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 №1

«Програмування потоків»

з дисципліни
«Програмне забезпечення високопродуктивних комп’ютерних систем»

Виконала:

студентка групи ІМ-11
Бащак Ярина Володимирівна

Перевірив:

доц. Корочкін О. В.

Київ 2024

Завдання.

Слід розробити паралельну програму за допомогою обраної мови Java, Ada, C# або бібліотеки паралельного програмування (WinAPI), яка забезпечує паралельне виконання трьох математичних функцій ***F1, F2, F3*** згідно отриманому варіанту ЛР1.

Стандартна структура програми для ЛР1:

- модуль (клас) ***Data*** , що містить ресурси для створення потоків (типи, допоміжні процедури та функції та інше…);
- потоки ***T1, T2, T3***.

Кожен потік

- має локальні змінні,
- здійснює дії, що необхідні для паралельного обчислення відповідної функції ***Fi***:
  - введення відповідних даних в потоці,
  - обчислення функції ***Fi,***
  - виведення результату виконання потоку Fi.

  Необхідні ресурси для побудови потоку беруться з модулю ***Data***;
- головну процедуру ***(Lab1),*** яка створює потоки і запускає їх на виконання.

Необхідне виконати налагодження паралельної програми та дослідити її виконання для малих та великих значень N. Розмір векторів та матриць обов'язково позначаються в програмі через змінну N.

Для невеликого значення N=4 введення в потоці векторів та матриць здійсняються за допомогою клавіатуру. При цьому всі елементи векторів та матриць отримують для потоку T1 значення 1, для T2 - значення 2, для T3 - значення 3. Дослідити і пояснити проблеми, яки з'являються в програмі при введення даних з клавіатури і виведення результатів на монітор, пояснити це в протоколі ЛР1.

Для великих значень N =1000 для введення даних передбачити створювання в модулі Data ресурсів (методів), що дозволяють реалізувати введення шляхом:

- формування файлів з наступним зчитування даних з них
- встановлення всіх елементів даних заданому значенню (наприклад 1, 2 або 3)
- використання генератору випадкових значень.

Під час виконання програми для N=1000 прослідити та проаналізувати процес завантаження програмою ядер багатоядерного процесора за допомогою Диспетчеру задач ОС Windows. Зробити в протоколі висновки, що до завантаження багатоядерного процесора потоками. Здійснити встановлення кількості працюючих ядер 3 – по кількості потоків (через відключення зайвих в Менеджері задач ОС).

Опис програми.

Головний клас Lab1.

- Містить змінну N, яка встановлюється на основі вводу користувача.
- Метод Main виконує запуск трьох потоків (t1, t2, t3) з однаковим розміром стека та нормальним пріоритетом. Кожен потік виконує відповідні потокові функції (Func1, Func2, Func3).
- Метод SetThreadAffinity, що дозволяє встановити прив'язку потоку до певного процесорного ядра.
- Func1, Func2, Func3 є статичними методами, кожен з яких ініціалізує дані, обчислює і записує результати, викликаючи відповідні методи класу Data.

Клас Data:

- Містить вектори та матриці, які потрібні потоками для обчислення.
- Методи ініціалізації даних (InitializeForFunc1, InitializeForFunc2, InitializeForFunc3), кожен з яких отримує ті дані (вектори і матриці), які потрібні відповідному потоку. Якщо значення N мале, ці дані запитуються через консоль. Якщо велике, то беруться з файлу, або заповнюються заданим числом, або випадковими числами. При цьому в консоль виводиться інформація про те, звідки дані для кожного з потоків були отримані.

  Окремо є приватний метод ReadValue, який виконує читання значення з консолі, що використовується для малих N. Якщо введене значення в консолі є коректним і його можна зчитати як число, тоді це число повертається. А якщо значення не коректне або не задане – встановлюється значення за замовчуванням для цього потоку (1, 2 або 3).

