Лабораторная работа 4

Вычисление наибольшего общего делителя

Климин Никита Денисович

Содержание

1. Цель работы

Изучение и реализация алгоритмов вычисления наибольшего общего делителя целых чисел: алгорим евклида, бинарный, расширенный и расширенный бинарный.

2. Задание

Реализовать четыре алгоритма вычисления НОД, проверить их работу и вывести результаты.

3. Теоретическое введение

Наибольший общий делитель (НОД) целых чисел `a` и `b` --- это число $d \neq 0$, которое делит оба числа, и любое другое число, делящее `a` и `b`, делится на `d`.

НОД можно представить как линейную комбинацию: d=ax+by, где x, y □ Z

Числа называются **взаимно простыми**, если их НОД равен 1.

Алгоритмы вычисления НОД: - **Классический алгоритм Евклида** использует повторное деление с остатком. - **Бинарный алгоритм (Штейна)** применяет побитовые операции для ускорения вычислений. - **Расширенные алгоритмы** позволяют дополнительно находить коэффициенты `x` и `y` для линейной комбинации.

4. Выполнение лабораторной работы

Программа была написана на Julia.

```
gcd_euclid(a::Int, b::Int) = b == 0 ? abs(a) : gcd_euclid(b, a % b)

function gcd_binary(a::Int, b::Int) # бинарный алгоритм
    a, b = abs(a), abs(b) # ипользуем положительные числа
    a == 0 && return b # ели одно из чисел 0, то возвращаем другое
    b == 0 && return a

    k = 0
    while iseven(a) && iseven(b) # пока оба числа чётные
        a >>= 1
        b >>= 1
        k += 1
    end

    while a != b # пока числа не равны
        if iseven(a)
            a >>= 1 # делим а на 2 если оно чётное
        elseif iseven(b)
            b >>= 1 # делим б на 2 если оно чётное
        elseif a > b
            a = (a - b) >> 1 # вычитаем и делим на 2
        else
            b = (b - a) >> 1 # вычитаем и делим на 2
        end
```

```
    end

    return a << k # возвращаем нод
end

function gcd_extended_euclid(a::Int, b::Int) # расширенный алгоритм
    old_r, r, old_x, x, old_y, y = a, b, 1, 0, 0, 1
    while r != 0
        q = div(old_r, r) # целая часть деления
        old_r, r, old_x, x, old_y, y = r, old_r - q*r, x, old_x - q
    end
    return abs(old_r) # возвращаем модуль нод
end

function gcd_extended_binary_euclid(a::Int, b::Int) # расширенный би
    a, b = abs(a), abs(b) # ипользуем положительные числа
    a == 0 && return b # ели одно из чисел 0, то возвращаем другое
    b == 0 && return a

    k = 0
    while iseven(a) && iseven(b) # пока оба числа чётные
        a >>= 1
        b >>= 1
        k += 1
    end

    u, v, A, B, C, D = a, b, 1, 0, 0, 1
    while u != 0
        while iseven(u)
            u >>= 1
            (iseven(A) && iseven(B)) ? (A>>=1; B>>=1) : (A=(A+b)>>1
        end
        while iseven(v)
```

```
            v >>= 1
            (iseven(C) && iseven(D)) ? (C>>=1; D>>=1) : (C=(C+b)>>1
        end
        if u >= v
            u -= v; A -= C; B -= D
        else
            v -= u; C -= A; D -= B
        end
    end

    return v << k
end

function main()
    a, b = 12345, 25
    println("алгоритм Евклида: ", gcd_euclid(a, b))
    println("бинарный алгоритм: ", gcd_binary(a, b))
    println("расширенный алгоритм: ", gcd_extended_euclid(a, b))
    println("расширенный бинарный алгоритм: ", gcd_extended_binary_
end

main()
```

Пример работы программы в терминале

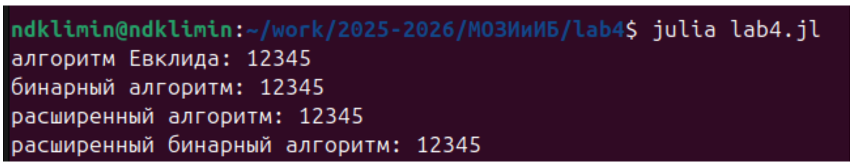

Рис. 4.1.: Пример работы программы

5. Выводы

Все алгоритмы корректно вычисляют НОД. Практическая проверка показала идентичные результаты для всех методов

Список литературы