

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ПРЕЗЕНТАЦИЯ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 4

дисциплина: Математические основы защиты информации и информационной безопасности

Студент: Пиняева Анна Андреевна

Группа: НПИмд-01-24

МОСКВА

2025

Цель работы

Изучение и реализация вычисления НОД с помощью алгоритма Евклида на языке Julia.

Ход работы

1. Алгоритм Евклида

```
function euclidean_gcd(a::Int, b::Int)
    a, b = abs(a), abs(b)
    while b != 0
        a, b = b, a % b
    end
    return a
end
```

2. Бинарный алгоритм Евклида

```
function binary_gcd(a::Int, b::Int)
    a, b = abs(a), abs(b)

    a == 0 && return b
    b == 0 && return a

    shift = 0
    while iseven(a) && iseven(b)
        a ÷= 2
        b ÷= 2
        shift += 1
    end

    while a != 0
        while iseven(a)
            a ÷= 2
        end
        while iseven(b)
            b ÷= 2
        end

        if a >= b
            a = a - b
        else
            b = b - a
        end
    end

    return b << shift # b * 2^shift
end
```

3. Расширенный алгоритм Евклида

```
function extended_euclidean(a::Int, b::Int)
    a, b = abs(a), abs(b)

    if b == 0
        return (a, 1, 0)
    end

    x0, x1 = 1, 0
    y0, y1 = 0, 1

    while b != 0
        q = a ÷ b
        a, b = b, a % b
        x0, x1 = x1, x0 - q * x1
        y0, y1 = y1, y0 - q * y1
    end

    return (a, x0, y0)
end
```

4. Расширенный бинарный алгоритм Евклида

```
function extended_binary_gcd(a::Int, b::Int)
    a, b = abs(a), abs(b)

    if a == 0
        return (b, 0, 1)
    elseif b == 0
        return (a, 1, 0)
    end

    g = 1
    while iseven(a) && iseven(b)
        a ÷= 2
        b ÷= 2
        g *= 2
    end

    u, v = a, b
    A, B, C, D = 1, 0, 0, 1

    while u != 0
        while iseven(u)
            u ÷= 2
            if iseven(A) && iseven(B)
                A ÷= 2
                B ÷= 2
            else
                A = (A + b) ÷ 2
                B = (B - a) ÷ 2
            end
        end
        while iseven(v)
            v ÷= 2
            if iseven(C) && iseven(D)
                C ÷= 2
                D ÷= 2
            end
        end
    end
```

```
        else
            C = (C + b) ÷ 2
            D = (D - a) ÷ 2
        end
    end

    if u >= v
        u -= v
        A -= C
        B -= D
    else
        v -= u
        C -= A
        D -= B
    end
end

d = g * v
return (d, C, D)
end
```

5. Вывод результатов

```
function test_algorithms()
    test_cases = [
        (54, 24),
        (12345, 24690),
        (12345, 54321),
        (12345, 12541),
        (91, 105),
        (105, 154),
        (17, 13), # Простые числа
        (100, 25), # Одно делит другое
        (0, 15), # Ноль
        (15, 0), # Ноль
        (-54, 24), # Отрицательные числа
        (54, -24) # Отрицательные числа
    ]

    println("ТЕСТИРОВАНИЕ АЛГОРИТМОВ НОД")
    println("="^50)

    for (i, (a, b)) in enumerate(test_cases)
        println("\nТест $i: НОД($a, $b)")
        println("-"^30)

        # Алгоритм Евклида
        gcd1 = euclidean_gcd(a, b)
        println("Алгоритм Евклида: $gcd1")
    # Бинарный алгоритм
        gcd2 = binary_gcd(a, b)
        println("Бинарный алгоритм: $gcd2")

        # Расширенный алгоритм
        gcd3, x3, y3 = extended_euclidean(a, b)
        println("Расширенный алгоритм: $gcd3")
        println("Коэффициенты Безу:  $a \times x3 + b \times y3 = (a \times x3 + b \times y3)$ ")
```



```

# Расширенный бинарный алгоритм
gcd4, x4, y4 = extended_binary_gcd(a, b)
println("Расшир. бинарный: $gcd4")
println("Коэффициенты Безу:  $a \times x4 + b \times y4 = (a \times x4 + b \times y4)$ ")

# Проверка согласованности
if gcd1 == gcd2 == gcd3 == gcd4
    println("✓ Все алгоритмы дали одинаковый результат")
else
    println("X Ошибка: результаты различаются!")
end

# Проверка взаимной простоты
if gcd1 == 1
    println("✓ Числа взаимно просты")
else
    println("Числа не взаимно просты")
end
end
end

function main()

    println()

    test_algorithms()
end
main()

```

Результат тестирования представлен на рис.1

Рис. 1 Тестирование:

ТЕСТИРОВАНИЕ АЛГОРИТМОВ НОД

Тест 1: НОД(54, 24)

Алгоритм Евклида: 6
Бинарный алгоритм: 6
Расширенный алгоритм: 6
Коэффициенты Безу: $54 \times 1 + 24 \times -2 = 6$
Расшир. бинарный: 6
Коэффициенты Безу: $54 \times 9 + 24 \times -20 = 6$
✓ Все алгоритмы дали одинаковый результат
Числа не взаимно просты

Тест 2: НОД(12345, 24690)

Алгоритм Евклида: 12345
Бинарный алгоритм: 12345
Расширенный алгоритм: 12345
Коэффициенты Безу: $12345 \times 1 + 24690 \times 0 = 12345$
Расшир. бинарный: 12345
Коэффициенты Безу: $12345 \times 12345 + 24690 \times -6172 = 12345$
✓ Все алгоритмы дали одинаковый результат
Числа не взаимно просты

Тест 3: НОД(12345, 54321)

Алгоритм Евклида: 3
Бинарный алгоритм: 3
Расширенный алгоритм: 3
Коэффициенты Безу: $12345 \times 3617 + 54321 \times -822 = 3$
Расшир. бинарный: 3
Коэффициенты Безу: $12345 \times -14490 + 54321 \times 3293 = 3$
✓ Все алгоритмы дали одинаковый результат
Числа не взаимно просты

Вывод: В ходе данной работы мной были изучены и реализованы различные вариации метода Евклида для вычисления НОД. Написан программный код на языке Julia и протестирован.