НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Лабораторна робота №3

з дисципліни **«**Програмування паралельних комп’ютерних систем**»**

Виконав:

студент 3 курсу

ФІОТ гр. ІО-21

Кузьменко Володимир

Перевірив:

Корочкін О. В.

Київ – 2015 р.

Тема: Програмування для комп’ютерних систем зі спільною пам’яттю. Мова C#

Розробити програму для розв’язання в ПКС зі СП (структура на рис. Рисунок 1) математичної задачі: A = sort(B)∙α + β∙Z∙(MO∙MK)

Бібліотека: C#.

Засоби організації взаємодії: примітиви синхронізації (Semaphore, Mutex, Monitor, Lock), volatile-змінні.



Структурна схема ПКС

**Виконання роботи:**

**Етап 1. Побудова паралельного алгоритму**

1. BH = sort(BH)
2. B2H = mergeSort (BH, BH)
3. AH = BH∙α + β∙Z∙(MO∙MKH)

Спільний ресурс: α, β, Z, MO

**Етап 2. Розроблення алгоритмів роботи кожного процесу**

|  |  |
| --- | --- |
| **Задача Т1** | ТС, КД |
| 1. Введення MK, α |  |
| 2. **Чекати** на введення Z, B в T4 | W4 – 1 |
| 3. Обчислення BH = sort(BH) |  |
| 4. **Чекати** на завершення обчислень BH в T2 | W2 – 2 |
| 5. Злиття B2H = mergeSort(BH, BH) |  |
| 6. **Сигнал** Т4 про завершення злиття в Т1 | S4 – 1 |
| 7. **Чекати** завершення злиття B в Т4 | W4-3 |
| 8. Копіювати α1 := α | КД |
| 9. Копіювати β1 := β | КД |
| 10. Копіювати Z1 := Z | КД |
| 11. Копіювання MO1 := MO | КД |
| 12. Обчислення AH = BH∙α1 + β1∙Z1∙(MO1∙MKH) |  |
| 13. **Сигнал** Т4 про завершення обчислень | S4 – 2 |
| **Задача T2** | ТС, КД |
| 1. **Чекати** на введення B, Z у задачі T4 | W4 – 1 |
| 2. Обчислення BH = sort(BH) |  |
| 3. **Сигнал** Т1 про завершення сортування BH | S1 – 1 |
| 4. **Чекати** на завершення злиття B в Т4 | W4 – 2 |
| 5. Копіювати α2 := α | КД |
| 6. Копіювати β2 := β | КД |
| 7. Копіювати Z2 := Z | КД |
| 8. Копіювання MO2 := MO | КД |
| 9. Обчислення AH = BH∙α2 + β2∙Z2∙(MO2∙MKH) |  |
| 10. **Сигнал** Т4 про завершення обчислень | S4– 2 |
| **Задача Т3** | ТС, КД |
| 1. Введення МO, β |  |
| 2. **Чекати** на введення B, Z у задачі T4 | W4 – 1 |
| 3. Обчислення BH = sort(BH) |  |
| 4. **Сигнал** Т4 про завершення BH | S4 – 1 |
| 5. **Чекати** на завершення злиття B в Т4 | W4 – 2 |
| 6. Копіювати α3 := α | КД |
| 7. Копіювати β3 := β | КД |
| 8. Копіювати Z3 := Z | КД |
| 9. Копіювання MO3 := MO | КД |
| 10. Обчислення AH = BH∙α3 + β3∙Z3∙(MO3∙MKH) |  |
| 11. **Сигнал** Т4 про завершення обчислень | S4 – 2 |
| **Задача Т4** | ТС, КД |
| 1. Введення B, Z |  |
| 2. **Сигнал** задачам T1, T2, T3 про введення B, Z | S1,2,3 – 1 |
| 3. Обчислення BH = sort(BH) |  |
| 4. **Чекати** на завершення обчислень BH в T3 | W3 – 1 |
| 5. Злиття B2H = mergeSort(BH, BH) |  |
| 6. **Чекати** завершення злиття в Т1 | W1 – 2 |
| 7. B = mergeSort(B2H, B2H) |  |
| 8. **Сигнал** задачам T1, T2, Т3 про завершення злиття B | S1,2,3 – 2 |
| 9. Копіювати α4 := α | КД |
| 10. Копіювати β4 := β | КД |
| 11. Копіювати Z4 := Z | КД |
| 12. Копіювання MO4 := MO | КД |
| 13. Обчислення AH = BH∙α4 + β4∙Z4∙(MO4∙MKH) |  |
| 14. **Чекати** завершення обчислень AH в задачах T1, T2, T3 | W1,2,3 – 3 |
| 15. Вивести A |  |

