Министерство науки и высшего образования РФ Федеральное государственное бюджетное образовательное учреждение Высшего образования

«Ярославский государственный технический университет» Кафедра «Информационные системы и технологии»

Отчет по лабораторной работе защищен С оценкой _____ Руководитель, доцент кафедры «Информационные системы и технологии» к.пед.н _____ А. В. Никитенко «07» 11 2023

БАЗОВЫЕ АЛГОРИТМЫ

Лабораторная работа №5 по курсу «Криптографические методы защиты информации»

ЯГТУ 09.03.02

Отчет выполнил Студент гр. ЦИСБ-24

> — И.А. Лавров

 $\ll 18 \gg \underline{12} \ 2023$

В лабораторной работе разработать набор подпрограмм, реализующих базовые алгоритмы:

- 1. возведение в степень по модулю (a^x mod m);
- 2. вычисление наибольшего общего делителя (НОД (a, b));
- 3. вычисление мультипликативной инверсии (x^{-1}) mod m);
- 4. проверки чисел на простоту (тест Ферма, тест Миллера-Рабина);
- 5. генерации больших простых чисел.

```
Программный код:
using System.Numerics;
using System.Security.Cryptography;
namespace WinFormsApp1
   public partial class Form1 : Form
        public Form1()
            InitializeComponent();
        }
            private void Form1_Load(object sender, EventArgs e)
        {
        }
        private void button1_Click(object sender, EventArgs e)
        }
             static BigInteger ModularExponentiation(BigInteger a, BigInteger e,
BigInteger m)
                   BigInteger result = 1;
                   while (e != 0)
                          if (e % 2 == 1)
                                result = (result * a) % m;
                          e = e / 2;
                          a = (a * a) % m;
                   return result;
             }
             private void textBox2_TextChanged(object sender, EventArgs e)
        {
        }
        private void textBox1_TextChanged(object sender, EventArgs e)
        }
        private void textBox1_TextChanged_1(object sender, EventArgs e)
```

```
{
        }
        private void textBox3_TextChanged(object sender, EventArgs e)
        }
             private void label1_Click(object sender, EventArgs e)
             }
             private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
             }
             private void label2_Click(object sender, EventArgs e)
             }
             private void label3_Click(object sender, EventArgs e)
             private void button1_Click_1(object sender, EventArgs e)
             }
             private void label4_Click(object sender, EventArgs e)
             private void menuStrip1_ItemClicked(object sender,
ToolStripItemClickedEventArgs e)
             }
             private void label1_Click_1(object sender, EventArgs e)
             }
             private void button1_Click_2(object sender, EventArgs e)
                   try
                          BigInteger a = Convert.ToInt64(textBox1.Text);
                          BigInteger x = Convert.ToInt64(textBox2.Text);
                          BigInteger m = Convert.ToInt64(textBox3.Text);
                          BigInteger result = ModularExponentiation(a, x, m);
                          textBox4.Text = $"Результат: {result}";
                   catch (FormatException)
                          MessageBox.Show("Введите корректные числовые значения в
поля", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                   catch (OverflowException)
```

```
MessageBox.Show("Введенное число слишком большое", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                   catch (Exception ex)
                          MessageBox.Show($"Произошла ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
             }
             private void label7_Click(object sender, EventArgs e)
             }
             private void label5_Click(object sender, EventArgs e)
             }
             private void textBox6_TextChanged(object sender, EventArgs e)
             private void label6_Click(object sender, EventArgs e)
             }
             private void textBox5_TextChanged(object sender, EventArgs e)
             }
             private void textBox7_TextChanged(object sender, EventArgs e)
             private void button3_Click(object sender, EventArgs e)
                    try
                    {
                          if (BigInteger.TryParse(textBox7.Text, out BigInteger a) &&
BigInteger.TryParse(textBox6.Text, out BigInteger b))
                          {
                                 BigInteger gcd = GCD(a, b);
                                 textBox5.Text = \"HOД({a}, {b}) = {gcd}";
                          }
                          else
                                 MessageBox.Show("Введите корректные числовые
значения", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                   catch (Exception ex)
                          MessageBox.Show($"Произошла ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
             private BigInteger GCD(BigInteger a, BigInteger b)
                   while (b != 0)
                          BigInteger temp = b;
                          b = a % b;
```

```
a = temp;
                    }
                    return a;
             }
             private void label13_Click(object sender, EventArgs e)
             private BigInteger ModInverse(BigInteger a, BigInteger m)
                    BigInteger m0 = m;
                    BigInteger x0 = 0;
                    BigInteger x1 = 1;
                    if (m == 1)
                          return 0;
                    while (a > 1)
                    {
                          BigInteger q = a / m;
                          BigInteger t = m;
                          m = a % m;
                          a = t;
                          t = x0;
                          x0 = x1 - q * x0;
                          x1 = t;
                    }
                    while (x1 < 0)
                          x1 += m0;
                    return x1;
             }
             private void button4_Click(object sender, EventArgs e)
                    try
                    {
                          if (BigInteger.TryParse(textBox13.Text, out BigInteger x) &&
BigInteger.TryParse(textBox12.Text, out BigInteger m))
                                 BigInteger result = ModInverse(x, m);
                                 textBox11.Text = result.ToString();
                          }
                          else
                          {
                                 MessageBox.Show("Введите корректные числа в textBox13
и textBox12");
                          }
                    catch (Exception ex)
                          MessageBox.Show($"Произошла ошибка: Для данного числа не
существует обратного");
             }
             private void label9_Click(object sender, EventArgs e)
             }
             private void label8_Click(object sender, EventArgs e)
             }
```

```
private void textBox9_TextChanged(object sender, EventArgs e)
             }
             private void textBox8_TextChanged(object sender, EventArgs e)
             }
             private void button2_Click(object sender, EventArgs e)
                    try
                    {
                          if (BigInteger.TryParse(textBox9.Text, out BigInteger n) &&
BigInteger.TryParse(textBox10.Text, out BigInteger a))
                                 bool isPrime = FermatTest(n, a);
                                 if (isPrime)
                                       textBox8.Text = "True";
                                 else
                                       textBox8.Text = "False";
                          }
                          else
                                 MessageBox.Show("Введите корректные числа в textBox9
и textBox10");
                    }
                    catch (Exception ex)
                          MessageBox.Show($"Произошла ошибка: {ex.Message}");
             }
             private bool FermatTest(BigInteger n, BigInteger a)
                    if (a <= 1 || n <= 1)
                          return false;
                    BigInteger result = BigInteger.ModPow(a, n - 1, n);
                    return result == 1;
             }
             private void button5_Click(object sender, EventArgs e)
                    try
                    {
                          if (BigInteger.TryParse(textBox15.Text, out BigInteger n))
                                 bool isProbablyPrime = MillerRabinTest(n);
                                 if (isProbablyPrime)
                                       textBox14.Text = $"{n} вероятно простое число
по тесту Миллера-Рабина";
                                 else
                                       textBox14.Text = $"{n} не является простым
числом по тесту Миллера-Рабина";
                          else
                                 MessageBox.Show("Введите корректное число в
textBox15");
                          }
                   catch (Exception ex)
                          MessageBox.Show($"Произошла ошибка: {ex.Message}");
                    }
```

