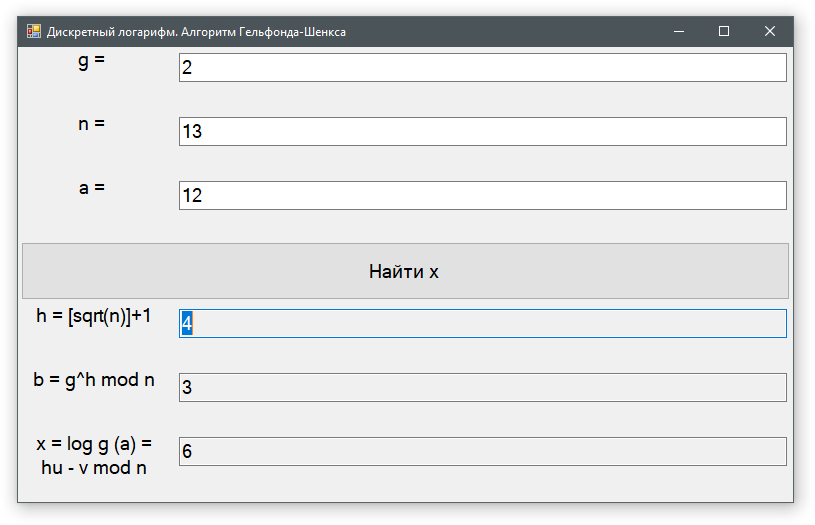
Лабораторная работа № 1. Метод Шенкса.

Десятов Александр, 09-641



using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Алгоритм\_Гельфонда\_Шенкса

{

public partial class FormGelfond : Form

{

public FormGelfond()

{

InitializeComponent();

}

private void btn\_calc\_Click(object sender, EventArgs e)

{

// Очистка полей

tB\_h.Text = "";

tB\_b.Text = "";

tB\_x.Text = "";

// Запрет на редактирование до окончания подсчета

tB\_g.ReadOnly = true;

tB\_n.ReadOnly = true;

tB\_a.ReadOnly = true;

// Запрет нажимать кнопку

btn\_calc.Enabled = false;

// Вызвать функцию подсчета X в отдельном потоке

bW\_calc.RunWorkerAsync();

}

// Функция подсчета X

private void bW\_calc\_DoWork(object sender, DoWorkEventArgs e)

{

// Считывание g

if (BigInteger.TryParse(tB\_g.Text, out BigInteger g))

{

// Проверка на положительность

if (g <= 0)

{

MessageBox.Show("g должно быть положительным!");

return;

}

}

else

{

MessageBox.Show("Не удалось считать g!");

return;

}

// Считывание n.

if (BigInteger.TryParse(tB\_n.Text, out BigInteger n))

{

// Проверка на положительность

if (n <= 0)

{

MessageBox.Show("n должно быть положительным!");

return;

}

}

else

{

MessageBox.Show("Не удалось считать n!");

return;

}

// Считывание a

if (BigInteger.TryParse(tB\_a.Text, out BigInteger a))

{

// Проверка на положительность

if (a < 0)

{

MessageBox.Show("a должно быть неотрицательным!");

return;

}

}

else

{

MessageBox.Show("Не удалось считать a!");

return;

}

// Алгоритм Гельфонда-Шенкса

BigInteger h = Sqrt(n) + 1;

tB\_h.Text = h.ToString();

BigInteger b = BigInteger.ModPow(g, h, n);

tB\_b.Text = b.ToString();

// Построение таблицы

// Шаги Ребенка

List<BigInteger> babySteps = new List<BigInteger>();

babySteps.Add((a \* g) % n);// первый элемент

for (BigInteger v = 2; v <= h; v++)

{

babySteps.Add((babySteps[babySteps.Count - 1] \* g) % n);

}

// Получить исходные индексы отсортированного списка babySteps, чтобы потом найти x

int[] argsort\_baby = Enumerable.Range(0, babySteps.Count).ToArray();

Array.Sort<int>(argsort\_baby, (aa, bb) => babySteps[aa].CompareTo(babySteps[bb]));

// Сортировка, чтобы потом проводить бинарный поиск

babySteps.Sort();

// Шаги Великана

BigInteger prevGiantStep = b % n; // первый элемент

// Поиск одинаковых элементов

for (BigInteger u = 1; u <= h; u++)

{

// Индекс в отсортированном giantSteps

int index\_in\_sort = babySteps.BinarySearch(prevGiantStep);

// Если число prevGiantStep найдено babySteps

if (index\_in\_sort >= 0)

{

BigInteger v = argsort\_baby[index\_in\_sort] + 1; // так как отсчет от 0

BigInteger x = (h\*u - v) % n;

tB\_x.Text = x.ToString();

return;

}

prevGiantStep = (prevGiantStep \* b) % n;

}

MessageBox.Show("x не найден! :(");

}

// Событие завершения функции bW\_calc\_DoWork

private void bW\_calc\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

// Разрешение на редактирование входных данных

tB\_g.ReadOnly = false;

tB\_n.ReadOnly = false;

tB\_a.ReadOnly = false;

// Разрешение нажимать кнопку

btn\_calc.Enabled = true;

}

// Событие закрытия формы

private void FormGelfond\_FormClosing(object sender, FormClosingEventArgs e)

{

// Принудительно завершить поток, если он не завершен

if (bW\_calc.IsBusy)

{

bW\_calc.WorkerSupportsCancellation = true;

bW\_calc.CancelAsync();

}

}

// Sqrt для BigInteger есть в java, но нет .Net

public BigInteger Sqrt(BigInteger n)

{

if (n == 0) return 0;

if (n > 0)

{

int bitLength = Convert.ToInt32(Math.Ceiling(BigInteger.Log(n, 2)));

BigInteger root = BigInteger.One << (bitLength >> 1);

while (!IsSqrt(n, root))

{

root += n / root;

root >>= 1;

}

return root;

}

throw new ArithmeticException("NaN");

}

private bool IsSqrt(BigInteger n, BigInteger root)

{

BigInteger lowerBound = root \* root;

return n >= lowerBound && n <= lowerBound + (root << 1);

}

}

}