

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

Лабораторная работа №6

по дисциплине: «Теоретико-числовые методы в криптографии»

на тему: «Квадратичное сравнение.»

Выполнил обучающийся гр. ВКБ42

Михайлов Александр Сергеевич

          Проверила:

Ст. пр. Артамонова Е.А.

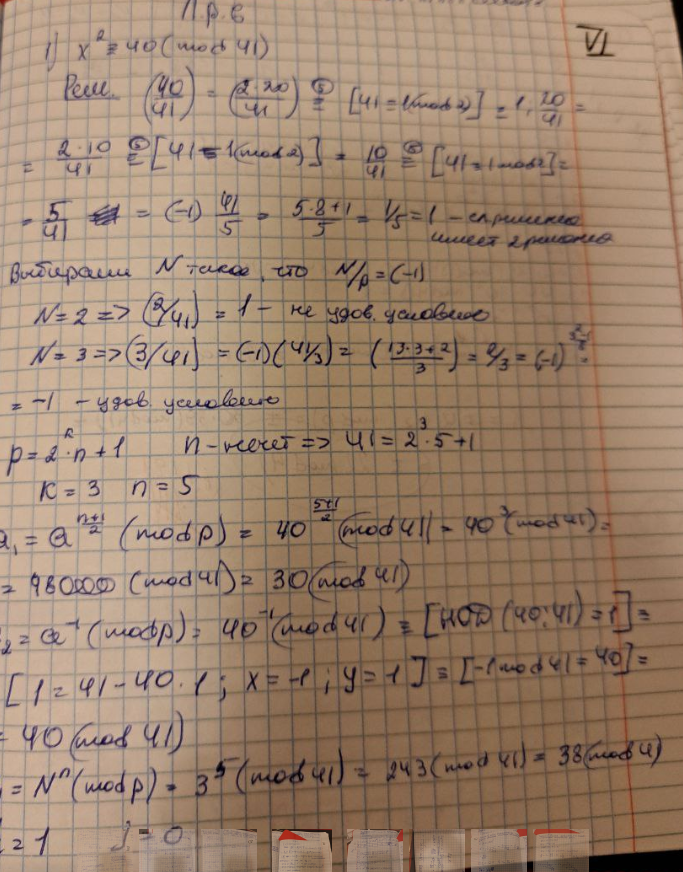
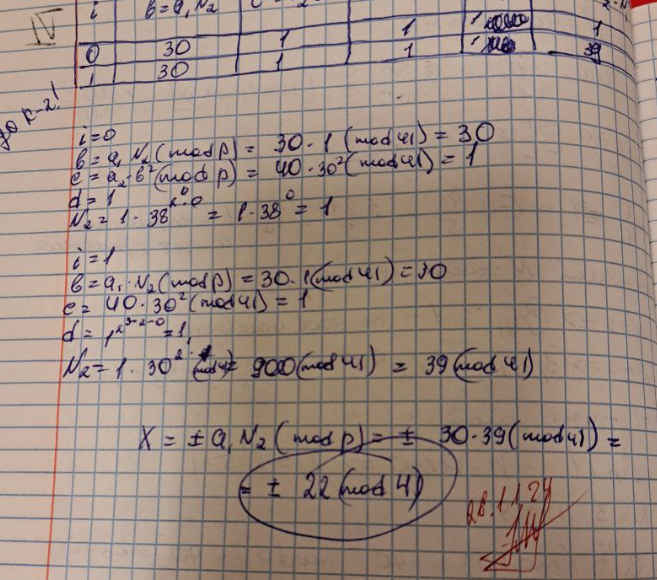
Ростов-на-Дону

2024

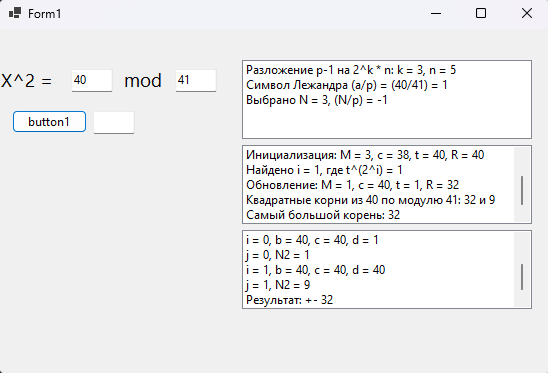
Лабораторная работа №6

«Квадратичное сравнение.»

Вариант №20

**Ручная реализация.** 

**Входные данные.**

****

**Листинг программы.**

using System.Windows.Forms;

namespace КвадратичноеСравнениеПоПростомуМодулю

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

listBox1.Items.Clear();

listBox2.Items.Clear();

listBox3.Items.Clear();

int a = int.Parse(textBox1.Text);

int p = int.Parse(textBox2.Text);

// Проверка на простое число

if (!IsPrime(p) || p == 2)

{

listBox2.Items.Add("Модуль должен быть простым числом и не равным 2.");

return;

}

// Проверка на форму P = 2^k \* n + 1

int k = 0;

int n = p - 1;

while (n % 2 == 0)

{

n /= 2;

k++;

}

if (n % 2 == 0)