**Етап 3. Розроблення структурної схеми взаємодії задач**

На структурній схемі взаємодії задач уведено такі умовні позначення

* Evn4\_123 – подія для синхронізації із завершенням вводу в T4
* Evn4\_B – подія для синхронізації із завершенням злиття В в T4
* sem2\_1 семафор для синхронізації завершення сортування BH у задачі Т2
* sem1\_4 семафор для синхронізації завершення злиття B2H у задачі Т1
* sem3\_4 семафор для синхронізації завершення сортування BH у задачі Т3
* sem2А семафор для синхронізації завершення обчислень AH у задачі T2
* sem1А семафор для синхронізації завершення обчислень AH у задачі T1
* sem3А семафор для синхронізації завершення обчислень AH у задачі T3
* volatile – ключове слово для доступу до спільного ресурсу α
* mutex\_beta – мютекс для доступу до спільного ресурсу β
* lock\_Z – замок для доступу до спільного ресурсу Z
* monitor\_MO – монітор для доступу до спільного ресурсу MO



Структурна схема взаємодії задач

Етап 4. Розроблення програми

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

/\*\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \*

\* Parallel and Distributed Computing \*

\* Laboratory work #3. C# \*

\* \*

\* Task: A = sort(B)\*α + β\*Z\*(MO\*MK) \*

\* \*

\* File Program.cs \*

\* @author Kuzmenko Volodymyr \*

\* @group IO-21 \*

\* @date 24.03.15 \*

\* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\*/

namespace pro2\_lab3

{

class Program : Operations

{

public const int N = 1600;

public const int P = 4;

public const int H = N / P;

public static EventWaitHandle Evn4\_123;

public static EventWaitHandle Evn4\_B;

public static object lock\_Z = new object();

public static object monitor\_MO = new object();

public static Mutex mutex\_beta = new Mutex(false);

public static Semaphore sem2\_1, sem1\_4, sem3\_4, sem2A, sem1A, sem3A;

public static volatile int alpha;

public static int beta;

public static Vector A = new Vector(N);

public static Vector B = new Vector(N);

public static Vector Z = new Vector(N);

public static Matrix MO = new Matrix(N);

public static Matrix MK = new Matrix(N);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void T1()

{

Console.WriteLine("Task 1 started");

/\*1. Введення MK, α\*/

alpha = 1;

MK = inputMatrix(N,1);

/\*2. Чекати на введення Z, B в T4\*/

Evn4\_123.WaitOne();

/\*3. Обчислення BH = sort(BH)\*/

Vector buf = new Vector(H);

buf = sort(B, 0, H);

for (int i = 0; i < H; i++)

{

B.set(i, buf.get(i));

}

/\*4. Чекати на завершення обчислень BH в T2\*/

sem2\_1.WaitOne();

/\*5. Злиття B2H = mergeSort(BH, BH)\*/

mergeSort(B, 0, 2 \* H);

/\*6. Сигнал Т4 про завершення злиття в Т1\*/

sem1\_4.Release();

/\*7. Чекати завершення злиття B в Т4\*/

Evn4\_B.WaitOne();

/\*8. Копіювати α1 := α\*/

int alpha1 = alpha;

/\*9. Копіювати β1 := β\*/

mutex\_beta.WaitOne();

int beta1 = beta;

mutex\_beta.ReleaseMutex();

Vector Z1 = new Vector(N);

/\*10. Копіювати Z1 := Z\*/

lock (lock\_Z)

{

Z1 = Z;

}

Matrix MO1 = new Matrix(N);

/\*11. Копіювання MO1 := MO\*/

try

{

Monitor.Enter(monitor\_MO);

MO1 = MO;

}

finally

{

Monitor.Exit(monitor\_MO);

