Классический алгоритм Евклида

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

Расширенный алгоритм Евклида

function extended_gcd(a::Int, b::Int)::Tuple{Int, Int, Int} x, last_x = 0, 1 y, last_y = 1, 0 while b != 0 quotient = div(a, b) a, b = b, a % b x, last_x = last_x - quotient * x, x y, last_y = last_y - quotient * y, y end return (abs(a), last_x, last_y) end

Бинарный алгоритм Евклида

function binary_gcd(a::Int, b::Int)::Int if a == 0 return abs(b) elseif b == 0 return abs(a) end

```
shift = 0
while ((a | b) & 1) == 0
    a >>= 1
    b >>= 1
    shift += 1
end

while (a & 1) == 0
    a >>= 1
end

while b != 0
    while (b & 1) == 0
        b >>= 1
    end
    if a > b
        a, b = b, a
    end
    b -= a
end

return abs(a << shift)
```

end

Расширенный бинарный алгоритм Евклида

function extended_binary_gcd(a::Int, b::Int)::Tuple{Int, Int, Int} if a == 0 return (abs(b), 0, 1) elseif b == 0 return (abs(a), 1, 0) end

```
shift = 0
while ((a | b) & 1) == 0
    a >>= 1
    b >>= 1
    shift += 1
end

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

while u != 0
    while (u & 1) == 0
        u >>= 1
        if (A & 1) == 0 && (B & 1) == 0
            A >>= 1
            B >>= 1
        else
            A = (A + b) >> 1
            B = (B - a) >> 1
        end
    end

    while (v & 1) == 0
        v >>= 1
        if (C & 1) == 0 && (D & 1) == 0
            C >>= 1
            D >>= 1
        else
            C = (C + b) >> 1
            D = (D - a) >> 1
        end
    end

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

return (abs(v << shift), C, D)
```

end

Примеры использования всех функций

println("Классический алгоритм Евклида:") println(gcd_euclidean(12345, 24690)) # Вывод: 12345

println("\nРасширенный алгоритм Евклида:") gcd_val, x, y = extended_gcd(12345, 24690) println("НОД: $gcd_val, x: $x, y: $y") # Вывод: НОД: 12345, x: 1, y: -1

println("\nБинарный алгоритм Евклида:") println(binary_gcd(12345, 24690)) # Вывод: 12345

println("\nРасширенный бинарный алгоритм Евклида:") gcd_val, x, y = extended_binary_gcd(12345, 24690) println("НОД: $gcd_val, x: $x, y: $y")