  Також були створені допоміжні приватні методи, які реалізують 3 способи задання векторів і матриць для великих N:

  1. LoadFromFile – зчитує дані з файлу і заповнює надані вектори і матриці;
  2. InitializeWithFixedValue – заповнює надані вектори і матриці заданим числом;
  3. InitializeWithRandomValues - заповнює надані вектори і матриці випадковими числами.
- Методи обчислень функцій F1, F2, F3, які виконують дії над векторами і матрицями відповідно до своєї формули та повертають результат.

  Також були створені окремі приватні методи, які власне виконують логіку дій над векторами і матрицями, такі як MultiplyMatrixVector, MultiplyMatrices, FindMax.

- Метод виводу OutputResults, який залежно від N виводить результати або в консоль, або записує в файл.

Формат запису вхідних даних у файл для великих N:

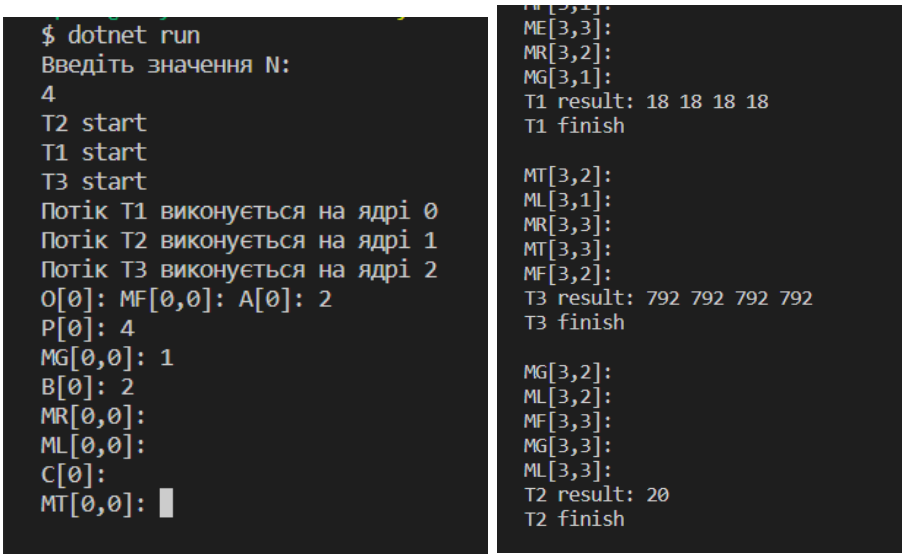
Щоб задані дані для певого потоку через файл, потрібно створити файл з назвою funcX_data.txt, де X – номер потоку, наприклад, func1_data.txt. Всередині потрібно задати значення спочатку всім векторам, а потім всім матрицям в алфавітному порядку. Всі значення записуються через кому. Перед записом даних кожного вектора і матриці потрібно пропустити один рядок, де для зручності можна написати коментар з назвою елементу, що задається.

Наприклад, ось приклад правильного формату файлу для потоку 2

// Вектор O
1,2,3,4,5
// Вектор P
6,7,8,9,10
// Матриця MR
1,2,3,4,5
6,7,8,9,10
11,12,13,14,15
16,17,18,19,20
21,22,23,24,25
// Матриця MT
26,27,28,29,30
31,32,33,34,35
36,37,38,39,40
41,42,43,44,45
46,47,48,49,50

Приклади роботи програми:

- N = 4



Мал. 1-2. Консоль початку і кінця виконання програми

Кожен потік запитує свої вхідні дані. Можна задати будь-які числа, але, як згадувалося раніше, якщо введене значення не може зчитатися як число – воно замінюється значенням за замовчуванням для цього потоку (1, 2 або 3). Тому для пришвидшення тестування програми можна пропускати введення певних (або всіх) значень – вони все одно будуть задані.

Коли потік отримує всі потрібні дані – починає обчислення і при завершенні виводить результат в консоль.

Тут є проблема на самому початку в тому, що при старті програми всі три потоки одночасно запитують свій перший елемент. Один із способів вирішення

цієї проблеми – синхронізація роботи потоків, що дозволить одному потоку чекати на сигнал від іншого потоку перед наступним запитом. Але це виходить за межі теми лабораторної роботи №1.