}

/\*12. Обчислення AH = BH∙α1 + β1∙Z1∙(MO1∙MKH) \*/

buf = add(mult(alpha1,B,0,H),mult(beta1,mult(Z,mult(MO1, MK,0,H),0,H),0,H),0,H);

for (int i = 0; i < H; i++)

{

A.set(i, buf.get(i));

}

/\*13. Сигнал Т4 про завершення обчислень\*/

sem1A.Release();

Console.WriteLine("Task 1 finished");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void T2()

{

Console.WriteLine("Task 2 started");

/\*1. Чекати на введення B, Z у задачі T4\*/

Evn4\_123.WaitOne();

Vector buf = new Vector(H);

/\*2. Обчислення BH = sort(BH)\*/

buf = sort(B, H, 2\*H);

for (int i = H; i < 2\* H; i++)

{

B.set(i, buf.get(i));

}

/\*3. Сигнал Т1 про завершення сортування BH \*/

sem2\_1.Release();

/\*4. Чекати на завершення злиття B в Т4\*/

Evn4\_B.WaitOne();

/\*5. Копіювати α2 := α\*/

int alpha2 = alpha;

/\*6. Копіювати β2 := β\*/

mutex\_beta.WaitOne();

int beta2 = beta;

mutex\_beta.ReleaseMutex();

Vector Z2 = new Vector(N);

/\*7. Копіювати Z2 := Z\*/

lock (lock\_Z)

{

Z2 = Z;

}

Matrix MO2 = new Matrix(N);

/\*8. Копіювання MO2 := MO\*/

try

{

Monitor.Enter(monitor\_MO);

MO2 = MO;

}

finally

{

Monitor.Exit(monitor\_MO);

}

/\*9. Обчислення AH = BH∙α2 + β2∙Z2∙(MO2∙MKH)\*/

buf = add(mult(alpha2, B, H, 2 \* H), mult(beta2, mult(Z, mult(MO2, MK, H, 2 \* H), H, 2 \* H), H, 2 \* H), H, 2 \* H);

for (int i = H; i < 2\*H; i++)

{

A.set(i, buf.get(i));

}

/\*10. Сигнал Т4 про завершення обчислень\*/

sem2A.Release();

Console.WriteLine("Task 2 finished");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т3 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void T3()

{

Console.WriteLine("Task 3 started");

/\*1. Введення МO, β\*/

beta = 1;

MO = inputMatrix(N, 1);

/\*2. Чекати на введення B, Z у задачі T4\*/

Evn4\_123.WaitOne();

/\*3. Обчислення BH = sort(BH)\*/

Vector buf = new Vector(H);

buf = sort(B, 2\*H, 3 \* H);

for (int i = 2\*H; i < 3 \* H; i++)

{

B.set(i, buf.get(i));

}

/\*4. Сигнал Т4 про завершення BH\*/

sem3\_4.Release();

/\*5. Чекати на завершення злиття B в Т4\*/

Evn4\_B.WaitOne();

/\*6. Копіювати α3 := α\*/

int alpha3 = alpha;

/\*7. Копіювати β3 := β\*/

mutex\_beta.WaitOne();

int beta3 = beta;

mutex\_beta.ReleaseMutex();

Vector Z3 = new Vector(N);

/\*8. Копіювати Z3 := Z\*/

lock (lock\_Z)

{

Z3 = Z;

}

Matrix MO3 = new Matrix(N);

/\*9. Копіювання MO3 := MO\*/

try

{

Monitor.Enter(monitor\_MO);

MO3 = MO;

}

finally

{

Monitor.Exit(monitor\_MO);

}

/\*10. Обчислення AH = BH∙α3 + β3∙Z3∙(MO3∙MKH)\*/

buf = add(mult(alpha3, B, 2\*H, 3 \* H), mult(beta3, mult(Z, mult(MO3, MK, 2\*H, 3 \* H), 2\*H, 3 \* H), 2\*H, 3 \* H),2\*H, 3 \* H);

for (int i = 2\*H; i < 3 \* H; i++)

{

A.set(i, buf.get(i));

}

/\*11. Сигнал Т4 про завершення обчислень\*/

sem3A.Release();

Console.WriteLine("Task 3 finished");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Задача Т4 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

public static void T4()

{

Console.WriteLine("Task 4 started");

/\*1. Введення B, Z\*/

B = inputVector(N , 1);

