Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

Институт Вычислительной математики и информационных технологий

ОТЧЕТ

по научно-исследовательской работе (производственной) практике

Обучающийся Гусев Виталий Евгеньевич гр.09-335 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО студента) (Группа) (Подпись)

Научный руководитель:

доцент КСАИТ Мубараков Б.Г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Руководитель практики от кафедры:

ст.преподаватель КСАИТ Тихонова О.О. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Оценка за практику \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Подпись)

Дата сдачи отчета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Казань – 2025

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc197019104)

[1. Разработка тестов для базовых и модифицированных алгоритмов дискретного логарифмирования 4](#_Toc197019105)

[ЗАКЛЮЧЕНИЕ 6](#_Toc197019106)

[СПИСОК ЛИТЕРАТУРЫ 8](#_Toc197019107)

[ПРИЛОЖЕНИЯ 9](#_Toc197019108)

ВВЕДЕНИЕ

Производственная практика проходила на кафедре системного анализа и информационных технологий Института вычислительной математики и информационных технологий КФУ с 21 марта 2025 года по 25 мая 2025 года.

Целью практики является исследование и реализация тестов для базовых и модифицированных алгоритмов дискретного логарифмирования с экспоненциальной и субэкспоненциальной сложностью.

Задачами практики являются:

1) разработать тесты для базовых и модифицированных методов дискретного логарифмирования,

2) программно реализовать тесты для базовых и модифицированных методов дискретного логарифмирования,

3) провести эксперименты на реализованных тестах для базовых и модифицированных методов дискретного логарифмирования.

1. Разработка тестов для базовых и модифицированных алгоритмов дискретного логарифмирования

В процессе практики были реализованы и исследованы тесты для базовых и модифицированных алгоритмов дискретного логарифмирования на языке программирования C# на .NET8 в Windows Forms (рисунок 1). Для тестирования данных алгоритмов был использован генератор параметров Диффи-Хеллмана и возведение числа в степень по модулю [1]. Также для тестирования данных алгоритмов был использован замер времени выполнения алгоритма и количество затраченной памяти на выполнение алгоритма.



Рисунок 1 - Реализованная программа

Были реализованы и исследованы тесты для базовых и модифицированных экспоненциальных алгоритмов дискретного логарифмирования: алгоритм Шенкса [2], алгоритм Полига-Хеллмана [3], ро-метод Полларда [4], а также тесты для базовых и модифицированных субэкспоненциальных алгоритмов дискретного логарифмирования: алгоритм Адлемана [5], алгоритм COS [6], решето числового поля [7].

Разработанная программа позволяет вносить в текстовые поля необходимые значения параметров возведения чисел в степень по модулю: g, a, p, A [8, 9], либо целых чисел N для разложения на простые множители [10] и выводить результат вычисления. В процессе практики были проведены тесты алгоритмов на различных параметрах с замером времени и затраченной памятью вычисления алгоритмов (рисунок 2).



Рисунок 2 - Вычисление модифицированных алгоритмов

2. Тестирование базового и модифицированного алгоритма Шенкса

Были проведены тесты базового и модифицированного алгоритма «Шаг младенца - шаг великана» - в теории групп детерминированный алгоритм дискретного логарифмирования в мультипликативной группе кольца вычетов по модулю простого числа. Начальный алгоритм был предложен советским математиком Александром Гельфондом в 1962 году и Дэниелом Шенксом в 1972 году. Метод теоретически упрощает решение задачи дискретного логарифмирования, на вычислительной сложности которой построены многие криптосистемы с открытым ключом. Относится к методам встречи посередине. Это был один из первых методов, который показал, что задача вычисления дискретного логарифма может быть решена значительно быстрее, чем методом перебора. Идея алгоритма состоит в выборе оптимального соотношения времени и памяти, а именно в усовершенствованном поиске показателя степени.

Пусть задано сравнение , необходимо найти натуральное число , удовлетворяющее данному сравнению.

Начальный алгоритм реализован следующим образом:

1) сначала берутся два целых числа и , такие, что . Как правило ;

2) вычисляются два ряда чисел:

,

.

Все вычисления проводятся по модулю ;

3) идёт поиск таких и , для которых выполняется равенство . То есть ищется во втором ряду такое число, которое присутствует и в первом ряду. Запоминаются показатели степени и , при которых данные числа получались;

4) в результате работы алгоритма неизвестная степень вычисляется по формуле .

Была реализована модификация алгоритма, состоящая в распараллеливании 2 и 3 шага алгоритма. На 2 шаге алгоритма параллельно вычисляются два ряда чисел. На 3 шаге был сделан параллельный поиск результата с начала и с конца ряда.

