МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

Кафедра СТ

Отчет

о выполнении лабораторной работы №6

«МНОГОПОТОЧНОСТЬ И ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ»

по дисциплине «Программирование под платформу .Net»

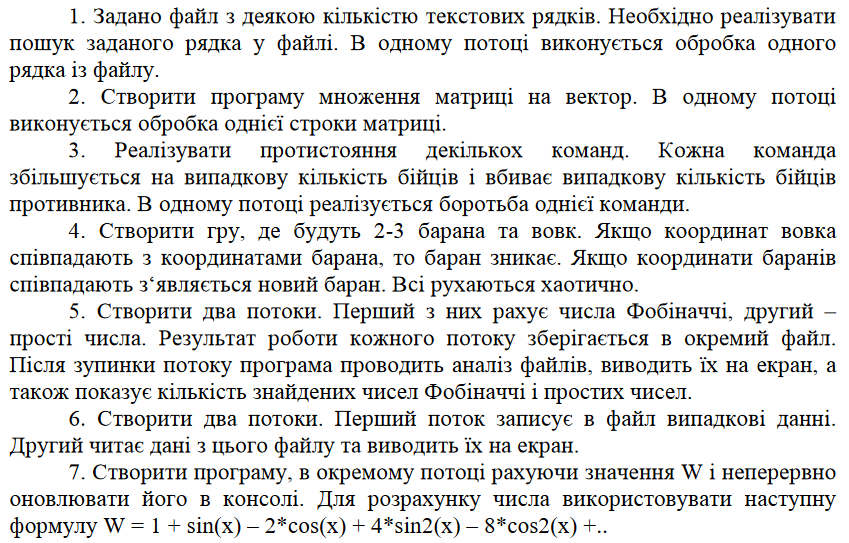
|  |  |
| --- | --- |
| Выполнил ст. гр. КН-15-2:  Антонюк М.В. | Проверила:  Жернова П.Е. |

Харьков 2018

**Цель работы**

Научиться создавать программы с использованием многопоточности. Получение навыков организации параллельных вычислений.

**Задание**

**  
Ход работы**

**Задание 1**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part1

{

string[] lines;

private static object \_syncRoot = new object();

public Part1(string path)

{

lines = File.ReadAllLines(path);

}

public void IsContains(string s)

{

Thread[] threads = new Thread[lines.Length - 1];

for(int i = 0; i < lines.Length - 1; i++)

{

threads[i] = new Thread(() =>

{

lock (\_syncRoot)

Console.Write(Search(lines[i], s) ? Environment.NewLine + "Found" : "");

});

threads[i].Start();

}

foreach(Thread t in threads)

{

t.Join();

}

}

private bool Search(string line, string s)

{

lock(\_syncRoot)

return line.Contains(s);

}

}

}

**Задание 2**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Matrix

{

double[,] matrix;

private static object \_syncRoot = new object();

public int Row { get; protected set; }

public int Column { get; protected set; }

public Matrix(int row, int column)

{

Row = row;

Column = column;

matrix = new double[row, column];

}

public Matrix Multiple(Matrix value)

{

List<Thread> list = new List<Thread>();

Matrix result = new Matrix(Row, value.Column);

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

for (int j = 0; j < value.Column; j++)

{

Thread t = new Thread(new ParameterizedThreadStart(func));

list.Add(t);

t.Start(new Part2Param(result, value, matrix, i, j));

}

foreach(Thread t in list)

{

t.Join();

}

return result;

}

public static void func(object obj)

{

Part2Param p = (Part2Param)obj;

for (int k = 0; k < p.value.Row; k++)

{

lock (\_syncRoot)

{

p.result.matrix[p.i, p.j] += p.matrix[p.i, k] \* p.value.matrix[k, p.j];

}

Console.WriteLine(p.result.matrix[p.i, p.j]);

}

}

public void Read()

{

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

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

{

Console.Write("Введите элемент [{0},{1}]: ", i + 1, j + 1);

matrix[i, j] = System.Convert.ToDouble(Console.ReadLine());

}

}

public void Print()

{

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

{

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

Console.Write("{0:f2} ", matrix[i, j]);

Console.WriteLine();

}

}

}

public class Part2Param

{

public Matrix result;

public Matrix value;

public double[,] matrix;

public int i;

public int j;

public Part2Param(Matrix result, Matrix value, double[,] matrix, int i, int j)

{

this.result = result;

this.value = value;

this.matrix = matrix;

this.i = i;

this.j = j;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

public class Part2

{

public static void MainFunction()

{

Matrix vector = new Matrix(1, 2);

Matrix matrix = new Matrix(2, 2);

Console.WriteLine("Ввод вектора");

vector.Read();

Console.WriteLine("\nВвод матрицы");

matrix.Read();

Matrix result = vector.Multiple(matrix);

Console.WriteLine("Вектор");

vector.Print();

Console.WriteLine("\nМатрица");

matrix.Print();

Console.WriteLine("\nРезультат умножения матрицы на вектор");

result.Print();

