Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| *институт* |
| Кафедра Прикладной математики и компьютерной безопасности |
| *кафедра* |

**ЛАБОРАТОРНАЯ РАБОТА №1**

|  |
| --- |
| **Базовые алгоритмы теории чисел** |
| *тема* |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель | | |  |  |  | В.И.Вайнштейн |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ15-01 №031508683 | |  |  |  | М.С.Димаксян |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы, фамилия* |

Красноярск 2019

**Задание:**

Написать и отладить набор подпрограмм, реализующих базовые алгоритмы, используемые в изученных криптосистемах с открытым ключом: возведение в степень по модулю (), вычисление наибольшего общего делителя (), вычисление инверсии ().

**Теория:**

Теорема. Пусть и – два целых положительных числа. Тогда существуют целые (не обязательно положительные) числа и , такие, что

|  |  |
| --- | --- |
|  | (1) |

Обобщенный алгоритм Евклида служит для отыскания наибольшего общего делителя целых чисел , () и , , удовлетворяющих выражению (1).

**Исходный код:**

**Функции интерфейса:**

private void f1\_1\_buttonDegCount\_Click(object sender, EventArgs e)

{

Decimal res = BinaryModPow(f1\_1\_numericNumber.Value, Math.Abs(Convert.ToInt64(f1\_1\_numericDeg.Value)), f1\_1\_numericMod.Value);

if (f1\_1\_numericDeg.Value < 0) //Пользователь ввел отрицательную степень

{

res = Foo(res, f1\_1\_numericMod.Value);

if (res == 0) //Обратный элемент не существует

f1\_1\_textBoxResult.Text = "-";

else

f1\_1\_textBoxResult.Text = Convert.ToString(res); //Обратный элемент найден

}

else

{

f1\_1\_textBoxResult.Text = Convert.ToString(res);

}

}

private void f1\_2\_buttonCountGCD\_LCM\_Click(object sender, EventArgs e)

{

f1\_2\_textBoxGCD.Text = Convert.ToString(GCD(f1\_2\_numericNumberA.Value, f1\_2\_numericNumberB.Value, out Decimal x, out Decimal y));

Decimal myLCM = LCM(f1\_2\_numericNumberA.Value, f1\_2\_numericNumberB.Value);

if (myLCM != 0)

f1\_2\_textBoxLCM.Text = Convert.ToString(LCM(f1\_2\_numericNumberA.Value, f1\_2\_numericNumberB.Value));

else

{

if (f1\_2\_numericNumberA.Value == f1\_2\_numericNumberB.Value)

f1\_2\_textBoxLCM.Text = f1\_2\_textBoxGCD.Text;

else

f1\_2\_textBoxLCM.Text = "overflow";

}

}

private void f1\_3\_buttonCountReverse\_Click(object sender, EventArgs e)

{

Decimal res = Foo(f1\_3\_numericNumber.Value, f1\_3\_numericMod.Value);

if (res != 0)

f1\_3\_textBoxResult.Text = Convert.ToString(res);

else

f1\_3\_textBoxResult.Text = "-";

}

**Функция бинарного возведения в степень:**

Decimal BinaryModPow(Decimal Number, Int64 Deg, Decimal Mod)

{

Decimal Result = 1;

Decimal Bit = Number;

if (Bit < 0) Bit = (Number + Mod) % Mod;

else

Bit = Number % Mod;

while (Deg > 0)

{

if ((Deg % 2) == 1)

{

Result \*= Bit;

Result %= Mod;

}

Deg >>= 1;

if (Deg > 0)

{

Bit \*= Bit;

Bit %= Mod;

}

}

return Result;

}

**Функция нахождения обратного элемента в кольце или поле:**

public static Decimal Foo(Decimal element, Decimal Mod)

{

if (element < 0) element = (element + Mod) % Mod;

Decimal x, y;

Decimal g = GCD(element, Mod, out x, out y);

if (g == -1)

{

g = 1;

x = -x;

y = -y;

}

//Обратный элемент не существует, если кольцо не является полем (модуль не является степенью простого числа) и элемент не взаимно прост с модулем

//Возвращает 0 в случае, если обратный элемент не существует

if (g != 1) x = 0;

//Приведем результат в положительный вид по модулю

x = x % Mod;

if (x < 0) x = (x + Mod) % Mod;

return x;

}

**Функция нахождения НОД:**

public static Decimal GCD(Decimal a, Decimal b, out Decimal x, out Decimal y)

{

if (a == 0)

{

x = 0;

y = 1;

return b;

}

Decimal x1, y1;

Decimal d = GCD(b % a, a, out x1, out y1);

x = y1 - (IntegerDivision(b,a)) \* x1;

y = x1;

return d;

}

**Функция нахождения НОК по НОД:**

Decimal LCM(Decimal a, Decimal b)

{

if (a<b)

{

Decimal tmp = a;

a = b;

b = tmp;