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и - 16 битные числа, а параметр - 8 битное число:

Таблица 1- Результаты тестов базового алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 9347 | 99 | 14629 | 6331 | 4 | 278312 |
| 11873 | 107 | 14401 | 2419 | 1 | 270368 |
| 11922 | 78 | 26399 | 22034 | 1 | 417888 |
| 606 | 75 | 4973 | 3597 | 2 | 139664 |
| 8173 | 49 | 14143 | 8610 | 1 | 263168 |
| 32464 | 14 | 32717 | 20677 | 1 | 263168 |
| 6229 | 118 | 32257 | 5301 | 2 | 444096 |
| 7921 | 106 | 16703 | 100 | 2 | 296064 |
| 16108 | 101 | 31271 | 6732 | 7 | 5862760 |
| 5901 | 85 | 7019 | 3639 | 2 | 164480 |

Таблица 2 - Результаты тестов модифицированного алгоритма Шенкса

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g | a | p | A | Время (мс) | Память (байт) |
| 9347 | 99 | 14629 | 6331 | 47 | 278592 |
| 11873 | 107 | 14401 | 2419 | 27 | 270368 |
| 11922 | 78 | 26399 | 22034 | 23 | 409408 |
| 606 | 75 | 4973 | 3597 | 24 | 139808 |
| 8173 | 49 | 14143 | 8610 | 23 | 271392 |
| 32464 | 14 | 32717 | 20677 | 35 | 466992 |
| 6229 | 118 | 32257 | 5301 | 34 | 452320 |
| 7921 | 106 | 16703 | 100 | 21 | 296064 |
| 16108 | 101 | 31271 | 6732 | 30 | 452320 |
| 5901 | 85 | 7019 | 3639 | 32 | 172704 |

В результате тестов среднее время выполнения базового алгоритма Шенкса равно 2.3 мс, а модифицированного алгоритма Шенкса равно 29.6 мс. Средняя затраченная память базового алгоритма Шенкса равна 839996.8 байт, а модифицированного алгоритма Шенкса равна 320996.8 байт. Базовый алгоритм показал лучше результаты в скорости, а модифицированный алгоритм показал лучше результаты в затраченной памяти.

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и - 16 битные числа, а параметр - 16 битное число:

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и – 32 битные числа, а параметр - 8 битное число:

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и – 32 битные числа, а параметр - 16 битное число:

Были сгенерированы параметры и проведены тесты базового (таблица 1) и модифицированного (таблица 2) алгоритма Шенкса, где , и – 32 битные числа, а параметр - 32 битное число:

ЗАКЛЮЧЕНИЕ

В результате практики были реализованы и исследованы тесты для базовых и модифицированных алгоритмов дискретного логарифмирования.

За период практики были приобретены следующие компетенции (таблица N):

|  |  |  |
| --- | --- | --- |
| Компетенция | Расшифровка компетенции | Описание приобретенных знаний, умений и навыков |
| УК-1 | Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий | Получена способность осуществлять критический анализ проблемных ситуаций на основе системного подхода при разработке тестов дискретного логарифмирования |
| УК-6 | Способен определять и реализовывать приоритеты собственной деятельности и способы ее совершенствования на основе самооценки | Приобретена способность реализовывать приоритеты собственной деятельности при разработке тестов дискретного логарифмирования на основе самооценки |
| ОПК-1 | Способен находить, формулировать и решать актуальные проблемы прикладной математики, фундаментальной информатики и информационных технологий | Получен навык находить, формулировать и решать актуальные проблемы прикладной математики при разработке тестов дискретного логарифмирования |
| ОПК-2 | Способен применять компьютерные/суперкомпьютерные методы, современное программное обеспечение (в том числе отечественного производства) для решения задач профессиональной деятельности | Получена способность применять компьютерные методы, современное программное обеспечение для решения задач профессиональной деятельности при разработке тестов дискретного логарифмирования |
| ОПК-3 | Способен проводить анализ математических моделей, создавать инновационные методы решения прикладных задач профессиональной деятельности в области информатики и математического моделирования | Приобретена способность проводить анализ математических моделей, создавать инновационные методы решения прикладных задач профессиональной деятельности в области информатики при разработке тестов дискретного логарифмирования |
| ОПК-4 | Способен оптимальным образом комбинировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности | Приобретён навык оптимальным образом комбинировать существующие информационно-коммуникационные технологии для решения задач в области профессиональной деятельности с учетом требований информационной безопасности при разработке тестов дискретного логарифмирования |

На основе тестов есть возможность сделать вывод, что определённые модифицированные алгоритмы дискретного логарифмирования показали лучше показатели в скорости выполнения или в затраченной памяти по сравнению с базовыми алгоритмами.

СПИСОК ЛИТЕРАТУРЫ

1) Теоретический минимум и алгоритмы цифровой подписи / Молдовян Н. А. – Текст: непосредственный // Книжный Дом «ЛИБРОКОМ», 2010. — С. 304.