}

```
private bool MillerRabinTest(BigInteger n)
                    if (n <= 1)
                          return false;
                    BigInteger d = n - 1;
                    int s = 0;
                    while (d % 2 == 0)
                    {
                           d /= 2;
                           s++;
                    int numberOfIterations = 20;
                    Random random = new Random();
                    for (int i = 0; i < numberOfIterations; i++)</pre>
                           BigInteger a = RandomBigInteger(2, n - 2, random);
                           BigInteger x = BigInteger.ModPow(a, d, n);
                           if (x == 1 || x == n - 1)
                                 continue;
                           for (int r = 1; r < s; r++)
                                 x = BigInteger.ModPow(x, 2, n);
                                 if(x == 1)
                                        return false;
                                 if (x == n - 1)
                                        break;
                           }
                           if (x != n - 1)
                                 return false;
                    }
                    return true;
             private BigInteger RandomBigInteger(BigInteger min, BigInteger max,
Random random)
                    byte[] bytes = new byte[max.ToByteArray().Length];
                    random.NextBytes(bytes);
                    BigInteger value = new BigInteger(bytes);
                    return BigInteger.Remainder(BigInteger.Add(BigInteger.Abs(value),
min), max - min) + min;
             private void button6_Click(object sender, EventArgs e)
                    try
                           int bitLength = GetRandomBitLength();
                           BigInteger primeNumber = GeneratePrimeNumber(bitLength);
                           textBox16.Text = primeNumber.ToString();
                    catch (Exception ex)
                           MessageBox.Show($"Произошла ошибка: {ex.Message}");
             private int GetRandomBitLength()
                    Random random = new Random();
                    return random.Next(100, 512);
```

```
private BigInteger GeneratePrimeNumber(int bitLength)
                                 using (RandomNumberGenerator rng = RandomNumberGenerator.Create())
                                            while (true)
                                                       byte[] bytes = new byte[(bitLength + 7) / 8];
                                                        rng.GetBytes(bytes);
                                                       BigInteger candidate = new BigInteger(bytes);
                                                        candidate |= BigInteger.One << (bitLength - 1);</pre>
                                                        if (MillerRabinTest(candidate))
                                                                   return candidate;
                                            }
                                 }
                     }
          }
 📵 Возведение в степень по модулю
                                                                                            Модул
65
                                                342
                                                                                                                                            Возведение в степень по модулю Вычисление НОД Инверсия Тест Ферма Тест Миллера-Рабина Генерация большого простого числа
 Результат
14
                                                                                               Результат
                                                       Введите число а (основание)
                                                       3
                                                                                               Результат: 14
                                                       Введите число х (степень)
Алгоритмы быстрого
Если применять наивный способ возведения в степень - прост
                                                       Введите число n (модуль)
степени. Несмотря на всю мощь современных компьютеров, т
большие, чем стандартные 64-битные целые. Например, в прос
используем как значение показателя степени по-умолчанию, н
Чтобы оперировать подобными показателями требуются алго
                                                       65
позволяет свести к минимуму число операций умножения. Одн
степеней не подходит из-за ограничений по ресурсам.
Поэтому в данном калькуляторе для вычисления степени мы
                                                         Вычислить
памяти. Вариант этого алгоритма описан в той же статье, одна
старшего бита до младшего. В нашем случае это несколько не заранее не представляем, сколько разрядов они занимают в п
Двоичный алгоритм возведе
Поэтому алгоритм обрабатывает двоичное представление пок
```

