Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Криптографические протоколы**

Работу выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д. Н. Баева

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Цель работы:** реализовать программный продукт решения сравнений первой степени с указанием всех промежуточных шагов вычисления (текущее значение коэффициентов в расширенном алгоритме Евклида), программный продукт так же должен реализовывать возможность того, что сравнение не имеет решений или имеет больше одного решения. В первом случае сообщать пользователю с пояснением, во втором строить все возможные решения.

**Ход работы:**

Сравнение с одним неизвестным *x* имеет вид ,где *m∈ N, m>1*. При *n = 1* сравнение называется сравнением первой степени.Любое сравнение первой степени с одним неизвестным *x* можно привести к виду , где .

Для сравнений первой степени возможно следующее количество решений:

1. единственное решение,
2. несколько решений,
3. не имеет решений.

**Теорема:** Для того, чтобы сравнение имело хотя бы одно решение, необходимо и достаточно, чтобы число b делилось на НОД(a,m).

**Пример:**

1)Сравнение   имеет решение, так как 6 делится на 3=НОД(9,12).

2)Сравнениене имеет решений, так как НОД(6,12)=6 , а 9 не делится на 6.

**Теорема:** Пусть сравнение разрешимо и d=НОД(a,m). Тогда множество решений сравнения состоит из d классов по модулю m, а именно, если  - одно из решений, то все другие решения – это

*.*

**Пример:**

1)Сравнение  имеет ровно три решения, так как НОД(9,12)=3. Эти решения:  , .

2) Сравнение имеет единственное решение  =7, так как НОД(11,15)=1.

Рассмотрим способы решения сравнений первой степени (также известных как линейные сравнения).

Пусть НОД(*a,m*)=1. Тогда решение сравнения можно искать по алгоритму Евклида. Действительно, используя расширенный алгоритм Евклида, представим число 1 в виде линейной комбинации чисел *a* и *m*: 1=*aq+mr*. Умножим обе части этого равенства на b, получим: , то естьи *bq* - решение сравнения.

Другой способ решения является использование теоремы Эйлера. Пусть, снова, НОД(*a,b*)=1. Применяем теорему Эйлера: . Умножим обе части сравнения на b: . Переписывая последнее выражение в виде: . Получаем, что  - решение сравнения.

Для реализации был выбран способ решения сравнений через расширенный алгоритм Евклида.

Сперва была реализована функция extended\_gcd(a,b)– расширенный алгоритм Евклида. Эта функция находит наибольший общий делитель для двух целых чисел a и b. Возвращает НОД(a, b) и коэффициенты и , такие, что В процессе работы используются следующие переменные: — счетчик итераций и — коэффициенты, которые обновляются на каждом шаге. Промежуточные значения выводятся на экран. Функция представлена на рисунке 1.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 1 – Расширенный алгоритм Евклида.

Далее следует функция solve\_linear\_congruence(a, b, m), которая решает сравнения первого порядка. Сначала находится НОД() и коэффициенты и с помощью расширенного алгоритма Евклида. Затем проверяется разрешимость сравнения: если не делится на НОД), то сравнение не имеет решения. Если разрешимо, вычисляется начальное значение x0 по формуле обратного элемента. После этого возвращается список всех возможных решений по модулю по соответствующей формуле. Фрагмент кода представлен на рисунке 2.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 2 – Решение сравнения.

Пользователь вводит коэффициенты и . Программа вычисляет решения сравнения или сообщает, что решений нет.

**Листинг программы**

Файл LW\_1.py

# решение сравнений первого порядка через расширенный алгоритм Евклида

def extended\_gcd(a, b):

"""

Расширенный алгоритм Евклида.

Возвращает НОД(a, b) и коэффициенты x, y такие, что ax + by = НОД(a, b).

"""

i = 1

# инициализация исходных значений

x, xx, y, yy = 1, 0, 0, 1

while b != 0:

# частное q от деления a на b

q = a // b

# обновление значений a и b так, чтобы a стало равным b, а b — остатку от деления a на b

# вывод промежуточных значений

print(f"Шаг {i}: a={a}, b={b}, x={x}, y={y}")

a, b = b, a % b

# поиск обратных элементов в кольце вычетов

x, xx = xx, x - xx \* q

y, yy = yy, y - yy \* q

i = i + 1

return a, x, y

def solve\_linear\_congruence(a, b, m):

"""

Решает сравнение ax ≡ b (mod m).

Возвращает либо решение, либо сообщение о его отсутствии.

"""

# возвращаем НОД(a, m) и коэффициенты x и y

gcd, x, y = extended\_gcd(a, m)

# проверка разрешимости сравнения

if b % gcd != 0:

return "Сравнение не имеет решения."

else:

# вычисление начальное значение x0

x0 = (x \* (b // gcd)) % m

# список solutions, в котором будут храниться все возможные решения

solutions = [(x0 + k \* (m // gcd)) % m for k in range(gcd)]

return solutions

a = int(input("Введите коэффициент а: "))

b = int(input("Введите коэффициент b: "))

m = int(input("Введите коэффициент m: "))

# получение списка решений

solutions = solve\_linear\_congruence(a, b, m)

if isinstance(solutions, str):

print(solutions)

else:

print(f"Решения сравнения {a}x ≡ {b} (mod {m}): {solutions}")