Національний технічний університет України

«Київський Політехнічний Інститут»

Факультет інформатики і обчислювальної техніки

Кафедра обчислювальної техніки

**Лабораторна робота №3**

**З предмету «Паралельні та розподілені обчислення»**

**C#**

Виконав:

Студент  
IІІ курсу ФІОТ  
групи ІО-12  
Бута С. О.

Залікова книжка №1205

Київ-2014

**Техническое задание**

1. Структура ПКС с ОП: 
2. Задача: .
3. Язык программирования: C#.
4. Средства взаимодействия задач: семафоры, события, мютексы, мониторы.

**Выполнение работы**

1. Разработка параллельного математического алгоритма.
2. AH = sort(B\*MOH); OP: B;
3. A2H = mergeSort(AH, AH);
4. A = mergeSort(A2H, A2H);
5. DH = l\*C\*(MK\*MTH); OP: l, C, MK;
6. AH = AH + DH;
7. Разработка алгоритмов процессов.

**Задача Т1:**

1. Ввод МТ, В
2. Сигнал Т2,Т3,Т4 о вводе S2,3,4;1
3. Ждать завершения ввода в Т2,Т3,Т4 W2,3,4;1
4. Копия1 В КУ
   1. B1 = B
5. Счёт1
   1. AH = sort(B1\*MOH)
6. Ждать завершение счёт1 в Т2 W2;1
7. Счет2
   1. A2H = mergeSort(AH, AH)
8. Ждать завершение счет2 в Т3 W3;1
9. Счёт3
   1. A = mergeSort(A2H, A2H)
10. Сигнал Т2,Т3,Т4 о счёте3 S2,3,4;2
11. Копии2 l, C, MK КУ
    1. l1 = l
    2. C1 = C
    3. MK1 = MK
12. Счёт4
    1. DH = l1\*C1\*(MK1\*MTH)
13. Счёт5
    1. AH = AH + DH
14. Ждать завершение счёт5 в Т2,Т3,Т4 W2,3,4;2
15. Вывод А

**Задача Т2:**

1. Ввод MO
2. Сигнал Т1,Т3,Т4 о вводе S1,3,4;1
3. Ждать завершения ввода в Т1,Т3,Т4 W1,3,4;1
4. Копия B КУ
   1. B2 = B
5. Счёт1
   1. AH = sort(B2\*MOH)
6. Сигнал Т1 о завершении счёт1 S1;12
7. Копии2 l, C, MK КУ
   1. l2 = l
   2. C2 = C
   3. MK2 = MK
8. Счёт4
   1. DH = l2\*C2\*(MK2\*MTH)
9. Ждать завершения счёт3 в Т1 W1;12
10. Счёт5
    1. AH = AH + DH
11. Сигнал Т1 о завершении счет5 S1;22

**Задача Т3:**

1. Ввод C
2. Сигнал Т1,Т2,Т4 о вводе S1,2,4;1
3. Ждать завершения ввода в Т1,Т2,Т4 W1,2,4;1
4. Копия1 В КУ
   1. B3 = B
5. Счёт1
   1. AH = sort(B3\*MOH)
6. Ждать завершение счёт1 в Т1 W4;1
7. Счет2
   1. A2H = mergeSort(AH, AH)
8. Сигнал Т1 о завершении счёт2 S1;13
9. Копии2 l, C, MK КУ
   1. l3 = l
   2. C3 = C
   3. MK3 = MK
10. Счёт4
    1. DH = l3\*C3\*(MK3\*MTH)
11. Ждать завершения счёт3 в Т1 W1;13
12. Счёт5
    1. AH = AH + DH
13. Сигнал Т1 о завершении счет5 S1;23

**Задача Т4:**

1. Ввод l, MK
2. Сигнал Т1,Т2,Т3 о вводе S1,2,3;1
3. Ждать завершения ввода в Т1,Т2,Т3 W1,2,3;1
4. Копия1 В КУ
   1. B4 = B
5. Счёт1
   1. AH = sort(B4\*MOH)
6. Сигнал Т3 о завершении счёт1 S3;1
7. Копии2 l, C, MK КУ
   1. l4 = l
   2. C4 = C
   3. MK4 = MK
8. Счёт4
   1. DH = l4\*C4\*(MK4\*MTH)
9. Ждать завершения счёт3 в Т1 W1;14
10. Счёт5
    1. AH = AH + DH
11. Сигнал Т1 о завершении счет5 S1;14

