

8 ноября 2025

# Общая информация о лабораторной работе

## Цель работы

Изучение и программная реализация вероятностных алгоритмов проверки чисел на простоту: теста Ферма, теста Соловья-Штрассена и теста Миллера-Рабина, а также сравнение их эффективности и точности.

## Задание

Реализовать все рассмотренные алгоритмы программно.

## Теоретическая часть

Основные понятия:

- Простое число - натуральное число, имеющее ровно два делителя: 1 и само число
- Составное число - натуральное число, имеющее более двух делителей
- Вероятностный алгоритм - алгоритм, использующий случайность и дающий правильный ответ с высокой вероятностью

# Выполнение лабораторной работы

## Алгоритм, реализующий тест Ферма.

```
function fermat_test(n::Int, k::Int=10)

    if n < 5
        return n == 2 || n == 3
    end

    if n % 2 == 0
        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

```



## Алгоритм вычисления символа Якоби.

```

function jacobi_symbol(a::Int, n::Int)
    if n % 2 == 0 || n < 3
        throw(ArgumentError("n должно быть нечетным и ≥ 3"))
    end

    a = a % n
    g = 1

    while true
        if a == 0
            return 0
        end

        if a == 1
            return g
        end

        k = 0
        a1 = a
        while a1 % 2 == 0
            a1 ÷= 2
            k += 1
        end

        s = 1
        if k % 2 == 1
            n_mod8 = n % 8
            if n_mod8 == 3 || n_mod8 == 5
                s = -1
            end
            # если n_mod8 == 1 или 7, s остается 1
        end

        if a1 == 1
            return g * s
        end
    end
end

```

```

        if n % 4 == 3 && a1 % 4 == 3
            s = -s
        end

        a, n = n % a1, a1
        g *= s
    end
end

```



## Алгоритм, реализующий тест Соловья-Штрассена.

```

function solovay_strassen_test(n::Int, k::Int=10)
    if n < 5
        return n == 2 || n == 3
    end

    if n % 2 == 0
        return false
    end

    for _ in 1:k
        a = rand(2:(n-2))

        if gcd(a, n) != 1
            return false
        end

        r = powermod(a, (n-1) ÷ 2, n)
        j = jacobi_symbol(a, n)

        if j == 0 || (r != j % n)
            return false
        end
    end

    return true
end

```



## Алгоритм, реализующий тест Миллера-Рабина.

```

function miller_rabin_test(n::Int, k::Int=10)
    if n < 5
        return n == 2 || n == 3
    end

    if n % 2 == 0
        return false
    end

```

```

end

s = 0
r = n - 1
while r % 2 == 0
    r /= 2
    s += 1
end

for _ in 1:k
    a = rand(2:(n-2))
    y = powermod(a, r, n)

    if y != 1 && y != n-1
        j = 1
        while j <= s-1 && y != n-1
            y = powermod(y, 2, n)
            if y == 1
                return false
            end
            j += 1
        end

        if y != n-1
            return false
        end
    end
end

return true
end

```



## Выводы

1. Тест Ферма является самым простым в реализации, но наименее надежным. Он часто ошибается на числах Кармайкла - составных числах, которые ведут себя как простые по малой теореме Ферма.
2. Тест Соловья-Штрассена более надежен, чем тест Ферма, благодаря использованию символа Якоби. Однако он сложнее в реализации и требует дополнительных вычислений.
3. Тест Миллера-Рабина показал наивысшую надежность среди всех рассмотренных алгоритмов. При достаточном количестве итераций вероятность ошибки становится пренебрежимо малой.
4. Для практического применения в криптографии рекомендуется использовать тест Миллера-Рабина с 10-20 итерациями, что обеспечивает высокую надежность при приемлемой скорости работы.
5. Все вероятностные алгоритмы имеют преимущество перед детерминированными в скорости работы, что особенно важно при работе с большими числами, используемыми в современных криптографических системах.