Министерство образования и науки Российской

**Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем

и программной инженерии

**Лабораторная работа № 3**

**по дисциплине**

**«Программирование компьютерной графики»**

**По теме: «Многопоточное программирование в C#»**

Выполнила:

ст. гр. ПРИ-118

Левченко В.П

Принял:

Жигалов И.Е

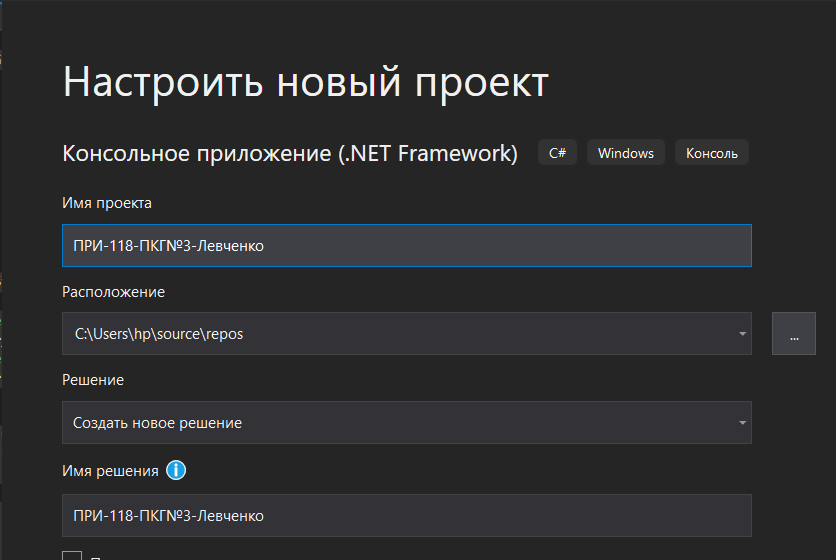
Владимир, 2021

**Цель работы**

Освоение средств организации многопоточных вычислений в C#.

**Выполнение работы**

Для начала создадим новый проект.

****Рисунок 1. Создание проекта

В первую очередь добавим метод для вывода названия проекта

static void WriteString(object \_Data)

{

System.Console.WriteLine("Название проекта:");

string str\_for\_out = System.IO.Path.GetFileName(System.Reflection.Assembly.GetExecutingAssembly().Location).ToString();

System.Console.WriteLine(str\_for\_out);

}

Далее добавляем переменные, исходя из варианта для индивидуального задания:

  
Рисунок 2. Вариант 5

static double a = 0;

static double b = 6;

static int n = 35;

static double[] h = new double[n + 1]; //для вычисления шага

static double[] x = new double[n + 1]; //для вычисления x

static double[] y = new double[n + 1]; //для вычисления y

Теперь добавим метод для вычисления шага, x и y:

static void step(object \_st)

{

double st = (double)\_st;

lock (locker)

{

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

{

h[i] = st / n;

}

}

}

static void X(object \_a)

{

double a = (double)\_a;

lock (locker)

{

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

{

x[i] = a + i \* h[i];

}

}

}

private static void Y()

{

lock (locker)

{

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

{

y[i] = 1 / ((Math.Exp(x[i]) - 4 \* x[i] + 4) - x[i] - 1) + Math.Log(5 \* Math.Tan(x[i])) - Math.Exp(7 \* Math.Sqrt(x[i])) + 0.3 \* (Math.Pow(x[i], 3) + Math.Exp(x[i]) - 1);

}

}

}

Теперь перейдём к точке входа в приложение:

Создаём потоки, в качестве параметров передаём имя выполняемой функции:

Thread th\_1 = new Thread(WriteString);

Thread stepThread = new Thread(step);

Thread xThread = new Thread(X);

Thread yThread = new Thread(Y);

Расставляем приоритеты для потоков:

th\_1.Priority = ThreadPriority.Highest; //название проекта- самый высокий

stepThread.Priority = ThreadPriority.BelowNormal; //вычисление шага - выше среднего

xThread.Priority = ThreadPriority.Normal; //вычисление х - средний

yThread.Priority = ThreadPriority.Lowest; //результат - низкий

Запускаем каждый поток:

th\_1.Start();

stepThread.Start(b - a);

xThread.Start(a);

yThread.Start();

Ждём завершения каждого потока и выводим результат вычислений:

th\_1.Join();

stepThread.Join();

xThread.Join();

yThread.Join();

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

Console.WriteLine($"Y[{i}] = {y[i]}");

Полный код:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ПРИ\_118\_ПКГ\_3\_Левченко

{

class Program

{

static void WriteString(object \_Data)

{

System.Console.WriteLine("Название проекта:");

string str\_for\_out = System.IO.Path.GetFileName(System.Reflection.Assembly.GetExecutingAssembly().Location).ToString();

System.Console.WriteLine(str\_for\_out);

}

static double a = 0;

static double b = 6;

static int n = 35;

static double[] h = new double[n + 1]; //для вычисления шага

static double[] x = new double[n + 1]; //для вычисления x

static double[] y = new double[n + 1]; //для вычисления y

static object locker = new object(); //взаимоисключающая блокировка

static void step(object \_st)

{

double st = (double)\_st;

lock (locker)

{

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

{

h[i] = st / n;

}

}

}

static void X(object \_a)

{

double a = (double)\_a;

lock (locker)

{

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

{

x[i] = a + i \* h[i];

}

}

}

private static void Y()

{

lock (locker)