}

if ( a / GCD(a, b, out Decimal x, out Decimal y) < 281474976710656 && b < 281474976710656)

return a / GCD(a, b, out x, out y) \* b;

else

return 0;

}

**Вспомогательная функция целочисленного деления для типа Decimal:**

static Decimal IntegerDivision (Decimal x, Decimal y)

{

//Производим целочисленное деление x/y

decimal celoe = Decimal.Round(x / y);

if (celoe > 0 && x / celoe < y) celoe--;

return celoe;

}

**Результат работы программы:**

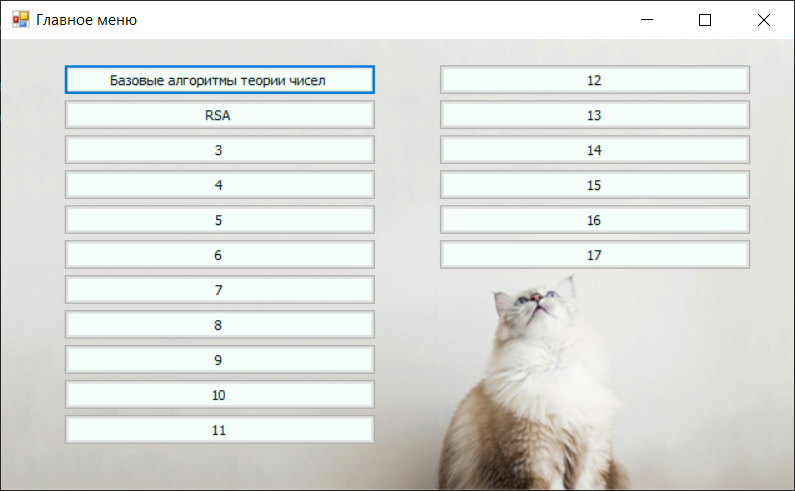


Рисунок 1. Главное меню программы.

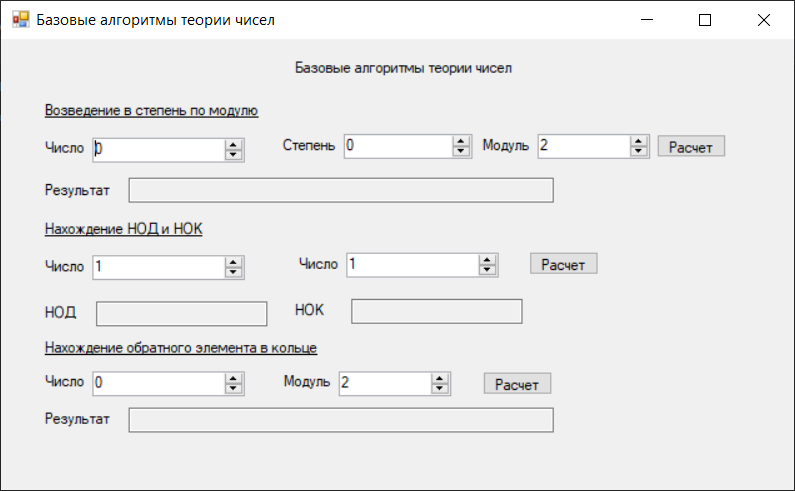


Рисунок 2. Окно базовых алгоритмов.

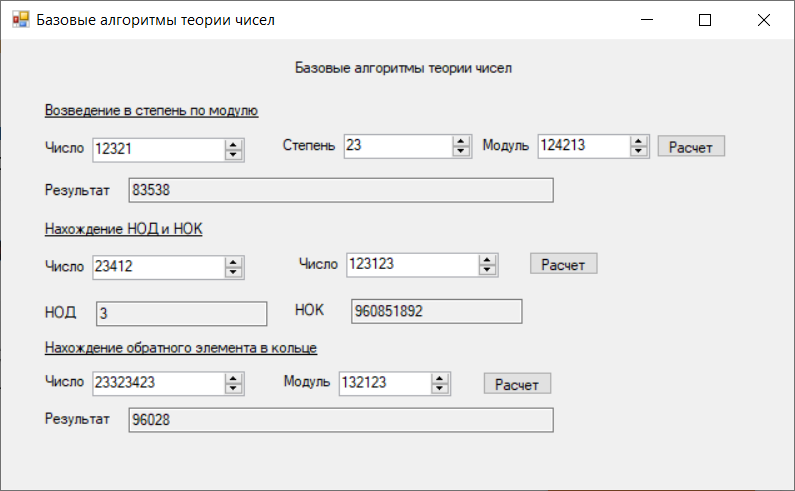


Рисунок 3. Проверка работы алгоритма на случайных примерах.

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с базовыми алгоритмами теории чисел, реализовала на практике программу по нахождению требуемых функций на языке C#. Данные навыки я могу применить при реализации криптосистем в дальнейших лабораторных работах.