- N = 1000

```
$ dotnet run
Введіть значення N:
1000
T2 start
T3 start
T1 start
Потік T2 виконується на ядрі 1
Потік T3 виконується на ядрі 2
Потік T1 виконується на ядрі 0
File func3_data.txt not found, so data was set to fixed value 3
File func2_data.txt not found, so data was randomly provided
Data from func1_data.txt was read
T3 finish
T2 finish
T1 finish
```

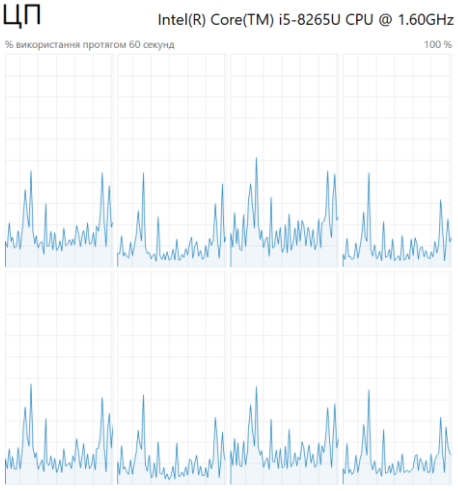
Мал. 3. Виконання програми при N = 1000

Тут продемонстровано всі 3 способи введення даних для великих значень N. Потік перевіряє чи є файл з назвою funcX_data.txt, якщо є, то бере дані з нього, якщо ні – задає всім елементам фіксоване або випадкове значення, про що повідомляє в консолі.

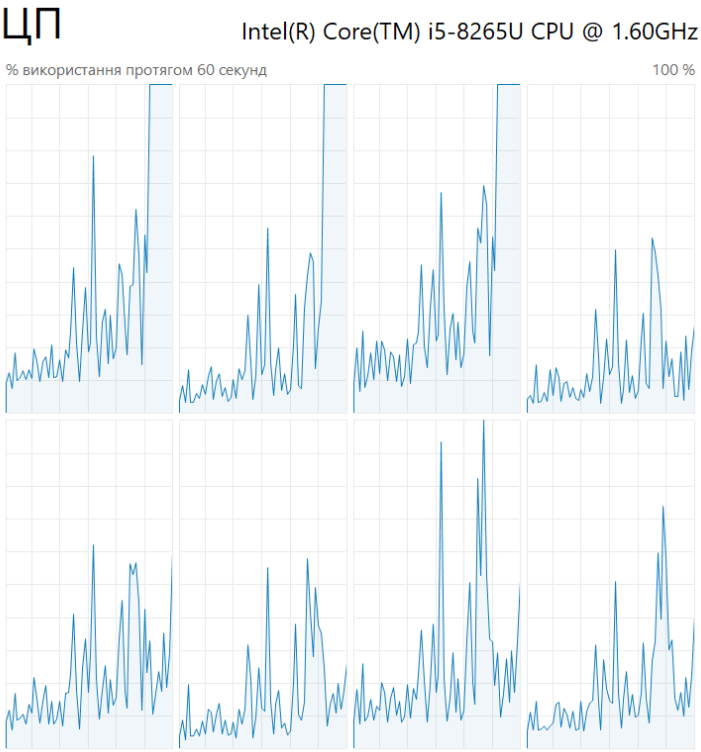
Результати обчилень кожного потоку записуються у відповідні файли з назвою results_funcX.txt. Наприклад, файл results_func1.txt може виглядати так:

T1 result: 1000002 1000003 1000004 1000005 1000006 1000002 1000003 1000004 1000005 1000006 1000002 1000003 1000004 1000005 1000006 1000002 1000003 1000004 1000005 1000006 1000002 1000003 1000004 …

