

# **Лабораторная работа №5**

**Вероятностные алгоритмы проверки чисел на простоту**

**Лебедева Ольга Андреевна**

# **Содержание**

<b>Цель работы</b>	<b>4</b>
<b>Задачи</b>	<b>5</b>
<b>Объект и предмет исследования</b>	<b>6</b>
<b>Условные обозначения и термины</b>	<b>7</b>
<b>Техническое оснащение и выбранные методы проведения работы</b>	<b>8</b>
<b>Теоретическое введение</b>	<b>9</b>
<b>Задание</b>	<b>10</b>
<b>Тест Ферма</b>	<b>11</b>
<b>Вычисление символа Якоби</b>	<b>12</b>
<b>Тест Соловэя–Штассена</b>	<b>13</b>
<b>Тест Миллера–Рабина</b>	<b>14</b>
<b>Полученные результаты</b>	<b>16</b>
<b>Заключение</b>	<b>17</b>
<b>Библиографическая справка</b>	<b>18</b>

# Список иллюстраций

1	Тест Ферма . . . . .	11
1	Вычисление символа Якоби . . . . .	12
1	Тест Соловэя–Штрассена . . . . .	13
1	Тест Миллера–Рабина . . . . .	14

# **Цель работы**

Изучить и реализовать на языке Julia[1] вероятностные методы проверки простоты чисел: тест Ферма[2], тест Соловэя-Штассена[3], тест Миллера-Рабина[4] и вычисления символа Якоби[5].

## **Задачи**

1. Ознакомиться с теоретическими основами вероятностных тестов на простоту.
2. Реализовать и протестировать алгоритмы Ферма, Соловея–Штрассена и Миллера–Рабина.
3. Изучить вычисление символа Якоби как вспомогательной функции для теста Соловея–Штрассена.
4. Сравнить точность и поведение различных тестов на примерах простых и составных чисел.

## **Объект и предмет исследования**

Объект исследования: методы проверки чисел на простоту.

Предмет исследования: вероятностные алгоритмы, основанные на свойствах чисел в модульной арифметике.

# **Условные обозначения и термины**

Простое число – натуральное число больше единицы, имеющее ровно два различных натуральных делителя: 1 и само себя.

Составное число – натуральное число, имеющее более двух делителей.

Модульная арифметика – система вычислений по модулю  $n$ , в которой рассматриваются остатки от деления чисел на  $n$ .

Вероятностный тест на простоту – алгоритм, который на основе случайно выбираемых параметров определяет, является ли число вероятно простым или точно составным; допускает малую вероятность ошибки в сторону “простое” для некоторых составных чисел.

Символ Якоби ( $a/n$ ) – обобщение символа Лежандра для нечётного положительного  $n$ . Принимает значения -1, 0 или 1 и используется для анализа квадратичных вычетов и в вероятностных тестах на простоту.

Основание теста ( $a$ ) – целое число, выбираемое в заданном диапазоне (обычно от 2 до  $n-2$ ), используемое в формулах проверок.

$\text{powermod}(a, b, n)$  – возведение числа  $a$  в степень  $b$  по модулю  $n$  с использованием быстрого алгоритма (используется для эффективности и предотвращения переполнения).

# **Техническое оснащение и выбранные методы проведения работы**

Программное обеспечение:

- Язык программирования Julia.
- Среда разработки JupyterLab / VS Code.

Методы:

- Использование модульной арифметики для возведения в степень по модулю.
- Генерация случайных оснований для тестов.
- Проверка выполнения сравнений в модульной системе.
- Многократное повторение тестов для снижения вероятности ошибки.

# Теоретическое введение

Проверка чисел на простоту является важной задачей теории чисел и криптографии. Детерминированные методы, такие как деление на все возможные делители до  $\sqrt{n}$  требуют слишком больших вычислений для больших чисел. Поэтому в практических системах используются вероятностные тесты, которые позволяют быстро определить, является ли число вероятно простым.

Среди таких тестов выделяются три наиболее известных метода: тест Ферма, тест Соловэя–Штассена и тест Миллера–Рабина. Они основаны на свойствах чисел в модульной арифметике и используют случайные основания для проверки выполнения определённых сравнений. Ошибка возможна, но вероятность её возникновения стремительно уменьшается с увеличением числа итераций.