Z = inputVector(N, 1);

/\*2. Сигнал задачам T1, T2, T3 про введення B, Z\*/

Evn4\_123.Set();

Vector buf = new Vector(H);

/\*3. Обчислення BH = sort(BH)\*/

buf = sort(B, 3 \* H, N);

for (int i = 3 \* H; i < N; i++)

{

B.set(i, buf.get(i));

}

/\*4. Чекати на завершення обчислень BH в T3\*/

sem3\_4.WaitOne();

/\*5. Злиття B2H = mergeSort(BH, BH)\*/

mergeSort(B, 2\*H, N);

/\*6. Чекати завершення злиття в Т1\*/

sem1\_4.WaitOne();

/\*7. B = mergeSort(B2H, B2H)\*/

mergeSort(B, 0, N);

/\*8. Сигнал задачам T1, T2, Т3 про завершення злиття B\*/

Evn4\_B.Set();

/\*9. Копіювати α4 := α\*/

int alpha4 = alpha;

/\*10. Копіювати β4 := β\*/

mutex\_beta.WaitOne();

int beta4 = beta;

mutex\_beta.ReleaseMutex();

Vector Z4 = new Vector(N);

/\*11. Копіювати Z4 := Z\*/

lock (lock\_Z)

{

Z4 = Z;

}

/\*12. Копіювання MO4 := MO\*/

Matrix MO4 = new Matrix(N);

try

{

Monitor.Enter(monitor\_MO);

MO4 = MO;

}

finally

{

Monitor.Exit(monitor\_MO);

}

/\*13. Обчислення AH = BH∙α4 + β4∙Z4∙(MO4∙MKH)\*/

buf = add(mult(alpha4, B, 3 \* H, N), mult(beta4, mult(Z, mult(MO4, MK, 3 \* H, N), 3 \* H, N), 3 \* H, N), 3 \* H, N);

for (int i = 3 \* H; i < N; i++)

{

A.set(i, buf.get(i));

}

/\*14. Чекати завершення обчислень AH в задачах T1, T2, T3\*/

sem1A.WaitOne();

sem2A.WaitOne();

sem3A.WaitOne();

/\*15. Вивести A\*/

outputVector(A);

Console.WriteLine("Task 4 finished");

}

static void Main(string[] args)

{

System.Console.WriteLine("Lab3 started");

Evn4\_123 = new ManualResetEvent(false);

Evn4\_B = new ManualResetEvent(false);

sem1\_4 = new Semaphore(0, 1);

sem2\_1 = new Semaphore(0, 1);

sem3\_4 = new Semaphore(0, 1);

sem1A = new Semaphore(0, 1);

sem2A = new Semaphore(0, 1);

sem3A = new Semaphore(0, 1);

Thread t1 = new Thread(T1);

Thread t2 = new Thread(T2);

Thread t3 = new Thread(T3);

Thread t4 = new Thread(T4);

t1.Start();

t2.Start();

t3.Start();

t4.Start();

System.Console.WriteLine("Lab3 finished");

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \*

\* Parallel and Distributed Computing \*

\* Laboratory work #3. C# \*

\* \*

\* Task: A = sort(B)\*α + β\*Z\*(MO\*MK) \*

\* \*

\* File Vector.cs \*

\* @author Kuzmenko Volodymyr \*

\* @group IO-21 \*

\* @date 24.03.15 \*

\* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\*/

namespace pro2\_lab3

{

class Vector

{

private int[] array;

public Vector(int n)

{

array = new int[n];

}

public void set(int index, int value)

{

array[index] = value;

}

public int get(int index)

{

return array[index];

}

public int size()

{

return array.Length;

}

public String toString()

{

String res = "";

for (int i = 0; i < array.Length; i++)

{

res += " " + array[i];

}

return res;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \*

\* Parallel and Distributed Computing \*

\* Laboratory work #3. C# \*

\* \*

\* Task: A = sort(B)\*α + β\*Z\*(MO\*MK) \*

\* \*

\* File Matrix.cs \*

\* @author Kuzmenko Volodymyr \*

\* @group IO-21 \*

\* @date 24.03.15 \*

\* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\*/

namespace pro2\_lab3

{

class Matrix

{

private Vector[] vector;

public Matrix(int n)