{

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

{

y[i] = 1 / ((Math.Exp(x[i]) - 4 \* x[i] + 4) - x[i] - 1) + Math.Log(5 \* Math.Tan(x[i])) - Math.Exp(7 \* Math.Sqrt(x[i])) + 0.3 \* (Math.Pow(x[i], 3) + Math.Exp(x[i]) - 1);

}

}

}

static void Main(string[] args)

{

//создаём потоки, в качестве параметров передаём имя выполняемой функции

Thread th\_1 = new Thread(WriteString);

Thread stepThread = new Thread(step);

Thread xThread = new Thread(X);

Thread yThread = new Thread(Y);

//приоритеты

th\_1.Priority = ThreadPriority.Highest; //название проекта- самый высокий

stepThread.Priority = ThreadPriority.BelowNormal; //вычисление шага - выше среднего

xThread.Priority = ThreadPriority.Normal; //вычисление х - средний

yThread.Priority = ThreadPriority.Lowest; //результат - низкий

//запускаем каждый поток

th\_1.Start();

stepThread.Start(b - a);

xThread.Start(a);

yThread.Start();

th\_1.Join();

stepThread.Join();

xThread.Join();

yThread.Join();

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

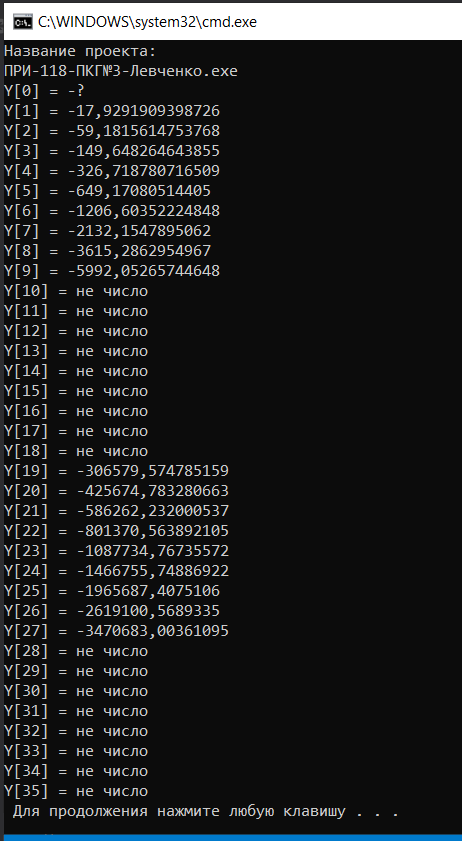
Console.WriteLine($"Y[{i}] = {y[i]}");

Console.ReadKey();

}

}

}

  
Рисунок 3. Результат

Пример из методического пособия:

static void WriteString(object \_Data)

{

//для получения строки используем преобразование типов:

// приводим переменную \_Data к типу string и записываем

// в переменную str\_for\_out

string str\_for\_out = (string)\_Data;

// теперь поток 1 тысячу раз выведит полученную строку (свой номер)

for (int i = 0; i <= 1000; i++)

Console.Write(str\_for\_out);

}

Потоки:

//создаем 4 потока, в качестве параметров передаем имя Выполняемой функции

Thread th\_1 = new Thread(WriteString);

Thread th\_2 = new Thread(WriteString);

Thread th\_3 = new Thread(WriteString);

Thread th\_4 = new Thread(WriteString);

Приоритет поток:

//расставляем приоритеты для потоков

th\_1.Priority = ThreadPriority.Highest; // самый высокий

th\_2.Priority = ThreadPriority.BelowNormal; // выше среднего

th\_3.Priority = ThreadPriority.Normal; // средний

th\_4.Priority = ThreadPriority.Lowest; // низкий

Запуск потоков:

// запускаем каждый поток, в качестве параметра передаем номер потока

th\_1.Start("1");

th\_2.Start("2");

th\_3.Start("3");

th\_4.Start("4");

Ожидание завершение потока:

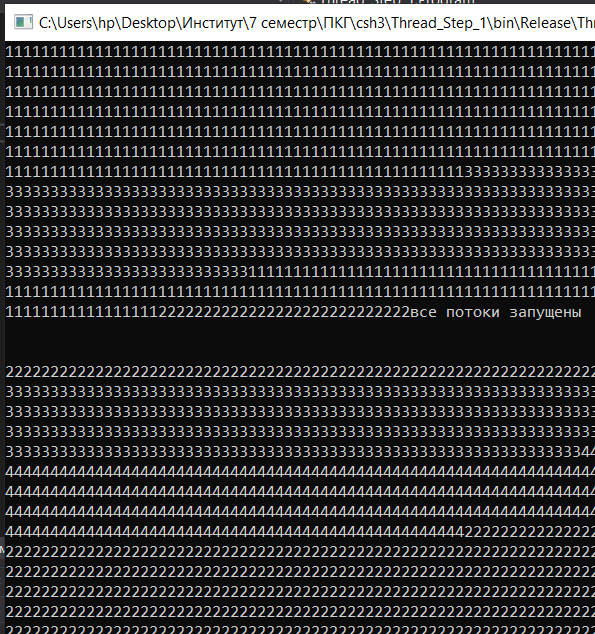
//Ждем заврешения каждого потока

th\_1.Join();

th\_2.Join();

th\_3.Join();

th\_4.Join();

  
Рисунок 4. Результат запуска

**Вывод**

Освоены средства организации многопоточных вычислений в C#.