

# **Отчёт по лабораторной работе №5**

## **Математические основы защиты информации и информационной безопасности**

**Вероятностные алгоритмы проверки чисел на простоту**

Выполнил: Мануэл Марсия Педру,  
НФИмд-02-25, 1032255503

# Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6

## Список иллюстраций

2.1	Реализация вычисления символа Якоб . . . . .	7
2.2	Тест Соловея-Штрассена . . . . .	7
2.3	Тест Миллера-Рабина . . . . .	8
2.4	Проверка чисел на простоту . . . . .	8

## Список таблиц

# 1 Цель работы

Изучить шифры простой замены и научиться их реализовывать.

## 2 Выполнение лабораторной работы

#2.1 Реализация вычисления символа Якоби Символ Якоби — теоретико-числовая функция двух аргументов, введённая К. Якоби в 1837 году. Является квадратичным характером в кольце вычетов. Символ Якоби обобщает символ Лежандра на все нечётные числа, большие единицы. Символ Кронекера — Якоби, в свою очередь, обобщает символ Якоби на все целые числа, но в практических задачах символ Якоби играет гораздо более важную роль, чем символ Кронекера — Якоби.

#2.2 Реализация алгоритма, реализующего тест Ферма Тест простоты Ферма в теории чисел — это тест простоты натурального числа  $n$ , основанный на малой теореме Ферма.

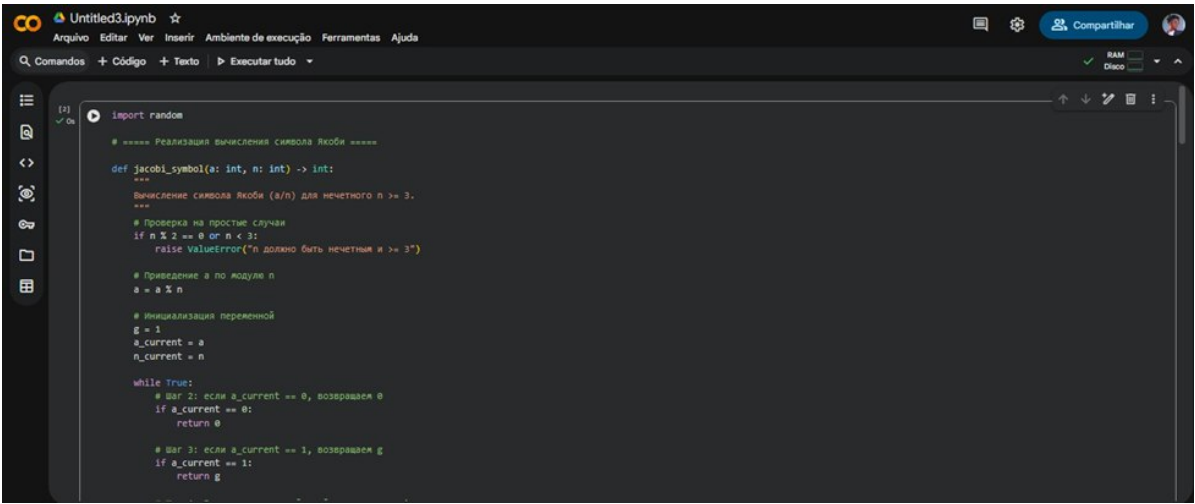
#2.3 Реализация алгоритма, реализующего тест

Соловья-Штрассена Тест Соловья—Штрассена — вероятностный тест простоты, открытый в 1970-х годах Робертом Мартином Соловеем совместно с Фолькером Штрассеном. Тест всегда корректно определяет, что простое число является простым, но для составных чисел с некоторой вероятностью он может дать неверный ответ. Основное преимущество теста заключается в том, что он, в отличие от теста Ферма, распознаёт числа Кармайкла как составные.

#2.4 Реализация алгоритма, реализующего тест Миллера-Рабина Тест Миллера—Рабина — вероятностный полиномиальный тест простоты. Тест Миллера—Рабина, наряду с тестом Ферма и тестом Соловья—Штрассена, позволяет эффективно определить, является ли данное число составным. Однако, с его помощью нельзя строго доказать простоту числа. Тем не менее тест Миллера—Ра-

бина часто используется в криптографии для получения больших случайных простых чисел.

