# 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+.

### Теоретичні відомості

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

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. Наприклад, програмний код для створення делегата ThreadStart має наступний вигляд:

class ThreadTest {

static void Main() {

Thread t = new Thread(new ThreadStart(Go));

t.Start();

// виконати Go() в новому потоці Go();

// одночасно запустити Go() в головному потоці

}

static void Go() { Console.WriteLine("flow method"); }

}

Фоновий потік виконує метод Go() одночасно з головним потоком. Результатом виконування декількох потоків є майже одночасний вивід повідомлення «flow method». Потік, який закінчив виконання, не може бути початий наново, що забезпечує потокову безпеку при паралельному виконанні коду.

### Робоче завдання

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

### Методичні вказівки

1. Для виконання даної лабораторної роботи та обчислення функції користувача на мові С# необхідно використати конструктор класу Thread та виклик методу Start для виконання потоку. Виконати ініціалізацію даних і змінних у робочому проекті в середовищі програмування Microsoft Visual Studio 2010+:

// Ініціалізація даних і змінних

using System;

using System.Threading;

namespace Example {

class MainClass {

// Ініціалізація масиву значень

static double[] x = {1, 2, 3, 4, 5};

static double[] y = {7.1, 27.8, 62.1, 110, 161};

static int n = 0;

// Підсумкові коефіцієнти рівняння

static double a1, b1, a2, b2;

// Похибка розрахунків

static double d1, d2;

public static void Main (string[] args) {

if (x.Length == y.Length) {

n = x.Length;

};

// Нормалізація даних

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

x [i] = Math.Log (x [i]);

}

// Створюємо перший потік

Thread thread1 = new Thread(ThreadFunction1);

thread1.Start();

// Створюємо другий потік

Thread thread2 = new Thread(ThreadFunction2);

thread2.Start();

// Підсумкова формула з найменшою похибкою

if (d1 < d2) {

Console.WriteLine ("Result Point Vector: "); Console.WriteLine ("y = " + a1 + "\* lnx +"+b1);

}

else {

Console.WriteLine ("Result Point Vector: ");

Console.WriteLine ("y = " +

Math.Pow (Math.E, a2) + " \* x^ " + b2);

}

}

}

1. Реалізувати процедури обчислення функцій ThreadFunction(), що використовуються при запуску потоків thread1 та thread2 паралельно з головним потоком:

// Реалізація функцій користувача

static void ThreadFunction1() {

// Компоненти для вирішення системи

double Xi = 0;

double Xi2 = 0;

double XiYi = 0;

double Yi = 0;

// Знайдемо необхідні компоненти для вирішення системи

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

Xi += x [i];

Xi2 += x [i] \* x [i];

XiYi += x [i] \* y [i]; Yi += y [i];

}

// Знайдемо підсумкові коефіцієнти і похибку

a1 = (Yi \* Xi2 \* n - XiYi \* n \* Xi) / (Xi2 \* n \* n - n \* Xi \* Xi);

b1 = (XiYi \* n - Yi \* Xi) / (Xi2 \* n - Xi \* Xi);

d1 = Math.Sqrt (((Yi - a1 \* Xi - b1) \* (Yi - a1 \* Xi - b1)) / (Yi \* Yi));

Console.WriteLine ("d1 = " + d1);

}

static void ThreadFunction2() {

double Xi = 0;

double Xi2 = 0;

double XiYi = 0;

double Yi = 0;

// Нормалізація даних для y = a\*x^b

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

y [i] = Math.Log (y [i]);

}

// Знайдемо необхідні компоненти для вирішення системи

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

Xi += x [i];

Xi2 += x [i] \* x [i];

XiYi += x [i] \* y [i];

Yi += y [i];

}

// Знайдемо підсумкові коефіцієнти і похибку

a2 = (Yi \* Xi2 \* n - XiYi \* n \* Xi) / (Xi2 \* n \* n - n \* Xi \* Xi);

b2 = (XiYi \* n - Yi \* Xi) / (Xi2 \* n - Xi \* Xi);

d2 = Math.Sqrt (((Yi - a2 \* Xi - b2) \* (Yi - a2 \* Xi - b2)) 48 48 / (Yi \* Yi));

Console.WriteLine ("d2 = " + d2);

}

1. За допомогою візуальних компонентів MFC для програмних додатків дослідити алгоритм виконання коду в середовищі програмування Microsoft Visual Studio 2010+.

### Індивідуальні 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 суму елементів матриці, як суму елементів верхньої трикутної та нижньої трикутною підматриць. Матриця задається рандомно. Розмірність матриці вводиться з консолі.
2. Знайти попарний скалярний добуток рядків прямокутної матриці. Кількість рядків матриці є парною. Матриця задається рандомно. Розмірність матриці вводиться з консолі.
3. Знайти суму цілих елементів проміжку [0, n] як суму усіх парних і непарних елементів. Значення n вводиться з консолі.
4. Згенерувати множину із n дійсних чисел. Знати добуток перших n/2 і суму решти елементів множини. Значення n є парним і вводиться з консолі.
5. Знайти суму норм векторів, що утворюються із головної та побічної діагоналей квадратної матриці. Матриця задається рандомно. Розмірність матриці вводиться з консолі.
6. Знайти суму найменших елементів стовпців матриці. Матриця задається рандомно. Розмірність матриці вводиться з консолі.
7. Знайти середнє найбільших елементів рядків матриці. Матриця задається рандомно. Розмірність матриці вводиться з консолі.

### Контрольні питання

1. У чому полягає відмінність між процесами та потоками?
2. Які є способи розподілення потоків або синхронізації дій потоків?
3. Поясніть створення потоків з використанням конструктора класу Thread.
4. Поясніть виклик методу Start для створеного потоку.