Рисунок 1 – Результат возведения числа в степень по модулю

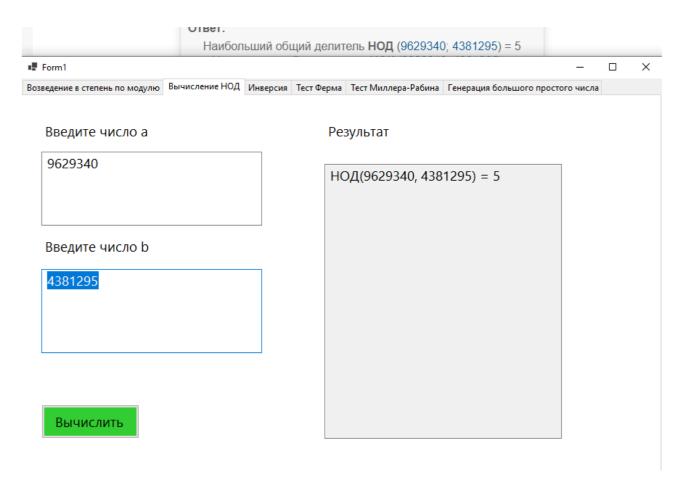


Рисунок 2 – Вычисление НОД

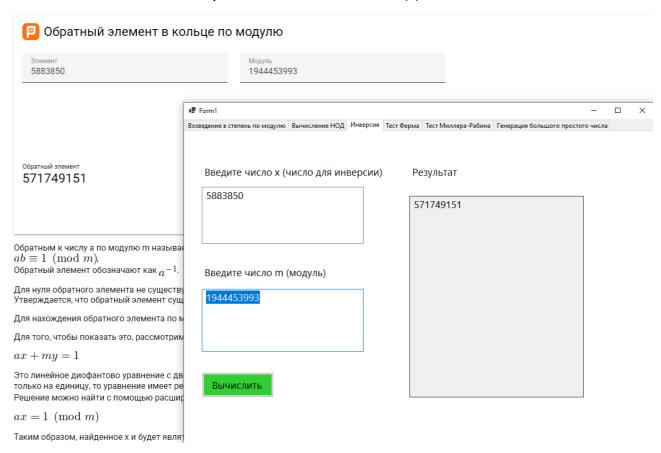


Рисунок 3 – Нахождение обратного числа по модулю

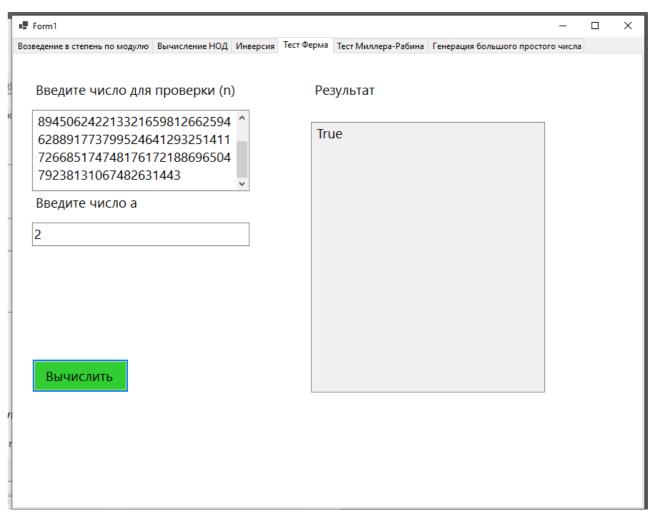


Рисунок 4 – Проверка первого числа по тесту Ферма

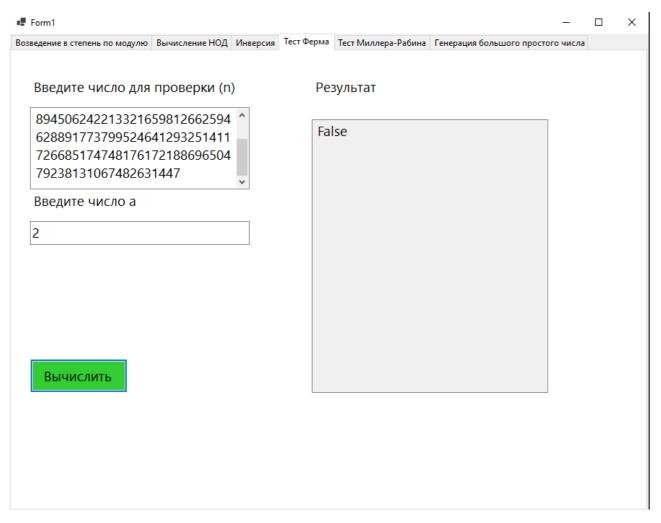