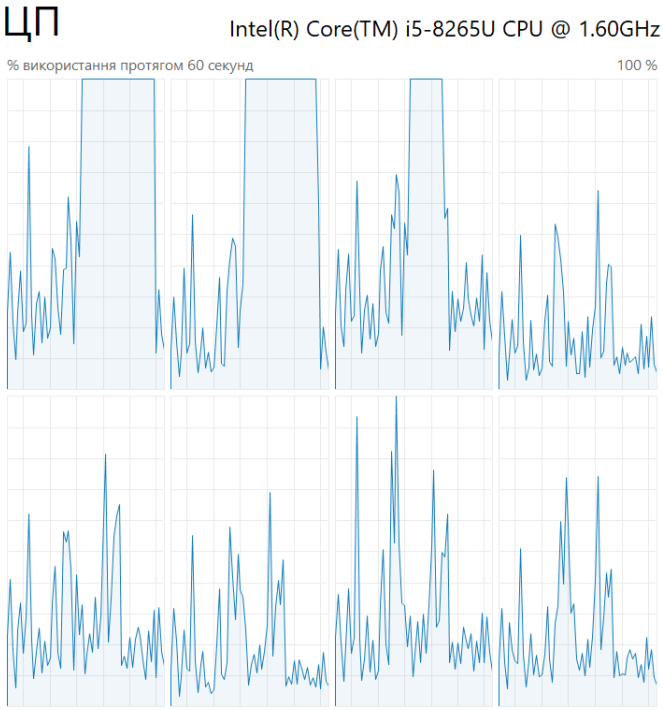
Процес завантаження ядер.



Мал. 4. Використання процесорів до запуску програми



Мал. 5. Використання процесорів під час роботи програми



Мал. 6. Використання процесорів після завершення роботи програми

При запуску програми з N = 1000 в «Диспетчері задач» чітко видно, що 3 процесори починають використовуватись на 100%. Отже, саме там проходять обчислення. До того ж в консолі потік №3 відпрацював помітно швидше, ніж 1 і 2, що також видно по графіках.

Висновки: програма з використанням паралельних потоків на мові C# була успішно розроблена, а всі результати у вигляді скріншотів представлено у звіті.

Під час виконання було виявлено, що у .NET Framework немає прямого способу встановити виконання потоку на конкретне ядро через обмеження платформи. Тому було знайдено рішення - застосувати WinAPI. Зокрема, було використано методи SetThreadAffinityMask та GetCurrentThread для прямого контролю над розподілом потоків по окремих ядрах процесора, що допомогло оптимізувати виконання програми.

Спостереження за роботою програми за допомогою «Диспетчера задач» підтвердили, що під час виконання програми три обрані ядра процесора використовувалися дуже інтенсивно, на 100%. Це свідчить про те, що програма ефективно розподіляє обчислювальне навантаження між потоками.

Проте під час виконання було виявлено ще одну невелику проблему: при запуску програми з невеликим значенням N всі три функції одночасно намагаються запитати свій перший елемент даних в консолі. Це явище виникає через властивість багатопоточності, де кожен потік працює незалежно та може одночасно намагатися виконати певні операції.

У рамках цієї лабораторної роботи проблему не було вирішено через обмеження на використання засобів, які зазвичай застосовуються для контролю доступу до спільних ресурсів. Одним з способів вирішення, є наприклад використання потокобезпечних методів вводу, але це потребує додаткових знань, які виходять за рамки цієї лабораторної роботи.