**Разработка программы.**

**Листинг**:

Исходный код Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace PRO\_Lab\_3\_sharp\_

{

class Program

{

const int N = 8;

const int P = 4;

const int H = N / P;

const int FILLER = 1;

static int l;

static Vector A = new Vector(N, "A");

static Vector B = new Vector(N, "B");

static Vector C = new Vector(N, "C");

static Vector D = new Vector(N);

static Matrix MO = new Matrix(N, "MO");

static Matrix MK = new Matrix(N, "MK");

static Matrix MT = new Matrix(N, "MT");

static EventWaitHandle E1 = new ManualResetEvent(false);

static EventWaitHandle E2 = new ManualResetEvent(false);

static EventWaitHandle E3 = new ManualResetEvent(false);

static EventWaitHandle E4 = new ManualResetEvent(false);

static EventWaitHandle E1\_2 = new ManualResetEvent(false);

static Semaphore S2 = new Semaphore(0, 1);

static Semaphore S3 = new Semaphore(0, 1);

static Semaphore S4 = new Semaphore(0, 1);

static Semaphore S1 = new Semaphore(0, 3);

static Mutex M = new Mutex(false);

static object EE = new object();

static void Func1()

{

Console.WriteLine("T1 started.");

//Ввод МТ, В

MT.input();

B.input();

//Сигнал Т2,Т3,Т4 о вводе

E1.Set();

//Ждать завершения ввода в Т2,Т3,Т4

E2.WaitOne();

E3.WaitOne();

E4.WaitOne();

//Копия1 В

M.WaitOne();

Vector B1 = new Vector(N);

B1.copyFrom(B);

M.ReleaseMutex();

//Счёт1

A.copyFrom(B.multiplication(MO, 0, H - 1), 0);

A.sort(0, H);

//Ждать завершение счёт1 в Т2

S2.WaitOne();

//Счет2

A.copyFrom(A.mergeSort(A.partOfVector(A, 0, H), A.partOfVector(A, H, H)), 0);

//Ждать завершение счет2 в Т3

S3.WaitOne();

//Счёт3

A.copyFrom(A.mergeSort(A.partOfVector(A, 0, 2\*H), A.partOfVector(A, 2\*H, 2\*H)), 0);

//Сигнал Т2,Т3,Т4 о счёте3

E1\_2.Set();

//Копии2 l, C, MK

Monitor.Enter(EE);

int l1 = l;

Vector C1 = new Vector(N);

C1.copyFrom(C);

Matrix MK1 = new Matrix(N);

MK1.copyFrom(MK);

Monitor.Exit(EE);

//Счёт4

D.copyFrom(C1.multiplication(MK1.multiplication(MT, 0, H - 1), 0, H-1), 0);

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

{

D[i] \*= l1;

}

//Счёт5

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

{

A[i] += D[i];

}

//Ждать завершение счёт5 в Т2,Т3,Т4

S1.WaitOne();

S1.WaitOne();

S1.WaitOne();

//Вывод А

A.output();

Console.WriteLine("T1 finished.");

}

static void Func2()

{

Console.WriteLine("T2 started.");

//Ввод МО

MO.input();

//Сигнал Т1,Т3,Т4 о вводе

E2.Set();

//Ждать завершения ввода в Т1,Т3,Т4

E1.WaitOne();

E3.WaitOne();

E4.WaitOne();

//Копия1 В

M.WaitOne();

Vector B2 = new Vector(N);

B2.copyFrom(B);

M.ReleaseMutex();

//Счёт1

A.copyFrom(B.multiplication(MO, H, 2\*H - 1), H);

A.sort(H, H);

//Сигнал Т1 о завершении счёт1

S2.Release();

//Копии2 l, C, MK

Monitor.Enter(EE);

int l2 = l;

