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 №5**

**Дисциплина: Криптографические протоколы**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_К.В.Стасюк

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

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

**Цель работы:** № 9 Реализовать (p-1) метод Полларда разложения на множители и РО(Rho) метод Полларда разложения на множители. Показать результаты для различных чисел

**Ход работы:**

Эта программа реализует два метода факторизации целых чисел: метод

p−1 Полларда и метод Rho . Оба метода используются для поиска множителей большого составного числа.

**Метод p−1 Полларда:**

Этот метод основан на следующей идее: если p - простое число, а a — целое число, то для всех целых j выполняется (a^j-1)mod p = 0. Из этого следует, что (a^j-1) делится на p. Метод p−1 Полларда применяет это свойство для поиска делителя числа n.

Программа начинает с выбора случайного начального значения *a*=2. Затем она итеративно возмещает *a* до степени *j*, где *j* изменяется от 2 до 99. Для этого используется операция возведения в степень по модулю: **a = pow(a, j, n)**. Затем находится, остаток от деления (a^j−1) на *n* и выполняется поиск наименьшего общего делителя *d*, который не равен 1 или *n*. Если такой делитель найден, он возвращается, иначе возвращается **None**.

**Метод Rho:**

Алгоритм Полларда "Rho" (Pollard's Rho) - это вероятностный алгоритм для факторизации целых чисел. Он основан на методе Флойда циклического обнаружения и используется для нахождения нетривиальных делителей больших составных чисел. Давайте подробно разберем его шаги:

1. **Вспомогательная функция f(x)**:

Это функция, которая принимает целое число x и возвращает результат вычисления выражения (x^2 + 1) mod n, где n - число, которое мы пытаемся факторизовать.

1. **Начальные значения x и y**:

Мы начинаем с двух случайно выбранных начальных значений x и y, оба равных 2.

1. **Поиск x и y**:

Мы последовательно вычисляем значения x и y с использованием функции f(x) и движемся по цепочке значений до тех пор, пока не встретим нетривиальный делитель n. Мы делаем это, вычисляя x = f(x) и y = f(f(y)).

1. **Нахождение НОД**:

После того как мы вычислили достаточное количество значений x и y, мы вычисляем НОД разности модулей (|x - y|) и n, где НОД - наибольший общий делитель.

1. **Проверка НОД**:

Если полученный НОД равен 1, то мы повторяем шаги 3 и 4, изменяя начальные значения x и y. Если НОД не равен 1, то это является нетривиальным делителем n, и мы можем остановить алгоритм.

1. **Возврат результата**:

**Результаты выполнения программы:**

Можно заметить, что методы находят разные делители, но оба-верные. Также первый алгоритм учитывает, что делителем числа не может быть само число, а второй – нет.

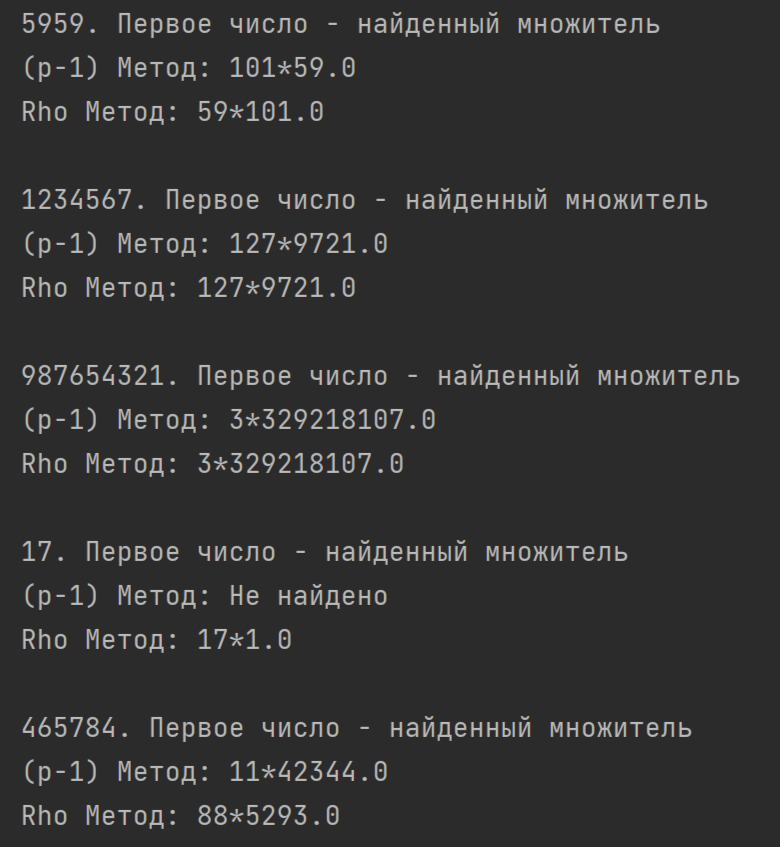


Рисунок 1 – Результат выполнения программы.

**Листинг программы**

import math  
import random  
  
#НОД  
def gcd(a, b):  
 while b:  
 a, b = b, a % b  
 return a  
  
#метод p-1  
def pollards\_p\_minus\_1(n):  
 a = 2  
 for j in range(2, 100):  
 #возводим a^j mod n  
 a = pow(a, j, n)  
 #пробуем найти нод  
 d = gcd(a - 1, n)  
 #возращаем если нетривиален  
 if 1 < d < n:  
 return d  
 return None  
  
#метод rho  
def pollards\_rho(n):  
 def f(x):  
 #вспомогательная функция = x^2+1  
 return (x \*\* 2 + 1) % n  
 #нач значение=2  
 x, y, d = 2, 2, 1  
 #ищем x y  
 while d == 1:  
 x = f(x)  
 y = f(f(y))  
 #нод  
 d = gcd(abs(x - y), n)  
 #возращаем делитель если не 1  
 return d  
  
#вывод  
def factorize\_and\_show\_results(number):  
 print(f"{number}. Первое число - найденный множитель")  
  
 # (p-1)  
 p\_minus\_1\_result = pollards\_p\_minus\_1(number)  
 if p\_minus\_1\_result:  
 print(f"(p-1) Метод: {p\_minus\_1\_result}\*{number/p\_minus\_1\_result}")  
 else:  
 print("(p-1) Метод: Не найдено")  
  
 # Rho  
 rho\_result = pollards\_rho(number)  
 if rho\_result:  
 print(f"Rho Метод: {rho\_result}\*{number/rho\_result}")  
 else:  
 print("Rho Метод: Не найдено")  
  
 print()  
  
# Примеры использования для различных чисел  
numbers\_to\_factorize = [5959, 1234567, 987654321, 17,465784]  
for num in numbers\_to\_factorize:  
 factorize\_and\_show\_results(num)