Лістинг програми.

```
// Програмне забезпечення високопродуктивних комп'ютерних систем
// Лабораторна робота №1: програмування потоків, потоки в мові C#
// номер в списку групи 3: 1.24  2.19  3.15
// F1: E = A + C *(MA*ME) + B
// F2: k = MAX(MF + MG*ML)
// F3: S = (O+P)*TRANS(MR * MT)
// Бащак Ярина Володимирівна
// група ІМ-11
// 16.02.2024

using System.Runtime.InteropServices;
using System.Text;
using lab1;

class Lab1
{
    public static int N;
    [DllImport("kernel32.dll")]
    static extern IntPtr GetCurrentThread();

    [DllImport("kernel32.dll")]
    static extern IntPtr SetThreadAffinityMask(IntPtr hThread, IntPtr
dwThreadAffinityMask);
```

```
    static void Main(string[] args)
    {
        Console.WriteLine("Введіть значення N:");
        while (!int.TryParse(Console.ReadLine(), out N) || N <= 0)
        {
            Console.WriteLine("Будь ласка, введіть коректне ціле число більше 0:");
        }

        int stackSize = 1024 * 1024;

        var t1 = new Thread(Func1, stackSize)
        {
            Name = "T1",
            Priority = ThreadPriority.Normal
        };

        var t2 = new Thread(Func2, stackSize)
        {
            Name = "T2",
            Priority = ThreadPriority.Normal
        };

        var t3 = new Thread(Func3, stackSize)
        {
            Name = "T3",
            Priority = ThreadPriority.Normal
        };

        t1.Start();
        t2.Start();
        t3.Start();
    }

    static void SetThreadAffinity(int processorNumber)
    {
        if (OperatingSystem.IsWindows())
        {
            IntPtr ptrThread = GetCurrentThread();
            SetThreadAffinityMask(ptrThread, new IntPtr(1 << processorNumber));
            Console.WriteLine($"Потік {Thread.CurrentThread.Name} виконується на ядрі
{processorNumber}");
        }
    }

    static void Func1()
    {
        Console.WriteLine("T1 start");

        // встановлення номера ядра
        SetThreadAffinity(0);

        var data = new Data(N);
        data.InitializeForFunc1();
        var result = data.F1();

        var output = new StringBuilder();
        output.Append("T1 result: ");
        foreach (var item in result)
        {
            output.Append(item + " ");
        }

        data.OutputResults(output.ToString(), "results_func1.txt");
```

```
        Console.WriteLine("T1 finish");
    }

    static void Func2()
    {
        Console.WriteLine("T2 start");

        // встановлення номера ядра
        SetThreadAffinity(1);

        var data = new Data(N);
        data.InitializeForFunc2();
        var result = data.F2();

        var output = "T2 result: " + result;

        data.OutputResults(output, "results_func2.txt");
        Console.WriteLine("T2 finish");
    }

    static void Func3()
    {
        Console.WriteLine("T3 start");
        // встановлення номера ядра
        SetThreadAffinity(2);

        var data = new Data(N);
        data.InitializeForFunc3();
        var result = data.F3();

        var output = new StringBuilder();
        output.Append("T3 result: ");
        foreach (var item in result)
        {
            output.Append(item + " ");
        }

        data.OutputResults(output.ToString(), "results_func3.txt");
        Console.WriteLine("T3 finish");
    }
}
```

```
namespace lab1;

public class Data
{
    public int N { get; private set; }
    public int[] A { get; private set; }
    public int[] B { get; private set; }
    public int[] C { get; private set; }
    public int[,] MA { get; private set; }
    public int[,] ME { get; private set; }
    public int[,] MF { get; private set; }
    public int[,] MG { get; private set; }
    public int[,] ML { get; private set; }
    public int[,] MR { get; private set; }
    public int[,] MT { get; private set; }
    public int[] O { get; private set; }
    public int[] P { get; private set; }
    private readonly Random random = new();

    public Data(int n)
```

```
    {
        N = n;
        A = new int[N];
        B = new int[N];
        C = new int[N];
        MA = new int[N, N];
        ME = new int[N, N];
        MF = new int[N, N];
        MG = new int[N, N];
        ML = new int[N, N];
        MR = new int[N, N];
        MT = new int[N, N];
        O = new int[N];
        P = new int[N];
    }

    public void InitializeForFunc1()
    {
        if (N < 100)
        {
            for (int i = 0; i < N; i++)
            {
                A[i] = ReadValue($"A[{i}]", 1);
                B[i] = ReadValue($"B[{i}]", 1);
                C[i] = ReadValue($"C[{i}]", 1);
                for (int j = 0; j < N; j++)
