Front matter

Front matter

lang: ru-RU title: "Математические основы защиты информации и информационной безопасности" subtitle: "Отчёт по лабораторной работе №5:Вероятностные алгоритмы проверки author: "Бекбузарова Роза" institute:

- Российский университет дружбы народов, Москва, Россия

i18n babel

babel-lang: russian babel-otherlangs: english

Formatting pdf

toc: false toc-title: Содержание slide_level: 2 aspectratio: 169 section-titles: true theme: metropolis header-includes:

- \metroset{progressbar=frametitle,sectionpage=progressbar,numbering=fraction}
- '\makeatletter'
- '\beamer@ignorenonframefalse'
- '\makeatother'

Цель работы

Основной целью работы является реализовать разными алгоритмами проверки чисел на простоту.

Выполнение лабораторной работы

Алгоритм, реализующий тест Ферма

[15]:
```
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
        # Шаг 1: вынесем все двойки из a
        while iseven(a)
            a ÷= 2
            if n % 8 == 3 || n % 8 == 5
                result = -result
            end
        end
        a, n = n, a  # меняем местами
        if a % 4 == 3 && n % 4 == 3
            result = -result
        end

        a %= n
    end
    return n == 1 ? result : 0
end
println("Символ Якоби (1001/9907) = ", jacobi(1001, 9907))  # должно быть -1
println("Символ Якоби (5/21) = ", jacobi(5, 21))          # должно быть 1
```

```
Символ Якоби (1001/9907) = -1
Символ Якоби (5/21) = 1
```

Алгоритм вычисление символа Якоби

```
    while a != 0
        while iseven(a)
            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
function solovay_strassen(n::Int; k::Int=10)
    if n < 2
        return false
    elseif n in (2, 3)
        return true
    elseif iseven(n)
        return false
    end

    for _ in 1:k
        a = rand(2:n-2)
        x = powermod(a, (n - 1) ÷ 2, n)
        j = jacobi(a, n) % n  # символ Якоби приводим к mod n

        if x != j
            return false  # точно составное
        end
    end
    return true  # вероятно простое
end
println("n = 17 (простое): ", solovay_strassen(17, k=10))
println("n = 21 (составное): ", solovay_strassen(21, k=10))
println("n = 561 (число Кармайкла): ", solovay_strassen(561, k=10))
```

```
n = 17 (простое): false
n = 21 (составное): false
n = 561 (число Кармайкла): false
```


Алгоритм, реализующий тест Соловэя-Штрассена

[19]:
```
using Random
function miller_rabin(n::Int; k::Int=10)
    if n < 2
        return false
    elseif n in (2, 3)
        return true
    elseif iseven(n)
        return false
    end
    # 1. Представляем n-1 как 2^s * d
    d = n - 1
    s = 0
    while iseven(d)
        d ÷= 2
        s += 1
    end
    # 2. k случайных проверок
    for _ in 1:k
        a = rand(2:n-2)
        x = powermod(a, d, n)

        if x == 1 || x == n-1
            continue
        end
        composite = true
        for _ in 1:(s-1)
            x = powermod(x, 2, n)
            if x == n-1
                composite = false
                break
            end
        end
        if composite
            return false # составное число
        end
    end
    return true # вероятно простое
end
println("n = 17 (простое): ", miller_rabin(17, k=10))
println("n = 21 (составное): ", miller_rabin(21, k=10))
println("n = 561 (число Кармайкла): ", miller_rabin(561, k=10))
```



```
n = 17 (простое): true
n = 21 (составное): false
n = 561 (число Кармайкла): false
```

Алгоритм, реализующий тест Миллера-Рабина

[13]:
```
using Random
function fermat_primality_test(n::Int; k::Int=5)
    if n <= 1
        return false
    elseif n <= 3
        return true
    elseif iseven(n)
        return false
    end
    for _ in 1:k
        a = rand(2:n-2)   # случайное a из диапазона [2, n-2]
        # проверяем условие Ферма: a^(n-1) mod n == 1
        if powermod(a, n-1, n) != 1
            return false  # точно составное
        end
    end
    return true  # вероятно простое
end
println("Тестируем числа 17, 21, 561 (псевдопростое):")
for n in [17, 21, 561]
    println("n = $n  ->  ", fermat_primality_test(n, k=10))
end
```

```
Тестируем числа 17, 21, 561 (псевдопростое):
n = 17  ->  true
n = 21  ->  false
n = 561  ->  false
```

Вывод

В ходе выполнения лабораторной работы было реализовано разными алгоритмами проверка чисел на простоту.