Выполним реализацию этого алгоритма на языке Python (рис. 2.1):



```
import random

# ===== Реализация вычисления символа Якоби =====
def jacobi_symbol(a: int, n: int) -> int:
    """
    Вычисление символа Якоби (a/n) для нечетного n >= 3.
    """
    # Проверка на простые случаи
    if n % 2 == 0 or n < 3:
        raise ValueError("n должно быть нечетным и >= 3")

    # Приведение a по модулю n
    a = a % n

    # Инициализация переменных
    g = 1
    a_current = a
    n_current = n

    while True:
        # Шаг 2: если a_current == 0, возвращаем 0
        if a_current == 0:
            return 0

        # Шаг 3: если a_current == 1, возвращаем g
        if a_current == 1:
            return g

        # Шаг 4: алгоритм вычисления символа Якоби
        # Шаг 4.1: выделение фактора 2 из a_current
        a_temp = a_current
        while a_temp % 2 == 0:
            k += 1
            a_temp //= 2

        a1 = a_temp

        # Шаг 5: вычисление s на основе k и n_current
        if k % 2 == 0:
            s = 1
        else:
            n_mod8 = n_current % 8
            if n_mod8 == 1 or n_mod8 == 7:
                s = 1
            else:
                s = -1

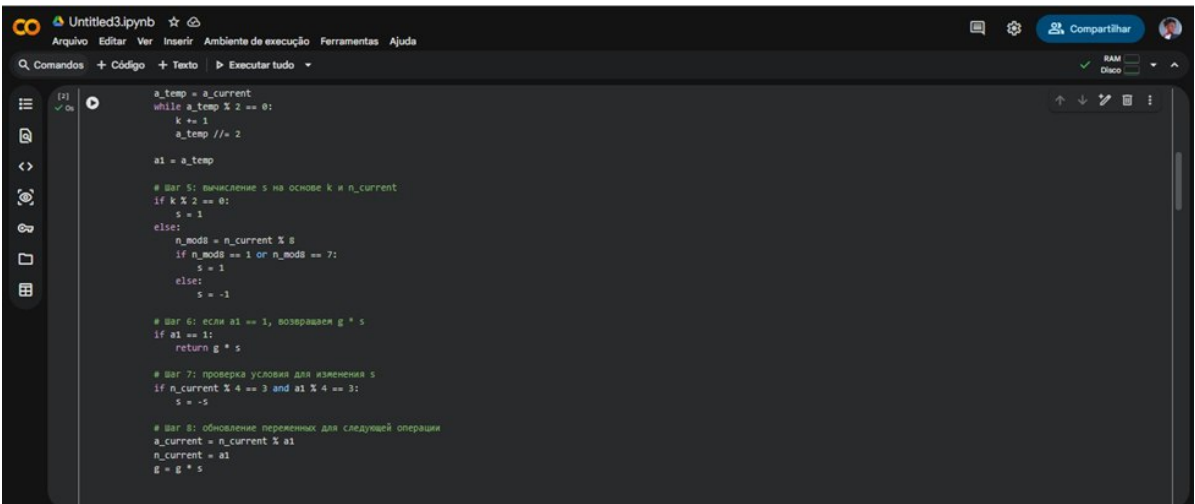
        # Шаг 6: если a1 == 1, возвращаем g * s
        if a1 == 1:
            return g * s

        # Шаг 7: проверка условия для изменения s
        if n_current % 4 == 3 and a1 % 4 == 3:
            s = -s

        # Шаг 8: обновление переменных для следующей операции
        a_current = n_current % a1
        n_current = a1
        g = g * s
```