                {
                    MA[i, j] = ReadValue($"MA[{i},{j}]", 1);
                    ME[i, j] = ReadValue($"ME[{i},{j}]", 1);
                }
            }
        }
        else
        {
            var filePath = "func1_data.txt";
            if (File.Exists(filePath))
            {
                LoadFromFile(filePath, new int[][] { A, B, C }, new int[][,] { MA, ME
});
            }
            else
            {
                // якщо файлу нема, всі елементи задаються або фіксованим числом,
наприклад, 1, або рандомним чином.
                // щоб змінити спосіб задання потрібно розкоментувати і закоментувати
виклики відповідних функцій нижче

                var fixedValue = 1;
                InitializeWithFixedValue(new int[][] { A, B, C }, new int[][,] { MA, ME
}, fixedValue, filePath);
                // InitializeWithRandomValues(new int[][] { A, B, C }, new int[][,] {
MA, ME }, filePath);
            }
        }
    }

    public void InitializeForFunc2()
    {
        if (N < 100)
        {
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < N; j++)
                {
```

```
                    MF[i, j] = ReadValue($"MF[{i},{j}]", 2);
                    MG[i, j] = ReadValue($"MG[{i},{j}]", 2);
                    ML[i, j] = ReadValue($"ML[{i},{j}]", 2);
                }
            }
        }
        else
        {
            var filePath = "func2_data.txt";
            if (File.Exists(filePath))
            {
                LoadFromFile(filePath, Array.Empty<int[]>(), new int[][,] { MF, MG, ML
});
            }
            else
            {
                // якщо файлу нема, всі елементи задаються або фіксованим числом,
наприклад, 2, або рандомним чином.
                // щоб змінити спосіб задання потрібно розкоментувати і закоментувати
виклики відповідних функцій нижче

                // var fixedValue = 2;
                // InitializeWithFixedValue(Array.Empty<int[]>(), new int[][,] { MF,
MG, ML }, fixedValue, filePath);
                InitializeWithRandomValues(Array.Empty<int[]>(), new int[][,] { MF, MG,
ML }, filePath);
            }
        }
    }

    public void InitializeForFunc3()
    {
        if (N < 100)
        {
            for (int i = 0; i < N; i++)
            {
                O[i] = ReadValue($"O[{i}]", 3);
                P[i] = ReadValue($"P[{i}]", 3);
                for (int j = 0; j < N; j++)
                {
                    MR[i, j] = ReadValue($"MR[{i},{j}]", 3);
                    MT[i, j] = ReadValue($"MT[{i},{j}]", 3);
                }
            }
        }
        else
        {
            var filePath = "func3_data.txt";
            if (File.Exists(filePath))
            {
                LoadFromFile(filePath, new int[][] { O, P }, new int[][,] { MR, MT });
            }
            else
            {
                // якщо файлу нема всі елементи задаються або фіксованим числом,
наприклад, 3, або рандомним чином.
                // щоб змінити спосіб задання потрібно розкоментувати і закоментувати
виклики відповідних функцій нижче

                var fixedValue = 3;
                InitializeWithFixedValue(new int[][] { O, P }, new int[][,] { MR, MT },
fixedValue, filePath);
                // InitializeWithRandomValues(new int[][] { O, P }, new int[][,] { MR,
MT }, filePath);
```

```
            }
        }
    }

    private void LoadFromFile(string filePath, int[][] vectors, int[][,] matrices)
    {
        string[] lines = File.ReadAllLines(filePath);

        int lineIndex = 1;
        // читаємо вектори
        foreach (var vector in vectors)
        {
            string[] row = lines[lineIndex].Split(',');
            for (int i = 0; i < N; i++)
            {
                vector[i] = int.Parse(row[i]);
            }
            lineIndex += 2;
        }

        // читаємо матриці
        foreach (var matrix in matrices)
        {
            for (int i = 0; i < N; i++)
            {
                string[] row = lines[lineIndex++].Split(',');
                for (int j = 0; j < N; j++)
                {
                    matrix[i, j] = int.Parse(row[j]);
                }
            }
            lineIndex++;
        }

        Console.WriteLine($"Data from {filePath} was read");
    }

    private void InitializeWithFixedValue(int[][] vectors, int[][,] matrices, int
fixedValue, string filePath)
    {
        foreach (var vector in vectors)
        {
            for (int i = 0; i < N; i++)
            {
                vector[i] = fixedValue;
            }
        }

        foreach (var matrix in matrices)