2) The infrastructure of a real quadratic field and its applications. Proceedings of the Number Theory Conference. / D. Shanks. – Текст: непосредственный // University of Colorado, Boulder, 1972. — С. 217-224.

3) An Improved Algorithm for Computing Logarithms Over GF(p) and its Cryptographic Significance (англ.) / S. C. Pohlig and M. E. Hellman. - Текст: непосредственный // IEEE Transactions on Information Theory. — 1978. — Vol. 1, no. 24. — С. 106-110.

4) Theorems on factorization and primality testing / Pollard J.M. - Текст: непосредственный // Mathematical Proceedings of the Cambridge Philosophical Society. — 1974. — Т. 76, вып. 03. — С. 521–528.

5) A subexponential algorithm for discrete logarithms over all finite fields / Adleman L. M., Demarrais J. - Текст: непосредственный // Mathematics of computation. — 1993.

6) Теоретико-числовые алгоритмы в криптографии. / Василенко О.Н. - Текст: непосредственный // N— М.: МЦНМО, 2003. — C. 328.

7) Методы факторизации натуральных чисел. / Ишмухаметов Ш. Т. - Текст: непосредственный // — Казань: Казан. ун.. — 2011. — C. 190.

8) Applied Cryptography: Protocols, Algorithms, and Source Code in C. / Schneier, Bruce – Текст: непосредственный // Second Edition. — 2nd. — Wiley, 1996.

9) Методы факторизации натуральных чисел. / Ишмухаметов Ш. Т. - Текст: непосредственный // — Казань: Казан. ун.. — 2011. — C. 10.

10) Методы факторизации натуральных чисел. / Ишмухаметов Ш. Т. - Текст: непосредственный // — Казань: Казан. ун.. — 2011. — C. 52.

ПРИЛОЖЕНИЯ

using DiscreteLogarithm.ExponentialAlgorithms;

using DiscreteLogarithm.MathFunctionsForCalculation;

using DiscreteLogarithm.ModifiedExponentialAlgorithms;

using DiscreteLogarithm.ModifiedSubExponentialAlgorithms;

using DiscreteLogarithm.SubExponentialAlgorithms;

using System.Diagnostics;

using System.Numerics;

using System.Security.Cryptography;

namespace DiscreteLogarithmCore

{

public partial class Form1 : Form

{

MathFunctions mathFunctions;

Shenks shenks;

ModifiedShenks modifiedShenks;

PoligHellman poligHellman;

ModifiedPoligHellman modifiedPoligHellman;

RoPollard roPollard;

ModifiedRoPollard modifiedRoPollard;

Adleman adleman;

ModifiedAdleman modifiedAdleman;

COS cos;

ModifiedCOS modifiedCOS;

GNFS gNFS;

ModifiedGNFS modifiedGNFS;

public Form1()

{

InitializeComponent();

mathFunctions = new MathFunctions();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

BigInteger N;

bool theValuesAreCorrect = true;

gNFS = new GNFS();

gNFS.CheckingTheInputValues(textBox1.Text, label28, ref theValuesAreCorrect, out N);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

try

{

gNFS.CalculateGNFS(N, label28);

}

catch (Exception ex)

{

label28.Text = "Error";

}

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label28.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button2\_Click(object sender, EventArgs e)

{

BigInteger g;

BigInteger A;

BigInteger p;

bool theValuesAreCorrect = true;

shenks = new Shenks();

shenks.CheckingTheInputValues(textBox2.Text, textBox3.Text, textBox4.Text, label15, ref theValuesAreCorrect, out g, out A, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

shenks.CalculateShenks(g, A, p, label15);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label15.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button3\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

poligHellman = new PoligHellman();

poligHellman.CheckingTheInputValues(textBox7.Text, textBox6.Text, textBox5.Text, label16, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

poligHellman.CalculatePoligHellman(a, b, p, label16);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label16.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button6\_Click(object sender, EventArgs e)

{

BigInteger N;

bool theValuesAreCorrect = true;

roPollard = new RoPollard();

roPollard.CheckingTheInputValues(textBox14.Text, textBox22, ref theValuesAreCorrect, out N);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

roPollard.CalculateRoPollard(N, textBox22);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

textBox22.Text += $"\n t = {stopwatch.ElapsedMilliseconds} мс \n{consumedInBytes} байт";

}

private void button7\_Click(object sender, EventArgs e)

{

BigInteger a = mathFunctions.Generate\_a(8);

List<BigInteger> p\_g = mathFunctions.Generate\_p\_g(16);

BigInteger A = mathFunctions.ExponentiationModulo(p\_g[1], a, p\_g[0]);

textBox16.Text = a.ToString();

textBox15.Text = p\_g[0].ToString();

textBox17.Text = p\_g[1].ToString();

textBox18.Text = A.ToString();

