc();
    WaitForMultipleObjects(THREADCOUNT-1, threads, TRUE,
INFINITE);

    if (N <= 20)
    {
        for (int i = 0; i < N; i++)
        {
            cout << A[i] << " ";
        }
        cout << endl;
    }

    DeleteCriticalSection(&CriticalSection);
    for (int i = 0; i < THREADCOUNT-1; i++)
    {
        CloseHandle(threads[i]);
        CloseHandle(events[i]);
        CloseHandle(semaphores[i]);
    }
}

```

```

void T1Calc()
{
    int **MZ1, *D1, e1;
    WaitForSingleObject(ghMutex, INFINITE);
    MZ1 = MZ;
    D1 = D;
    e1 = e;
    ReleaseMutex(ghMutex);

    for (int i = 0; i < H; i++)
    {
        A[i] = 0;
        for (int j = 0; j < N; j++)
        {
            int zt = e1 * MU[j][i];
            for (int k = 0; k < N; k++)
            {
                zt += MZ1[j][k] * MT[k][i];
            }
            A[i] += zt * D1[j];
        }
    }
}

```

```

void T2Calc()
{
    int **MZ2, *D2, e2;
    WaitForSingleObject(ghMutex, INFINITE);
    MZ2 = MZ;
    D2 = D;
    e2 = e;
    ReleaseMutex(ghMutex);

    for (int i = H; i < 2 * H; i++)
    {
        A[i] = 0;
        for (int j = 0; j < N; j++)
        {
            int zt = e2 * MU[j][i];
            for (int k = 0; k < N; k++)
            {
                zt += MZ2[j][k] * MT[k][i];
            }
            A[i] += zt * D2[j];
        }
    }
}

```

```

void T3Calc()
{
    int **MZ3, *D3, e3;
    WaitForSingleObject(ghMutex, INFINITE);
    MZ3 = MZ;
    D3 = D;

```



```

    e3 = e;
    ReleaseMutex(ghMutex);

    for (int i = 2 * H; i < 3 * H; i++)
    {
        A[i] = 0;
        for (int j = 0; j < N; j++)
        {
            int zt = e3 * MU[j][i];
            for (int k = 0; k < N; k++)
            {
                zt += MZ3[j][k] * MT[k][i];
            }
            A[i] += zt * D3[j];
        }
    }
}

void T4Calc()
{
    int **MZ4, *D4, e4;
    EnterCriticalSection(&CriticalSection);
    MZ4 = MZ;
    D4 = D;
    e4 = e;
    LeaveCriticalSection(&CriticalSection);

    for (int i = 3 * H; i < 4 * H; i++)
    {
        A[i] = 0;
        for (int j = 0; j < N; j++)
        {
            int zt = e4 * MU[j][i];
            for (int k = 0; k < N; k++)
            {
                zt += MZ4[j][k] * MT[k][i];
            }
            A[i] += zt * D4[j];
        }
    }
}

void T1DCalc()
{
    int *B1;
    EnterCriticalSection(&CriticalSection);
    B1 = B;
    LeaveCriticalSection(&CriticalSection);
    for (int i = 0; i < H; i++)
    {
        D[i] = 0;
        for (int j = 0; j < N; j++)
        {
            D[i] += B1[j] * M0[j][i];
        }
    }
}

```

```

    }
}

void T2DCalc()
{
    int *B2;
    EnterCriticalSection(&CriticalSection);
    B2 = B;
    LeaveCriticalSection(&CriticalSection);
    for (int i = H; i < 2 * H; i++)
    {
        D[i] = 0;
        for (int j = 0; j < N; j++)
        {
            D[i] += B2[j] * M0[j][i];
        }
    }
    ReleaseSemaphore(semaphores[0], 1, NULL);
}

void T3DCalc()
{
    int *B3;
    EnterCriticalSection(&CriticalSection);
    B3 = B;
    LeaveCriticalSection(&CriticalSection);
    for (int i = 2 * H; i < 3 * H; i++)
    {
        D[i] = 0;
        for (int j = 0; j < N; j++)
        {
            D[i] += B3[j] * M0[j][i];
        }
    }
    ReleaseSemaphore(semaphores[1], 1, NULL);
}

void T4DCalc()
{
    int *B4;
    EnterCriticalSection(&CriticalSection);
    B4 = B;
    LeaveCriticalSection(&CriticalSection);
    for (int i = 3 * H; i < 4 * H; i++)
    {
        D[i] = 0;
        for (int j = 0; j < N; j++)
        {
            D[i] += B4[j] * M0[j][i];
        }
    }
    ReleaseSemaphore(semaphores[2], 1, NULL);
}

```

```

void T1Read()
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            MT[i][j] = 1;
        }
    }
    e = 1;
}

void T2Read()
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            MU[i][j] = 1;
        }
    }
    SetEvent(events[0]);
}

void T3Read()
{
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            MO[i][j] = 1;
        }
    }
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            MZ[i][j] = 1;
        }
    }
    SetEvent(events[1]);
}

void T4Read()
{
    for (int i = 0; i < N; i++) {
        B[i] = 1;
    }
    SetEvent(events[2]);
}

DWORD WINAPI T2Proc( LPVOID lpParam )
{
    T2Read();

    WaitForSingleObject(DCalc, INFINITE);
    T2DCalc();

    WaitForSingleObject(Calc, INFINITE);
    T2Calc();
}

```

```
DWORD WINAPI T3Proc( LPVOID lpParam )
{
    T3Read();

    WaitForSingleObject(DCalc, INFINITE);
    T3DCalc();

    WaitForSingleObject(Calc, INFINITE);
    T3Calc();
}

DWORD WINAPI T4Proc( LPVOID lpParam )
{
    T4Read();

    WaitForSingleObject(DCalc, INFINITE);
    T4DCalc();

    WaitForSingleObject(Calc, INFINITE);
    T4Calc();
}
```