Рис. 2.1: Реализация вычисления символа Якоби

Проверим работу алгоритма (рис. 2.2):



```
a_temp = a_current
while a_temp % 2 == 0:
    k += 1
    a_temp //= 2

a1 = a_temp

# Шаг 5: вычисление s на основе k и n_current
if k % 2 == 0:
    s = 1
else:
    n_mod8 = n_current % 8
    if n_mod8 == 1 or n_mod8 == 7:
        s = 1
    else:
        s = -1

# Шаг 6: если a1 == 1, возвращаем g * s
if a1 == 1:
    return g * s

# Шаг 7: проверка условия для изменения s
if n_current % 4 == 3 and a1 % 4 == 3:
    s = -s

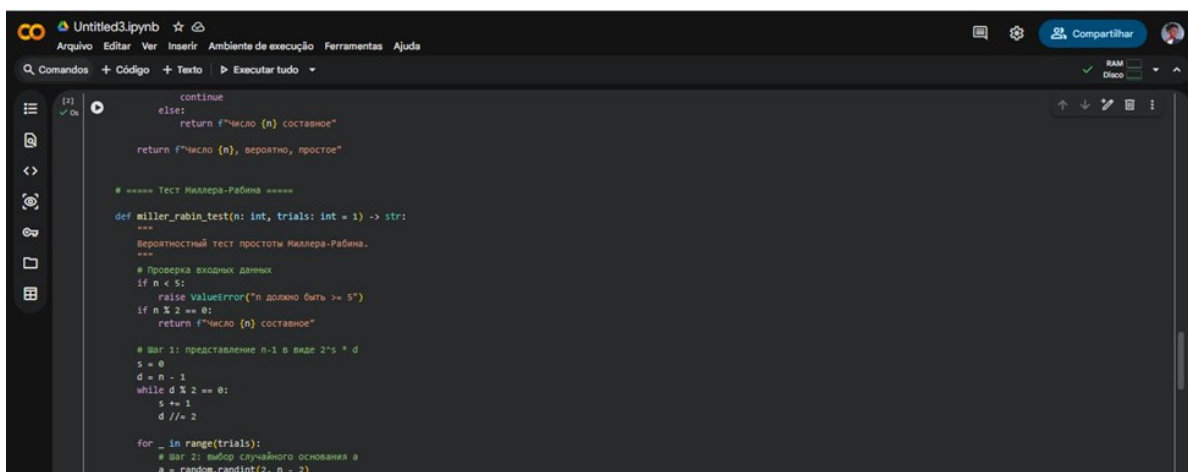
# Шаг 8: обновление переменных для следующей операции
a_current = n_current % a1
n_current = a1
g = g * s
```

Рис. 2.2: Тест Соловея-Штрассена

Проверим работу алгоритма (рис. ??):

[ Тест Соловея-Штрассена] (рис. ??):](image/3.png){ #fig:003 width=100% height=100% }

Выполним реализацию этого алгоритма на языке Python (рис. 2.3):



```

[1] continue
    else:
        return f"число {n} составное"

    return f"число {n}, вероятно, простое"

# ===== Тест Миллера-Рабина =====
def miller_rabin_test(n: int, trials: int = 1) -> str:
    """
    Вероятностный тест простоты Миллера-Рабина.
    """
    # Проверка входных данных
    if n < 5:
        raise ValueError("n должно быть >= 5")
    if n % 2 == 0:
        return f"число {n} составное"

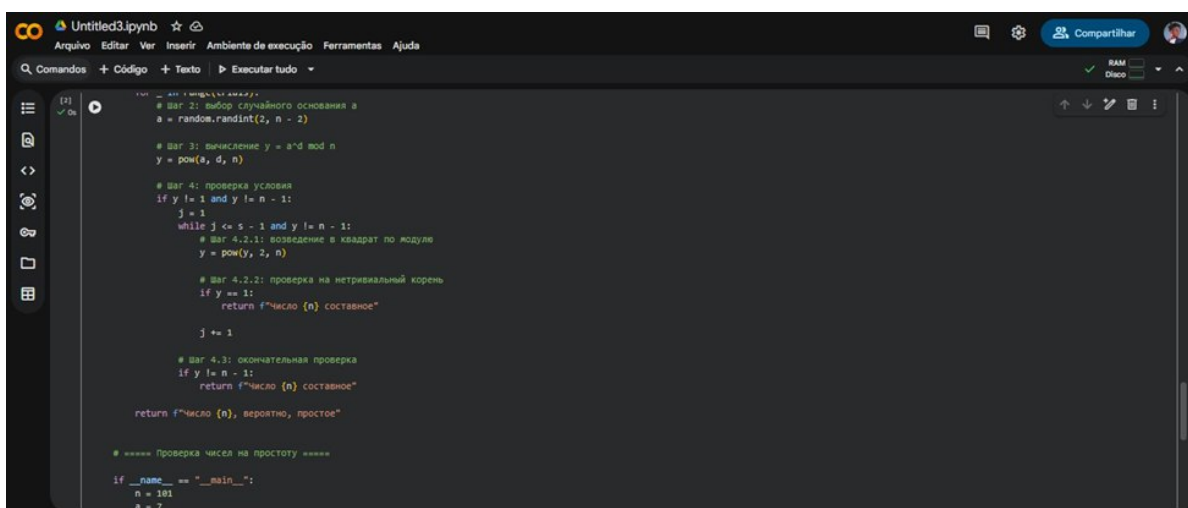
    # Шаг 1: представление n-1 в виде 2^s * d
    s = 0
    d = n - 1
    while d % 2 == 0:
        s += 1
        d //= 2

    for _ in range(trials):
        # Шаг 2: выбор случайного основания a
        a = random.randint(2, n - 2)

