Використання багатопотоковості для обчислення математичних функцій

Ціль роботи

– організація потоків на мові C# при паралельному виконанні коду через реалізацію методів багатопоточності для обчислення математичної функції в середовищі програмування Microsoft Visual Studio 2010+.

Робоче завдання

- Реалізувати просту математичну функцію на мові C# в середовищі програмування Microsoft Visual Studio 2010+ з використанням методів багатопоточності.
- Дослідити алгоритм виконання програми на мові C# та рівень забезпечення потокової безпеки при паралельному виконанні коду. Описати застосування потокових методів в середовищі програмування Microsoft Visual Studio 2010+.
- Задокументувати у звіті по виконаній роботі висновки за результатами досліджень і підготувати відповіді на контрольні питання.

Знайти суму норм векторів, що утворюються із головної та побічної діагоналей квадратної матриці. Матриця задається рандомно. Розмірність матриці вводиться з консолі.

```
namespace Program
{
   class MainProgram
   {


      public static void Main(string[] args)
      {
         while (true)
         {
            Console.WriteLine("5. Знайти суму норм векторів, що утворюються із головної та побічної діагоналей квадратної матриці.Матриця задається рандомно.Розмірність матриці вводиться з консолі\n" );


            string input = Console.ReadLine();
            if (input == "5") { new NormVector(); }
            else if (input == "0") { Environment.Exit(0); }
```

```
            }
        }

    }
class NormVector
    {
        private static int a;
        private static int[,] matrix;
        private static int vecNorm = 1;
        private static List<int> generalVector;
        private static List<int> subVector;
        private static double gvn;
        private static double svn;



        public NormVector()
        {


            Console.WriteLine("Введіть ромір матриці: ");
            a = Convert.ToInt32(Console.ReadLine());
            Console.WriteLine("Обрахувати:\n1:Евклідова норма\n2:Манхеттенська норма\nВведіть номер: ");
            if(Console.ReadLine() == "2") { vecNorm = 2; }
            else if (Console.ReadLine() == "1") { vecNorm = 1; }


            matrix = new int[a, a];
            generalVector = new List<int>();
            subVector = new List<int>();
            MatrixGener();


            Thread gv = new Thread(GeneralVector);
            gv.Start();
            Thread sv = new Thread(SubVector);
```

```
    sv.Start();

    Task.Run(() =>
    {
        gv.Join();
        sv.Join();

        Console.Write("Вектор головної діагоналі: ");
        printVector(generalVector);
        Console.WriteLine($"Норма вектора головної діагоналі: {gvn}");
        Console.Write("Вектор побічної діагоналі: ");
        printVector(subVector);
        Console.WriteLine($"Норма вектора побічної діагоналі: {svn}");
        Console.WriteLine($"Сума норм векторів {gvn + svn}");
    });
}

static void MatrixGener()
{
    Random rnd = new();

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

        for (int j = 0; j < a; j++)
        {
            matrix[i, j] = Convert.ToInt32(rnd.Next(-100, 100));
        }
    }

    for (int i = 0; i < a; i++)
    {
        for (int j = 0; j < a; j++)
```

```
        {
            Console.Write(matrix[i, j] + " ");
        }
        Console.WriteLine();
    }

}

static void GeneralVector()
{
    for (int i = 0; i < a; i++) {
        generalVector.Add(matrix[i,i]);
    }

    gvn = findVectorNorm(generalVector);
}

static void SubVector()
{
    for (int i = 0; i < a; i++)
    {
        subVector.Add(matrix[a-1-i, i]);
    }

    svn = findVectorNorm(subVector);
}

static double findVectorNorm(List<int> vector) {
    double res = 0;
    if (vecNorm == 1)
    {
        double sum = 0;
        foreach (int i in vector) { sum += Math.Pow(i, 2); }
```

```
            res = Math.Sqrt(sum);
        }
        else if (vecNorm == 2)
        {
            foreach (int i in vector) { res += Math.Abs(i); }
        }

        return res;
    }

    static void printVector(List<int> vector)
    {
        foreach (int i in vector) { Console.Write(i + " "); }
        Console.WriteLine();
    }
  }
}
```

Контрольні питання

1. У чому полягає відмінність між процесами та потоками?

Процес складається з коду, даних і інших системних ресурсів, таких як відкриті файли, канали (pipes), що синхронізують об'єкти. Потік (thread) - базовий об'єкт, якому операційна система розподіляє час центрального процесора.

2. Які є способи розподілення потоків або синхронізації дій потоків?

існує кілька способів розподілення потоків або синхронізації дій потоків у багатозадачних програмах. Вибір конкретного способу залежить від завдання та потреб програми. Ось деякі з найпоширеніших способів:

- Взаємовиключення (Mutex): Використовується для управління доступом до ресурсів, що можуть бути конфліктними, таких як спільна пам'ять або файли. Mutex дозволяє тільки одному потоку отримувати доступ до ресурсу в конкретний момент часу, інші потоки очікують на доступ.
- Семафори (Semaphores): Семафори дозволяють обмінювати сигналами між потоками та контролювати доступ до обмеженого кількості ресурсів. Наприклад, семафор може дозволити обмінювати даними між двома потоками лише після того, як обидва потоки готові.

- Умовні змінні (Condition Variables): Використовуються для створення бар'єрів для потоків, де деякі потоки чекають, доки інші потоки виконають певні дії. Зазвичай використовуються разом з м'ютексами.
- Блокування та чекання (Lock and Wait): Потоки можуть блокувати і чекати на виконання певних умов або подій, періодично перевіряючи ці умови.
- Атомарні операції: Деякі мови програмування надають можливість виконувати атомарні операції, які не потребують блокування і дозволяють безпечно змінювати спільні дані.

3. Поясніть створення потоків з використанням конструктора класу Thread.

Thread(ThreadStart): в якості параметру примає обєкт делегату ThreadStart, який представляє виконувану в потоці дію.

4. Поясніть виклик методу Start для створеного потоку.

Змушує операційну систему змінити стан поточного екземпляра на Running.