Отчет по лабораторной работе №4

Вычисление наибольшего общего делителя

Арам Грачьяевич Саргсян

Содержание

Список иллюстраций

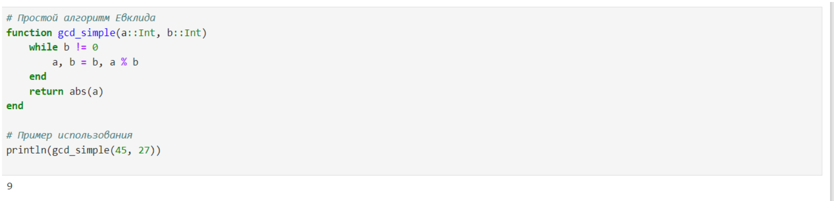
Список таблиц

1 Цель работы

Изучить алгоритмы вычисления НОД

2 Выполнение лабораторной работы

1. Я реализовал алгоритм Евклида для вычесления НОД на julia (рис. 2.1).



```julia
# Простой алгоритм Евклида
function gcd_simple(a::Int, b::Int)
    while b != 0
        a, b = b, a % b
    end
    return abs(a)
end

# Пример использования
println(gcd_simple(45, 27))
```

```
9
```

Рис. 2.1: Базовый алгоритм Евклида

2. Я реализовал бинарный алгоритм Евклида для вычесления НОД на julia (рис. 2.2).

```julia
# Бинарный алгоритм Евклида
function gcd_binary(a::Int, b::Int)
    # Базовые случаи
    if a == 0 return abs(b) end
    if b == 0 return abs(a) end

    # Оба числа четные
    if iseven(a) && iseven(b)
        return 2 * gcd_binary(a >> 1, b >> 1)
    # a четное, b нечетное
    elseif iseven(a)
        return gcd_binary(a >> 1, b)
    # a нечетное, b четное
    elseif iseven(b)
        return gcd_binary(a, b >> 1)
    # Оба числа нечетные
    else
        return gcd_binary(abs(a - b) >> 1, min(a, b))
    end
end

# Пример использования
println(gcd_binary(45, 27))  # Вывод: 9
```

```
9
```

Рис. 2.2: Бинарный алгоритм Евклида

3. Я реализовал расширенный алгоритм Евклида для вычесления НОД на julia (рис. 2.3).

```julia
# Расширенный алгоритм Евклида
function extended_gcd(a::Int, b::Int)
    if b == 0
        return (abs(a), 1, 0)  # Возвращаем (gcd, x, y)
    else
        gcd, x1, y1 = extended_gcd(b, a % b)
        x = y1
        y = x1 - (a ÷ b) * y1
        return (gcd, x, y)
    end
end

# Пример использования
gcd, x, y = extended_gcd(45, 18)
println("GCD: $gcd, x: $x, y: $y")
```

```
GCD: 9, x: 1, y: -2
```

Рис. 2.3: Расширенный алгоритм Евклида

4. Я реализовал расширенный алгоритм Евклида для вычесления НОД на julia (рис. 2.4).

```
# Вспомогательная функция для деления числа на 2 с учётом коэффициентов
function halve_with_coeffs(a::Int, x::Int, y::Int)
    while iseven(a)
        a >>= 1
        if iseven(x) && iseven(y)
            x >>= 1
            y >>= 1
        elseif iseven(x)
            x >>= 1
        elseif iseven(y)
            y >>= 1
        end
    end
    return a, x, y
end

function extended_gcd_binary(a::Int, b::Int)
    if a == 0 return (abs(b), 0, 1) end
    if b == 0 return (abs(a), 1, 0) end

    shift = 0

    # Убираем общие факторы 2
    while iseven(a) && iseven(b)
        a >>= 1
        b >>= 1
        shift += 1
    end

    x1, y1 = 1, 0
    x2, y2 = 0, 1

    while a != 0
        a, x1, y1 = halve_with_coeffs(a, x1, y1)
        b, x2, y2 = halve_with_coeffs(b, x2, y2)

        if a >= b
            a -= b
            y1 -= y2
```

Рис. 2.4: Расширенный бинарный алгоритм Евклида

```
    x1, y1 = 1, 0
    x2, y2 = 0, 1

    while a != 0
        a, x1, y1 = halve_with_coeffs(a, x1, y1)
        b, x2, y2 = halve_with_coeffs(b, x2, y2)

        if a >= b
            a -= b
            x1 -= x2
            y1 -= y2
        else
            b -= a
            x2 -= x1
            y2 -= y1
        end
    end

    gcd = b << shift  # Возвращаем НОД, умноженный на 2^shift
    return gcd, x2, y2
end

# Пример использования
gcd, x, y = extended_gcd_binary(45, 36)
println("GCD: $gcd, x: $x, y: $y")
```

```
GCD: 9, x: 0, y: 1
```

Рис. 2.5: Расширенный бинарный алгоритм Евклида

3 Выводы

Я реализовал алгоритмы Евклида для вычисления НОД.