{

vector = new Vector[n];

for (int i = 0; i < vector.Length; i++)

{

vector[i] = new Vector(n);

}

}

public void set(int n, int m, int val)

{

vector[n].set(m, val);

}

public int get(int n, int m)

{

return vector[n].get(m);

}

public Vector get(int index)

{

return vector[index];

}

public int size()

{

return vector.Length;

}

public String toString()

{

String res = "";

for (int i = 0; i < vector.Length; i++)

{

res += vector[i].toString();

if (i != vector.Length - 1)

{

res += "\n";

}

}

return res;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

/\*\*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\* \*

\* Parallel and Distributed Computing \*

\* Laboratory work #3. C# \*

\* \*

\* Task: A = sort(B)\*α + β\*Z\*(MO\*MK) \*

\* \*

\* File Operations.cs \*

\* @author Kuzmenko Volodymyr \*

\* @group IO-21 \*

\* @date 24.03.15 \*

\* \*

\* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \* \*

\*/

namespace pro2\_lab3

{

class Operations

{

public static Vector inputVector(int n, int value)

{

Vector vector = new Vector(n);

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

{

vector.set(i, value);

}

return vector;

}

public static void outputVector(Vector vector)

{

if (vector.size() < 9)

{

Console.WriteLine(vector.toString());

}

}

public static Vector mult(Vector left, Matrix right, int l, int r)

{

Vector result = new Vector(left.size());

for (int i = l; i < r; i++)

{

result.set(i, 0);

for (int j = 0; j < left.size(); j++)

{

result.set(i, result.get(i) + left.get(j) \* right.get(j, i));

}

}

return result;

}

public static Vector mult(int value, Vector vect, int l, int r)

{

Vector result = new Vector(vect.size());

for (int i = l; i < r; i++)

{

result.set(i, value \* vect.get(i));

}

return result;

}

public static Matrix inputMatrix(int n, int value)

{

Matrix matrix = new Matrix(n);

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

{

for (int j = 0; j < n; j++)

{

matrix.set(i, j, value);

}

}

return matrix;

}

public static void outputMatrix(Matrix matrix)

{

if (matrix.size() < 9)

{

Console.WriteLine(matrix.toString());

}

}

public static Matrix mult(Matrix left, Matrix right, int l, int r)

{

Matrix result = new Matrix(left.size());

for (int i = 0; i < left.size(); i++)

{

for (int j = l; j < r; j++)

{

result.set(i, j, 0);

for (int k = 0; k < left.size(); k++)

{

result.set(i, j, result.get(i, j) + left.get(i, k)

\* right.get(k, j));

}

}

}

return result;

}

public static Vector add(Vector left, Vector right, int l, int r)

{

Vector result = new Vector(left.size());

for (int i = l; i < r; i++)

{

result.set(i, left.get(i) + right.get(i));

}

return result;

}

public static Vector sort(Vector vector, int l, int r)

{

int tmp = vector.get(0);

Vector res;

res = vector;

for (int i = l; i < r; i++)

{

for (int k = i + 1; k < r; k++)

{

if (res.get(i) > res.get(k))

{

tmp = res.get(k);

res.set(k, res.get(i));

res.set(i, tmp);

}

}

}

return res;

}

private static int[] merge(int[] left, int[] right)

{

int a = 0, b = 0;

int[] merged = new int[left.Length + right.Length];

for (Int32 i = 0; i < left.Length + right.Length; i++)

{

if (b < right.Length && a < left.Length)

if (left[a] > right[b] && b < right.Length)

merged[i] = right[b++];

else

merged[i] = left[a++];

else

if (b < right.Length)

merged[i] = right[b++];

else

merged[i] = left[a++];

}

return merged;

}

public static void mergeSort(Vector vector, int l, int r)

{

if (vector.size() == 1)

return;

int mid = (r - l) / 2;

int[] merged = new int[r - l];

int[] array1 = new int[mid];

int[] array2 = new int[mid];

for (int i = 0; i < r - l; i++)

{

if (i < mid)

array1[i] = vector.get(i + l);

else

array2[i - mid] = vector.get(i + l);

}

merged = merge(array1, array2);

for (int i = l; i < r; i++)

vector.set(i, merged[i - l]);

}

}

}