        {
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < N; j++)
                {
                    matrix[i, j] = fixedValue;
                }
            }
        }

        Console.WriteLine($"File {filePath} not found, so data was set to fixed value
{fixedValue}");
    }
```

```
    private void InitializeWithRandomValues(int[][] vectors, int[][,] matrices, string
filePath)
    {
        foreach (var vector in vectors)
        {
            for (int i = 0; i < N; i++)
            {
                vector[i] = random.Next(1, 4);
            }
        }

        foreach (var matrix in matrices)
        {
            for (int i = 0; i < N; i++)
            {
                for (int j = 0; j < N; j++)
                {
                    matrix[i, j] = random.Next(1, 4);
                }
            }
        }

        Console.WriteLine($"File {filePath} not found, so data was randomly provided");
    }

    public void OutputResults(string output, string fileName)
    {
        if (N < 100)
        {
            Console.WriteLine(output);
        }
        else
        {
            File.WriteAllText(fileName, output);
        }
    }

    public int[] F1()
    {
        // E = A + C *(MA*ME) + B
        int[] E = new int[N];
        int[,] MAMEResult = MultiplyMatrices(MA, ME);
        int[] CMAMEResult = MultiplyMatrixVector(MAMEResult, C);

        for (int i = 0; i < N; i++)
        {
            E[i] = A[i] + CMAMEResult[i] + B[i];
        }

        return E;
    }

    public int F2()
    {
        // k = MAX(MF + MG*ML)
        int[,] MGMLResult = MultiplyMatrices(MG, ML);
        int[,] MFPlusMGML = new int[N, N];

        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < N; j++)
            {
                MFPlusMGML[i, j] = MF[i, j] + MGMLResult[i, j];
            }
```

```
        }

        return FindMax(MFPlusMGML);
    }

    public int[] F3()
    {
        // S = (O+P)*TRANS(MR * MT)
        int[,] MRMTResult = MultiplyMatrices(MR, MT);
        int[,] MRMTTransposed = TransposeMatrix(MRMTResult);
        int[] OPResult = new int[N];

        for (int i = 0; i < N; i++)
        {
            OPResult[i] = O[i] + P[i];
        }

        return MultiplyMatrixVector(MRMTTransposed, OPResult);
    }

    private static int[] MultiplyMatrixVector(int[,] matrix, int[] vector)
    {
        int n = matrix.GetLength(0);

        int[] result = new int[n];
        for (int i = 0; i < n; i++)
        {
            result[i] = 0;
            for (int j = 0; j < n; j++)
            {
                result[i] += matrix[i, j] * vector[j];
            }
        }
        return result;
    }

    private static int[,] MultiplyMatrices(int[,] matrix1, int[,] matrix2)
    {
        int n = matrix1.GetLength(0);

        int[,] result = new int[n, n];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                result[i, j] = 0;
                for (int k = 0; k < n; k++)
                {
                    result[i, j] += matrix1[i, k] * matrix2[k, j];
                }
            }
        }
        return result;
    }

    private static int FindMax(int[,] matrix)
    {
        int n = matrix.GetLength(0);

        int max = matrix[0, 0];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
```

```
                if (matrix[i, j] > max)
                {
                    max = matrix[i, j];
                }
            }
        }
        return max;
    }

    private static int[,] TransposeMatrix(int[,] matrix)
    {
        int n = matrix.GetLength(0);

        int[,] transposed = new int[n, n];
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n; j++)
            {
                transposed[j, i] = matrix[i, j];
            }
        }
        return transposed;
    }

    private int ReadValue(string prompt, int defaultValue)
    {
        Console.Write($"{prompt}: ");
        if (int.TryParse(Console.ReadLine(), out int value))
        {
            return value;
        }
        else
        {
            return defaultValue;
        }
    }
}
```

файл для вводу даних func1_data.txt (N = 1000)
// Вектор A

1,2,3,4,5,1,2,3,4,5,1,2,3,4,5,…
// Вектор B
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…
// Вектор C
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…
// Матриця MA
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…
… ще 998 рядків
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…
// Матриця ME
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…
… ще 998 рядків
1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,…