}

}

}

**Задание 3**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part3

{

private static object \_syncRoot = new object();

int team1 = 5, team2 = 5, team3 = 5;

public void MainFunction()

{

Console.Write(team1 + " " + team2 + " " + team3 + " ");

Thread.Sleep(600);

Thread teamOne = new Thread(t31);

Thread teamTwo = new Thread(t32);

Thread teamThree = new Thread(t33);

teamOne.Start();

teamTwo.Start();

teamThree.Start();

}

private void kill(int num)

{

Monitor.Enter(\_syncRoot);

int m;

switch (num)

{

case 1: team1 += new Random().Next(0, 10); break;

case 2: team2 += new Random().Next(0, 10); break;

case 3: team3 += new Random().Next(0, 10); break;

}

do

{

m = new Random().Next(1, 4); //m - команда, коорую убиваем

}

while (m == num);

int players = 0;

switch (m)

{

case 1:

players = new Random().Next(0, team1);

team1 -= players;

break;

case 2:

players = new Random().Next(0, team2);

team2 -= players;

break;

case 3:

players = new Random().Next(0, team3);

team3 -= players;

break;

}

Console.WriteLine(team1 + " " + team2 + " " + team3 + " ");

Thread.Sleep(1000);

Monitor.Exit(\_syncRoot);

}

private void t31() { for (int i = 0; i < 10; i++) kill(1); }

private void t32() { for (int i = 0; i < 10; i++) kill(2); }

private void t33() { for (int i = 0; i < 10; i++) kill(3); }

}

}

**Задание 4**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part4

{

List<Animal> rams;

Animal wolf;

Random rand;

public Part4()

{

rand = new Random();

rams = new List<Animal>();

rams.Add(new Animal(rand.Next(0, 100), rand.Next(0, 100), rand));

rams.Add(new Animal(rand.Next(0, 100), rand.Next(0, 100), rand));

rams.Add(new Animal(rand.Next(0, 100), rand.Next(0, 100), rand));

wolf = new Animal(rand.Next(0, 100), rand.Next(0, 100), rand);

}

public void Start()

{

wolf.Run();

foreach(Animal a in rams)

{

a.Run();

}

}

public void Game()

{

new Thread(() => {

int \_new = 0;

while (true)

{

if (rams.Count == 0)

{

Console.WriteLine("Баранов не осталось в живых!");

break;

}

for (int i = 0; i < rams.Count; i++)

{

if (wolf.X == rams[i].X && wolf.Y == rams[i].Y)

{

Console.WriteLine("Волк съел барана");

rams.Remove(rams[i]);

}

}

for (int i = 0; i < rams.Count; i++)

{

for (int j = i + 1; j < rams.Count; j++)

{

if (rams[i].X == rams[j].X && rams[i].Y == rams[j].Y)

{

\_new++;

Console.WriteLine("Бараны расплодились");

}

}

}

for (int i = 0; i < \_new; i++)

{

rams.Add(new Animal(rand.Next(0, 100), rand.Next(0, 100), rand));

}

\_new = 0;

}

}).Start();

}

}

public class Animal

{

Random rand;

int x;

int y;

bool isAlive;

public bool IsAlive

{

get { return isAlive; }

set { isAlive = value; }

}

public int X

{

get { return x; }

set { x = value; }

}

public int Y

{

get { return y; }

set { y = value; }

}

public Animal(int x, int y, Random rand)

{

this.rand = rand;

isAlive = true;

this.x = x;

this.y = y;

}

public void Run()

{

new Thread(() => {

while (isAlive)

{

lock (rand)

{

//Thread.Sleep(100);

x = rand.Next(0, 100);

y = rand.Next(0, 100);

}

}

}).Start();

}

}

}

**Задание 5**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part5

{

long counterFibonacci;

long counterNumber = 1;

bool isAlive = true;

private void Fibonacci()

{

new Thread(() =>

{

using (var sw = new StreamWriter("D://fibonacciLab6.txt"))

{

long a = 0;

long b = 1;

while (isAlive)

{

long temp = a;

a = b;

b = temp + b;

sw.WriteLine(b);

counterFibonacci++;

Thread.Sleep(500);

}

}

}).Start();

}

private void Numbers()

{

new Thread(() =>

{

using (var sw = new StreamWriter("D://numbersLab6.txt"))

{

while (isAlive)

{

sw.WriteLine(counterNumber++);

Thread.Sleep(500);

}

}

}).Start();

}

public void Start()

{

Numbers();

Fibonacci();

}

public void Stop()

{

Console.WriteLine("Закончить подсчеты?");

Console.ReadKey();

isAlive = false;

Thread.Sleep(1000);

}

public void Stat()

{

counterNumber = 1;

counterFibonacci = 1;

var sr1 = new StreamReader("D://numbersLab6.txt");

var sr2 = new StreamReader("D://fibonacciLab6.txt");

while(!sr1.EndOfStream && !sr2.EndOfStream)

{

Console.WriteLine((!sr1.EndOfStream ? sr1.ReadLine() : "") + "\t" + (!sr2.EndOfStream ? sr2.ReadLine() : ""));

if (!sr1.EndOfStream)

counterNumber++;

if (!sr2.EndOfStream)

counterFibonacci++;