```

Рис. 2.3: Тест Миллера-Рабина

Проверим работу алгоритма (рис. 2.4):



```

# Шаг 2: выбор случайного основания a
a = random.randint(2, n - 2)

# Шаг 3: вычисление y = a^d mod n
y = pow(a, d, n)

# Шаг 4: проверка условия
if y != 1 and y != n - 1:
    j = 1
    while j <= s - 1 and y != n - 1:
        # Шаг 4.2.1: возведение в квадрат по модулю
        y = pow(y, 2, n)

        # Шаг 4.2.2: проверка на нетривиальный корень
        if y == 1:
            return f"число {n} составное"

        j += 1

    # Шаг 4.3: окончательная проверка
    if y != n - 1:
        return f"число {n} составное"

    return f"число {n}, вероятно, простое"

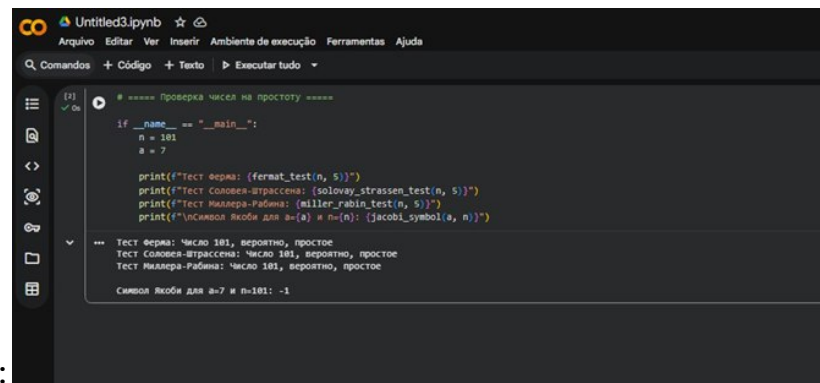
# ===== Проверка чисел на простоту =====
if __name__ == "__main__":
    n = 101
    a = 7

```

Рис. 2.4: Проверка чисел на простоту



Проверим работу алгоритма (рис. ??):



```
# ===== Проверка чисел на простоту =====  
  
if __name__ == "__main__":  
    n = 101  
    a = 7  
  
    print("Тест Ферма: {fermat_test(n, 5)}")  
    print("Тест Соловея-Штрассена: {solovay_strassen_test(n, 5)}")  
    print("Тест Миллера-Рабина: {miller_rabin_test(n, 5)}")  
    print("Символ Якоби для a={a} и n={n}: {jacobi_symbol(a, n)}")  
  
---  
Тест Ферма: Число 101, вероятно, простое  
Тест Соловея-Штрассена: Число 101, вероятно, простое  
Тест Миллера-Рабина: Число 101, вероятно, простое  
  
Символ Якоби для a=7 и n=101: -1
```