Символ Якоби, в свою очередь, является важным элементом при реализации теста Соловэя–Штассена, поскольку он позволяет связать поведение степени числа с теоретическими свойствами квадратичных вычетов.

# **Задание**

Реализовать на Julia четыре функции:

1. Тест Ферма.
2. Вычисление символа Якоби.
3. Тест Соловэя–Штрассена.
4. Тест Миллера–Рабина.

# Тест Ферма

Выполним задание 1 с помощью языка Julia:

Проверим результат работы кода: См. рис. 1

```
[23]: function fermat_test(n::Int, k::Int=5)
    if n < 5 || iseven(n)
        return false
    end
    for _ in 1:k
        a = rand(2: n-2)
        if gcd(a, n) != 1
            return false
        end
        if powermod(a, n-1, n) != 1
            return false
        end
    end
    return true
end

for n in [17, 21, 97, 99, 561]
    println("Число $n:")
    println("Ферма: ", fermat_test(n))
    println()
end
```

Число 17:  
Ферма: true  
Число 21:  
Ферма: false  
Число 97:  
Ферма: true  
Число 99:  
Ферма: false  
Число 561:  
Ферма: false

Рис. 1: Тест Ферма

Алгоритм основан на малой теореме Ферма, согласно которой, если число  $p$  простое, то для любого целого  $a$ , взаимно простого с  $p$ , выполняется соотношение  $a^{p-1} \equiv 1 \pmod{p}$ .

Проверка заключается в выборе нескольких случайных оснований  $a$  из диапазона  $2 \leq a \leq n - 2$  и вычислении значения  $a^{n-1} \pmod{n}$ . Если равенство выполняется для всех выбранных оснований, число считается вероятно простым. Метод обеспечивает простую и быструю проверку свойств числа в модульной арифметике, что делает его удобным для первичной оценки простоты.

# Вычисление символа Якоби

Выполним задание 2 с помощью языка Julia и проверим результат работы кода: См. рис. 2

```
function jacobi(a::Int, n::Int)
    if n <= 0 || iseven(n)
        error("n должно быть положительным и нечётным")
    end
    a %= n
    result = 1
    while a != 0
        while iseven(a)
            a = div(a, 2)
            if n % 8 == 3 || n % 8 == 5
                result = -result
            end
        end
        a, n = n, a
        if a%4==3 && n%4 ==3
            result = -result
        end
        a %= n
    end
    return n==1 ? result : 0
end
```

Рис. 1: Вычисление символа Якоби

Символ Якоби ( $(a/n)$ ) представляет собой обобщение символа Лежандра и используется для анализа свойств чисел в модульной арифметике. Он принимает значения 1,  $-1$  или 0 и вычисляется на основе ряда правил, включая выделение степеней числа 2 и закон квадратичной взаимности.

Значение символа Якоби позволяет установить связь между степенью числа и его квадратичными вычетами по модулю  $n$ . В контексте проверки простоты символ Якоби служит важным элементом, применяемым в тесте Соловэя–Штрассена.

# Тест Соловэя–Штрассена

Выполним задание 3 с помощью языка Julia. Проверим результат работы кода: См. рис.

3

```
function strassen(n::Int, k::Int=5)
    if n < 5 || iseven(n)
        return false
    end
    for _ in 1:k
        a = rand(2, n-2)
        if gcd(a, n) != 1
            return false
        end
        x = jacobi(a,n) % n
        y = powermod(a, div(n-1, 2), n)
        println("a = $a, символ Якоби (a/n) = $(jacobi(a, n)), a^((n-1)/2) mod n = $y")
        if x==0 || y==0
            return false
        end
    end
    return true
end

for n in [17, 21, 97, 99, 561]
    println("число $n:")
    println("Соловэй-штрассен: ", strassen(n))
end
```

Число 17:  
a = 12, символ Якоби (a/n) = -1, a^((n-1)/2) mod n = 16  
Соловэй-штрассен: false  
Число 21:  
a = 4, символ Якоби (a/n) = 1, a^((n-1)/2) mod n = 4  
Соловэй-штрассен: false  
Число 97:  
a = 19, символ Якоби (a/n) = -1, a^((n-1)/2) mod n = 96  
Соловэй-штрассен: false  
Число 99:  
a = 69, символ Якоби (a/n) = 1, a^((n-1)/2) mod n = 89  
Соловэй-штрассен: false  
Число 561:  
a = 403, символ Якоби (a/n) = 1, a^((n-1)/2) mod n = 67  
Соловэй-штрассен: false

Рис. 1: Тест Соловэя–Штрассена

Тест Соловэя–Штрассена объединяет идею теста Ферма и использование символа Якоби для уточнения результатов. Если число  $n$  простое, то для любого  $a$ , взаимно простого с  $n$ , выполняется соотношение  $a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \pmod{n}$ .