}

private void button8\_Click(object sender, EventArgs e)

{

BigInteger g;

BigInteger a;

BigInteger p;

bool theValuesAreCorrect = true;

mathFunctions.CheckingTheInputValues(textBox21.Text, textBox20.Text, textBox19.Text, label35, ref theValuesAreCorrect, out g, out a, out p);

if (!theValuesAreCorrect)

{

return;

}

mathFunctions.ExponentiationModuloWin(g, a, p, label35);

}

private void button4\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

adleman = new Adleman();

adleman.CheckingTheInputValues(textBox10.Text, textBox9.Text, textBox8.Text, label20, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

adleman.CalculateAdleman(a, b, p, label20);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label20.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button5\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

cos = new COS();

cos.CheckingTheInputValues(textBox13.Text, textBox12.Text, textBox11.Text, label24, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

cos.CalculateCOS(a, b, p, label24);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label24.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

async private void button9\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

modifiedShenks = new ModifiedShenks();

modifiedShenks.CheckingTheInputValues(textBox2.Text, textBox3.Text, textBox4.Text, label40, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

await modifiedShenks.CalculateModifiedShenksAsync(a, b, p, label40);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label40.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button10\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

modifiedPoligHellman = new ModifiedPoligHellman();

modifiedPoligHellman.CheckingTheInputValues(textBox7.Text, textBox6.Text, textBox5.Text, label41, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

modifiedPoligHellman.CalculatePoligHellman(a, b, p, label41);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label41.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button11\_Click(object sender, EventArgs e)

{

BigInteger N;

bool theValuesAreCorrect = true;

modifiedRoPollard = new ModifiedRoPollard();

modifiedRoPollard.CheckingTheInputValues(textBox14.Text, textBox23, ref theValuesAreCorrect, out N);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

modifiedRoPollard.CalculateRoPollard(N, textBox23);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

textBox23.Text += $"\n t = {stopwatch.ElapsedMilliseconds} мс \n{consumedInBytes} байт";

}

private void button12\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

modifiedAdleman = new ModifiedAdleman();

modifiedAdleman.CheckingTheInputValues(textBox10.Text, textBox9.Text, textBox8.Text, label43, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

modifiedAdleman.CalculateAdleman(a, b, p, label43);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label43.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button13\_Click(object sender, EventArgs e)

{

BigInteger a;

BigInteger b;

BigInteger p;

bool theValuesAreCorrect = true;

modifiedCOS = new ModifiedCOS();

modifiedCOS.CheckingTheInputValues(textBox13.Text, textBox12.Text, textBox11.Text, label44, ref theValuesAreCorrect, out a, out b, out p);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

modifiedCOS.CalculateCOS(a, b, p, label44);

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label44.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button14\_Click(object sender, EventArgs e)

{

BigInteger N;

bool theValuesAreCorrect = true;

modifiedGNFS = new ModifiedGNFS();

modifiedGNFS.CheckingTheInputValues(textBox1.Text, label45, ref theValuesAreCorrect, out N);

if (!theValuesAreCorrect)

{

return;

}

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

long before = GC.GetTotalMemory(false);

try

{

modifiedGNFS.CalculateGNFS(N, label45);

}

catch (Exception ex)

{

label45.Text = "Error";

}

long after = GC.GetTotalMemory(false);

int consumedInBytes = (int)(after - before);

consumedInBytes = consumedInBytes > 0 ? consumedInBytes : -consumedInBytes;

stopwatch.Stop();

label45.Text += $"\nt = {stopwatch.ElapsedMilliseconds} мс\n{consumedInBytes} байт";

}

private void button15\_Click(object sender, EventArgs e)

{

textBox2.Text = textBox17.Text;

textBox3.Text = textBox18.Text;

textBox4.Text = textBox15.Text;

}

private void button16\_Click(object sender, EventArgs e)

{

textBox7.Text = textBox17.Text;

textBox6.Text = textBox18.Text;

textBox5.Text = textBox15.Text;

}

private void button17\_Click(object sender, EventArgs e)

{

int byteCount = 8;

BigInteger generatedNumber = new BigInteger(RandomNumberGenerator.GetBytes(byteCount));

while (generatedNumber % 2 == 0 ||

generatedNumber % 3 == 0 ||

generatedNumber % 5 == 0 ||

generatedNumber % 7 == 0)

{

generatedNumber = new BigInteger(RandomNumberGenerator.GetBytes(byteCount));

}

generatedNumber \*= generatedNumber.Sign;

textBox14.Text = generatedNumber.ToString();

}

private void button18\_Click(object sender, EventArgs e)

{

textBox10.Text = textBox17.Text;

textBox9.Text = textBox18.Text;

textBox8.Text = textBox15.Text;

}

}

}