{

listBox2.Items.Add("Модуль не соответствует форме P = 2^k \* n + 1, где n - нечетное.");

return;

}

listBox1.Items.Add($"Разложение p-1 на 2^k \* n: k = {k}, n = {n}");

// Вычисление символа Лежандра

int legendreSymbol = LegendreSymbol(a, p);

listBox1.Items.Add($"Символ Лежандра (a/p) = ({a}/{p}) = {legendreSymbol}");

if (legendreSymbol != 1)

{

listBox2.Items.Add("Число a не является квадратичным вычетом по модулю p.");

return;

}

// Нахождение N такого, что (N/p) = -1

int N = FindMinNonResidue(p);

listBox1.Items.Add($"Выбрано N = {N}, (N/p) = -1");

// Нахождение квадратного корня методом Чиполлы

int[] roots = TonelliShanks(a, p, N);

if (roots != null)

{

int maxRoot = Math.Max(roots[0], roots[1]);

listBox2.Items.Add($"Квадратные корни из {a} по модулю {p}: {roots[0]} и {roots[1]}");

listBox2.Items.Add($"Самый большой корень: {maxRoot}");

// Вычисление сравнения первой степени

SolveFirstDegreeComparison(a, p, N, maxRoot);

}

else

{

listBox2.Items.Add("Не удалось найти квадратные корни.");

}

}

private bool IsPrime(int n)

{

if (n <= 1) return false;

if (n == 2 || n == 3) return true;

if (n % 2 == 0 || n % 3 == 0) return false;

for (int i = 5; i \* i <= n; i += 6)

{

if (n % i == 0 || n % (i + 2) == 0) return false;

}

return true;

}

private int LegendreSymbol(int a, int p)

{

return ModPow(a, (p - 1) / 2, p);

}

private int FindMinNonResidue(int p)

{

for (int N = 2; N < p; N++)

{

if (LegendreSymbol(N, p) == p - 1)

{

return N;

}

}

return -1; // Этот случай не должен произойти, так как p - простое число

}

private int[] TonelliShanks(int a, int p, int N)

{

int Q = p - 1;

int S = 0;

while (Q % 2 == 0)

{

Q /= 2;

S++;

}

listBox2.Items.Add($"Разложение p-1 на Q и S: Q = {Q}, S = {S}");

int z = N;

listBox2.Items.Add($"Выбрано z = {z}, (z/p) = -1");

int M = S;

int c = ModPow(z, Q, p);

int t = ModPow(a, Q, p);

int R = ModPow(a, (Q + 1) / 2, p);

listBox2.Items.Add($"Инициализация: M = {M}, c = {c}, t = {t}, R = {R}");

while (t != 1)

{

int i = 0;

int t2i = t;

while (t2i != 1)

{

t2i = (t2i \* t2i) % p;

i++;

}

listBox2.Items.Add($"Найдено i = {i}, где t^(2^i) = 1");

int b = ModPow(c, 1 << (M - i - 1), p);

M = i;

c = (b \* b) % p;

t = (t \* c) % p;

R = (R \* b) % p;

listBox2.Items.Add($"Обновление: M = {M}, c = {c}, t = {t}, R = {R}");

}

return new int[] { R, p - R };

}

private int ModPow(int baseValue, int exponent, int modulus)

{

int result = 1;

baseValue = baseValue % modulus;

while (exponent > 0)

{

if ((exponent & 1) == 1)

{

result = (result \* baseValue) % modulus;

}

exponent >>= 1;

baseValue = (baseValue \* baseValue) % modulus;

}

return result;

}

private void SolveFirstDegreeComparison(int a, int p, int N, int maxRoot)

{

int k = 0;

int n = p - 1;

while (n % 2 == 0)

{

n /= 2;

k++;

}

listBox3.Items.Add($"Разложение p-1 на 2^k \* n: k = {k}, n = {n}");

// Вычисление a1 и a2

int a1 = ModPow(a, (n + 1) / 2, p);

int a2 = ModPow(a, p - 2, p); // a^(-1) mod p

listBox3.Items.Add($"a1 = {a1}, a2 = {a2}");

// Вычисление N1 и N2

int N1 = ModPow(N, n, p);

int N2 = 1;

int j = 0;

listBox3.Items.Add($"Инициализация: N1 = {N1}, N2 = {N2}, j = {j}");

// Основной цикл

for (int i = 0; i < k - 1; i++)

{

int b = (a1 \* N2) % p;

int c = (a2 \* ModPow(b, 2, p)) % p;

int d = ModPow(c, (int)Math.Pow(2, k - 2 - i), p);

listBox3.Items.Add($"i = {i}, b = {b}, c = {c}, d = {d}");

if (d == 1)

{

j = 0;

}

else if (d == p - 1)

{

j = 1;

}

N2 = (N2 \* ModPow(N1, 2 \* j, p)) % p;

listBox3.Items.Add($"j = {j}, N2 = {N2}");

}

// Результат

int result1 = (a1 \* N2) % p;

int result2 = p - result1;

listBox3.Items.Add($"Результат: +- {maxRoot}");

}

}

}

**Вывод по работе:**

В ходе выполнения лабораторной работы были изучены лекционный материал и алгоритм решения сравнений второй степени.