}

Console.WriteLine("Числа :" + counterNumber + " Фиббоначи :" + counterFibonacci);

}

}

}

**Задание 6**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part6

{

static object \_syncRoot = new object();

const int WRITE = 1;

const int READ = 2;

Random rand;

public Part6()

{

rand = new Random();

}

public void MainFunction()

{

new Thread(Write).Start();

new Thread(Read).Start();

}

private void ReadWrite(int mode, int seek = 0)

{

lock (\_syncRoot)

{

switch (mode)

{

case WRITE:

File.AppendAllText("D://part6.txt", ((char)rand.Next(33, 126)).ToString());

break;

case READ:

FileStream file = new FileStream("D://part6.txt", FileMode.Open, FileAccess.Read);

file.Seek(seek, SeekOrigin.Begin);

Console.Write(Convert.ToChar(file.ReadByte()));

file.Close();

break;

}

}

}

private void Write()

{

while (true)

{

ReadWrite(WRITE);

Thread.Sleep(1000);

}

}

private void Read()

{

Thread.Sleep(500);

int i = 0;

while (true)

{

ReadWrite(READ, i++);

Thread.Sleep(1500);

}

}

}

}

**Задание 7**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab6

{

public class Part7

{

public void MainFunction()

{

new Thread(Count).Start();

}

private void Count()

{

double w = 1.0;

int x = new Random().Next(30, 210);

int j = 1;

int k = 1;

while (true)

{

w += k \* Math.Pow(Math.Sin(x), j);

k \*= 2;

Console.Clear();

Console.Write("W = " + w);

Thread.Sleep(500);

w -= k \* Math.Pow(Math.Cos(x), j);

j++;

Console.Clear();

Console.Write("W = " + w);

Thread.Sleep(500);

}

}

}

}

**Main**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Lab6

{

class Program

{

static void Main(string[] args)

{

//Console.WriteLine("~~~~~Task - 1~~~~~");

//Part1 part1 = new Part1("C:\\part1.txt");

//part1.IsContains("Его жизнь — то, что видно через занавески.");

//Console.WriteLine("");

//Console.WriteLine("~~~~~Task - 2~~~~~");

//Part2.MainFunction();

//Console.WriteLine("~~~~~Task - 3~~~~~");

//new Part3().MainFunction();

//Console.WriteLine("~~~~~Task - 4~~~~~");

//Part4 part4 = new Part4();

//part4.Start();

//part4.Game();

//Console.WriteLine("~~~~~Task - 5~~~~~");

//Part5 part5 = new Part5();

//part5.Start();

//part5.Stop();

//part5.Stat();

//Console.WriteLine("~~~~~Task - 6~~~~~");

//new Part6().MainFunction();

Console.WriteLine("~~~~~Task - 7~~~~~");

new Part7().MainFunction();

Console.ReadKey();

}

}

}

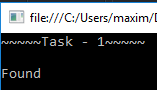


Рисунок 1 - Результат выполнения задания 1

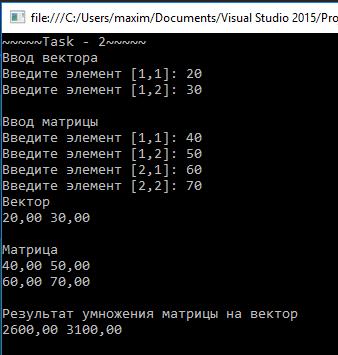


Рисунок 2 - Результат выполнения задания 2

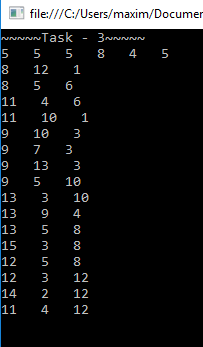


Рисунок 3 - Результат выполнения задания 3

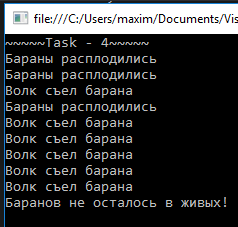


Рисунок 4 - Результат выполнения задания 4

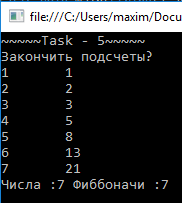
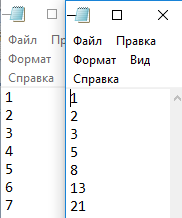
 

Рисунок 5 - Результат выполнения задания 5

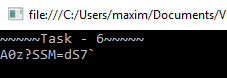
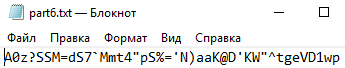
 

Рисунок 6 - Результат выполнения задания 6

Рисунок 7 - Результат выполнения задания 7

**Выводы.** В ходе выполнения лабораторной работы были изучены потоки и многопоточность. Были получены навыки использования многопоточности и выполнения параллельных вычислений. Также были освоены некоторые методы синхронизации потоков и закреплены навыки работы с текстовыми файлами.