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+.

Індивідуальне завдання

Використовуючи наведений у п.2 приклад і методичні вказівки написати програму, яка в режимі двох або більше користувацьких потоків, роз’язує наступне завдання:

1. Знайти попарний скалярний добуток рядків прямокутної матриці. Кількість рядків матриці є парною. Матриця задається рандомно. Розмірність матриці вводиться з консолі.

Хід роботи:

* Реалізація пошуку попарного скалярного добутку полягає в знаходженні суми добутків сусідніх елементів матриці. Алгоритм реалізовується в циклах.

```
public static void ThreadFunction1()
{
    int x = 0;

    for (int i = 0; i < N / 2; i++)
    {
        for (int j = 0; j < M; j++)
        {
            x = array[i, j] * array[i + 1, j];
            Scalar1 += x;
        }
    }
}

1 usage
static void ThreadFunction2()
{
    int z = 0;
    for (int i = 1; i < N - 1; i++)
    {
        for (int j = i + 1; j < M; j++)
        {
            z = matrix[i, j] * matrix[i + 1, j];
            Scalar2 += z;
        }
    }
```

* Для забезпечення безпеки та коректного виконання потоків передбачено затримку аби не спотворити результат одночасним доступом потоків до одного елементу.

```
while (thread1.IsAlive || thread2.IsAlive)
{
    Thread.Sleep( millisecondsTimeout: 50);
}
```

* У результаті можемо спостерігати виконання багатопотокової програми.

```
static void Main()
{
    matrix = CreateArray();

    Thread thread1 = new Thread(ThreadFunction1);
    thread1.Start();

    Thread thread2 = new Thread(ThreadFunction2);
    thread2.Start();

    while (thread1.IsAlive || thread2.IsAlive)
    {
        Thread.Sleep( millisecondsTimeout: 50);
    }

    Console.WriteLine($"{Scalar1 + Scalar2}");
}
```

```
Enter n:
6
Enter m:
5
4 2 0 1 0
0 4 0 2 1
3 0 2 2 3
1 3 3 4 3
4 4 1 1 0
1 1 4 2 2

80
```

Контрольні питання

1. У чому полягає відмінність між процесами та потоками?

Процеси та потоки пов'язані один з одним, але при цьому мають суттєві відмінності.

Процес - екземпляр програми під час виконання, незалежний об'єкт, якому виділено системні ресурси (наприклад, процесорний час і пам'ять). Кожен процес виконується в окремому адресному просторі: один процес не може отримати доступ до змінних і структур даних іншого. Якщо процес хоче отримати доступ до чужих ресурсів, необхідно використовувати міжпроцесну взаємодію.

Потік використовує той самий простір стека, що і процес, а безліч потоків спільно використовують дані своїх станів. Як правило, кожен потік може працювати (читати і писати) з однією і тією ж областю пам'яті, на відміну від процесів, які не можуть просто так отримати доступ до пам'яті іншого процесу. У кожного потоку є власні регістри і власний стек, але інші потоки можуть їх використовувати.

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

При будь-якій взаємодії між процесами необхідною є взаємна синхронізація (synchronization). Існує два основних види синхронізації — взаємне виключення (mutual exclusion) та умовна синхронізація (condition synchronization). Взаємне виключення забезпечує, щоб критичні секції операторів не виконувалися одночасно. Умовна синхронізація затримує процес до тих пір, поки не виконається певна умова.

3. Поясніть створення потоків з використанням конструктора класу Thread.

Для створення потоків використовується конструктор класу Thread, що приймає у якості параметра – делегат типу ThreadStart, який вказує метод, що потрібно виконати.

4. Поясніть виклик методу Start для створеного потоку.

Делегат ThreadStart на мові C# визначається наступним чином:

```
public delegate void ThreadStart();
```

Виклик методу Start починає виконання потоку. Потік триває до виходу з виконуваного методу.