Здесь  $\left(\frac{a}{n}\right)$  — символ Якоби, вычисляемый по отдельному алгоритму. В ходе проверки выбирается несколько случайных оснований  $a$ , для которых вычисляются обе части равенства. Если они совпадают, число считается вероятно простым. Метод позволяет уточнить оценку простоты числа и служит развитием идеи теста Ферма, опираясь на более глубокие свойства теории чисел.

# Тест Миллера–Рабина

Выполним задание 4 с помощью языка Julia. Проверим результат работы кода: См. рис.

4

```
function miller_rabin(n::Int, k::Int=5)
    if n < 5 || iseven(n)
        return false
    end
    d = n - 1
    s = 0
    while iseven(d)
        d = div(d, 2)
        s += 1
    end
    for _ in 1:k
        a = rand(2:n-2)
        x = powmod(a, d, n)
        if x == 1 || x == n - 1
            continue
        end
        passed = false
        for _ in 1:(s - 1)
            x = powmod(x, 2, n)
            if x == n - 1
                passed = true
                break
            end
        end
        if !passed
            return false
        end
    end
    return true
end

for n in [17, 21, 97, 99, 561]
    println("\nЧисло $n:")
    println("Миллер-Рабин: ", miller_rabin(n))
end
```

```
число 17:
Миллер-Рабин: true

число 21:
Миллер-Рабин: false

число 97:
Миллер-Рабин: true

число 99:
Миллер-Рабин: false

число 561:
Миллер-Рабин: false
```

Рис. 1: Тест Миллера–Рабина

Тест Миллера–Рабина представляет собой один из наиболее надёжных вероятностных методов проверки простоты числа и применяется в криптографических системах. Число  $n - 1$  представляется в виде  $n - 1 = 2^s \cdot d$ , где  $d$  — нечётное. Для случайного основания  $a$  проверяются условия:

$$a^d \equiv 1 \pmod{n} \text{ или } a^{2^r d} \equiv -1 \pmod{n} \text{ для некоторого } r < s.$$

Если хотя бы одно из этих условий выполняется, число рассматривается как вероятно

простое. Тест повторяется несколько раз для разных оснований, что повышает достоверность результата. Благодаря сочетанию эффективности и высокой вероятности правильного решения, метод Миллера–Рабина является практическим стандартом проверки простоты чисел.

# Полученные результаты

В ходе лабораторной работы были реализованы четыре алгоритма на языке Julia: тест Ферма, функция вычисления символа Якоби, тест Соловея–Штрассена и тест Миллера–Рабина. Для проверки корректности работы были проведены эксперименты на различных типах чисел: простых (17, 97), составных (21, 99) и числе Кармайкла 561.

Результаты показали ожидаемое поведение всех алгоритмов:

- для простых чисел тесты Ферма, Соловея–Штрассена и Миллера–Рабина подтвердили их простоту;
- для составных чисел значения корректно определялись как составные;
- число 561, обладающее особыми свойствами, также корректно было классифицировано тестами Соловея–Штрассена и Миллера–Рабина.

Таким образом, все реализованные функции отработали корректно, подтвердив как правильность кода, так и теоретические зависимости, заложенные в алгоритмы.

## **Заключение**

В результате выполнения лабораторной работы были изучены и реализованы вероятностные методы проверки чисел на простоту. Алгоритмы Ферма, Соловэя–Штрассена и Миллера–Рабина были исследованы как с теоретической, так и с практической точки зрения. Реализация на Julia позволила закрепить навыки работы с модульной арифметикой.

## **Библиографическая справка**

- [1] Julia
- [2] Малая теорема Ферма
- [3] Тест Соловэя–Штрассена
- [4] Тест Миллера–Рабина
- [5] Символ Якоби