Рисунок 5 – Проверка второго числа по тесту Ферма

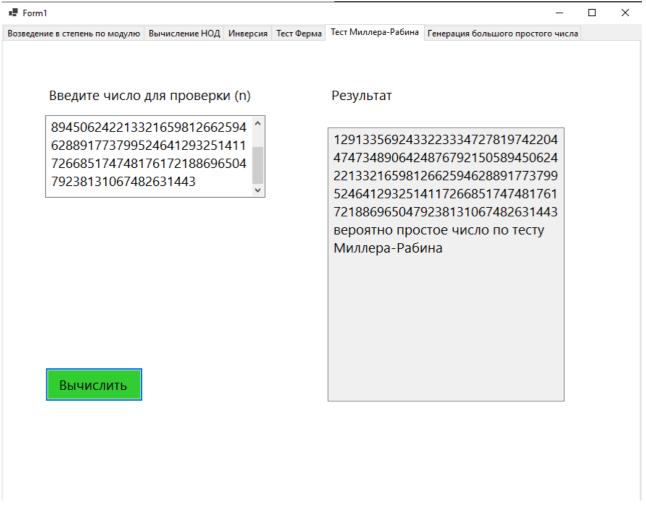


Рисунок 6 – Проверка числа тестом Миллера-Рабина

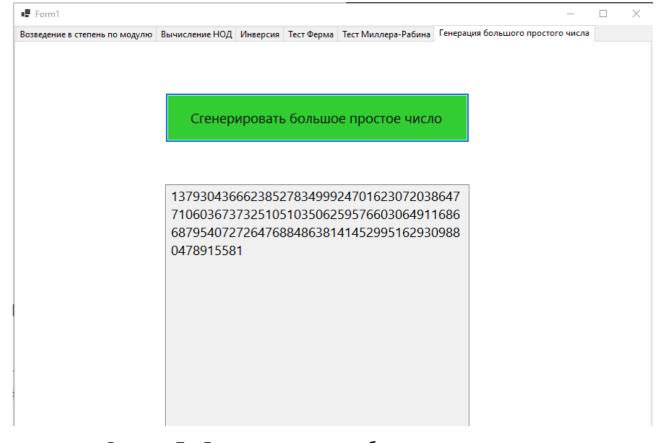


Рисунок 7 – Генерация простого большого числа

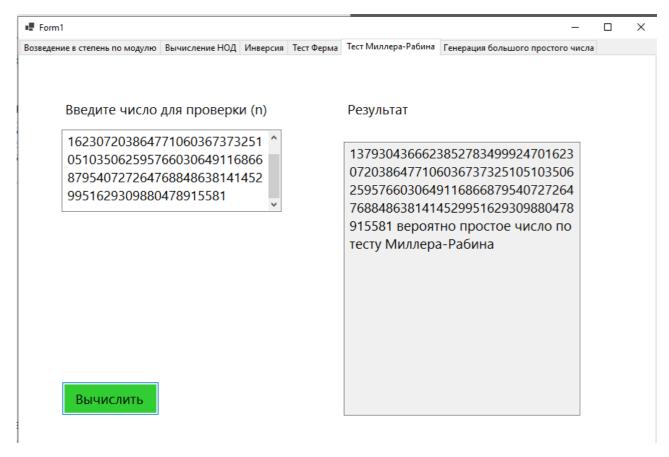


Рисунок 8 – Проверка сгенерированного числа тестом Миллера-Рабина