Vector C2 = new Vector(N);

C2.copyFrom(C);

Matrix MK2 = new Matrix(N);

MK2.copyFrom(MK);

Monitor.Exit(EE);

//Счёт4

D.copyFrom(C2.multiplication(MK2.multiplication(MT, H, 2\*H - 1), H, 2\*H - 1), H);

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

{

D[i] \*= l2;

}

//Ждать завершения счёт3 в Т1

E1\_2.WaitOne();

//Счёт5

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

{

A[i] += D[i];

}

//Сигнал Т1 о завершении счет5

S1.Release();

Console.WriteLine("T2 finished.");

}

static void Func3()

{

Console.WriteLine("T3 started.");

//Ввод С

C.input();

//Сигнал Т1,Т2,Т4 о вводе

E3.Set();

//Ждать завершения ввода в Т1,Т2,Т4

E1.WaitOne();

E2.WaitOne();

E4.WaitOne();

//Копия1 В

M.WaitOne();

Vector B3 = new Vector(N);

B3.copyFrom(B);

M.ReleaseMutex();

//Счёт1

A.copyFrom(B.multiplication(MO, 2\*H, 3\*H - 1), 2\*H);

A.sort(2\*H, H);

//Ждать завершение счёт1 в Т1

S4.WaitOne();

//Счет2

A.copyFrom(A.mergeSort(A.partOfVector(A, 2\*H, H), A.partOfVector(A, 3\*H, H)), 0);

//Сигнал Т1 о завершении счёт2

S3.Release();

//Копии2 l, C, MK

Monitor.Enter(EE);

int l3 = l;

Vector C3 = new Vector(N);

C3.copyFrom(C);

Matrix MK3 = new Matrix(N);

MK3.copyFrom(MK);

Monitor.Exit(EE);

//Счёт4

D.copyFrom(C3.multiplication(MK3.multiplication(MT, 2\*H, 3 \* H - 1), 2\*H, 3 \* H - 1), 2\*H);

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

{

D[i] \*= l3;

}

//Ждать завершения счёт3 в Т1

E1\_2.WaitOne();

//Счёт5

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

{

A[i] += D[i];

}

//Сигнал Т1 о завершении счет5

S1.Release();

Console.WriteLine("T3 finished.");

}

static void Func4()

{

Console.WriteLine("T4 started.");

//Ввод l, MK

l = FILLER;

MK.input();

//Сигнал Т1,Т2,Т3 о вводе

E4.Set();

//Ждать завершения ввода в Т1,Т2,Т3

E1.WaitOne();

E2.WaitOne();

E3.WaitOne();

//Копия1 В

M.WaitOne();

Vector B4 = new Vector(N);

B4.copyFrom(B);

M.ReleaseMutex();

//Счёт1

A.copyFrom(B.multiplication(MO, 3\*H, N - 1), 3\*H);

A.sort(3\*H, H);

//Сигнал Т3 о завершении счёт1

S4.Release();

//Копии2 l, C, MK

Monitor.Enter(EE);

int l4 = l;

Vector C4 = new Vector(N);

C4.copyFrom(C);

Matrix MK4 = new Matrix(N);

MK4.copyFrom(MK);

Monitor.Exit(EE);

//Счёт4

D.copyFrom(C4.multiplication(MK4.multiplication(MT, 3\*H, N - 1), 3\*H, N - 1), 3\*H);

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

{

D[i] \*= l4;

}

//Ждать завершения счёт3 в Т1

E1\_2.WaitOne();

//Счёт5

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

{

A[i] += D[i];

}

//Сигнал Т1 о завершении счет5

S1.Release();

Console.WriteLine("T4 finished.");

}

static void Main(string[] args)

{

Console.WriteLine("Main thread started.");

Thread T1 = new Thread(Func1);

Thread T2 = new Thread(Func2);

Thread T3 = new Thread(Func3);

Thread T4 = new Thread(Func4);

T1.Start();

T2.Start();

T3.Start();

T4.Start();

T1.Join();

Console.WriteLine("Main thread finished");

Console.Read();

}

}

}