Лабораторная работа №4: отчет.

Шифры простой замены

Евдокимов Максим Михайлович. Группа - НФИмд-01-24.

Содержание

Список иллюстраций

Цели и задачи работы

Цель лабораторной работы

Изучить и реализовать все представленные методы Евклида.

Задание

1. Реализовать классический алгоритм Евклида.
2. Реализовать бинарный алгоритм Евклида.
3. Реализовать расширенный алгоритм Евклида.
4. Реализовать расширенный бинарный алгоритм Евклида.

Теоретическое введение

Классический алгоритм Евклида:

Особенности:

- Основан на делении с остатком.
- Простейший и исторически первый вариант.

Алгоритм:

- Делим большее число на меньшее, получаем остаток.
- Заменяем большее число на меньшее, а меньшее - на остаток.
- Повторяем, пока остаток не станет равен нулю.

Последний ненулевой остаток - НОД.

Бинарный алгоритм Евклида:

Особенности:

- Основан на битовых операциях (сдвиги, сложение, вычитание).
- Работает быстрее на больших числах, чем классический.

Алгоритм:

- Используем свойства НОД: НОД(2a, 2b) = 2 * НОД(a, b), НОД(2a, b) = НОД(a, b) если b нечетно.
- Делим числа на 2, пока они оба не станут нечетными.
- Вычитаем меньшее из большего, пока они не сравняются.
- Умножаем результат на степени двойки, на которые мы делили.

Расширенный алгоритм Евклида:

Особенности:

- Находит не только НОД, но и коэффициенты x, y такие, что ax + by = НОД(a, b).
- Важен для решения диофантовых уравнений и работы с модульной арифметикой.

Алгоритм:

- Выполняем классический алгоритм, сохраняя промежуточные результаты.
- Выражаем НОД через исходные числа, используя промежуточные результаты.

Расширенный бинарный алгоритм Евклида:

Особенности:

- Сочетает в себе преимущества бинарного и расширенного алгоритмов.
- Эффективен и находит коэффициенты x, y.

Алгоритм:

- Выполняем бинарный алгоритм, сохраняя промежуточные результаты.
- Выражаем НОД через исходные числа, используя промежуточные результаты.

Ход работы

Задание 1

```julia
# 1. Алгоритм Евклида
function Euclid_alg(a::Int, b::Int)
    if b == 0
        return (a, 1, 0)
    else
        g, x1, y1 = Euclid_alg(b, a % b)
        y, x = x1 - (a ÷ b) * y1, y1
        return (g, x, y)
    end
end

tests = [30 12; 4 0; 7 1; 125 25; 13 31; 450 45; 3140 720; 330 18]
for i in 1:size(tests)[1]
    t = Euclid_alg(tests[i, 1], tests[i, 2])
    println("Тест ", i ,") Наибольший делитель для ", tests[i, 1]," и ", tests[i, 2],":  ", t[1])
end
```

Рис. 1: Классический алгоритм Евклида

Результат 1

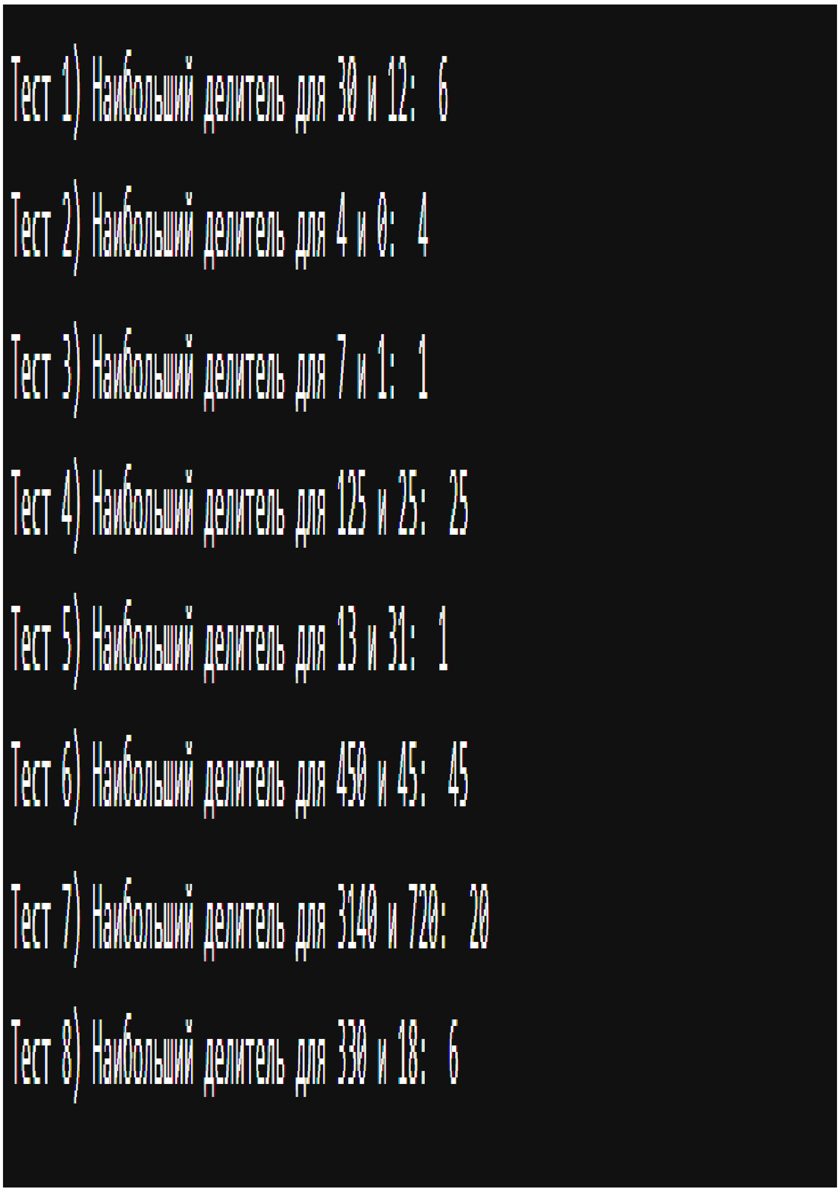

Рис. 2: Результат алгоритма Евклида

Задание 2

```julia
# 2. Бинарный алгоритм Евклида
function bin_Euclid_alg(a::Int, b::Int)
    if a == 0
        return (abs(b), 0, sign(b))
    elseif b == 0
        return (abs(a), sign(a), 0)
    end
    a, b, shift = abs(a), abs(b), 0

    while iseven(a) && iseven(b)
        a >>= 1
        b >>= 1
        shift += 1
    end

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

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

```

Рис. 3: Бинарный алгоритм Евклида 1

```
    while u != v
        if iseven(v)
            v >>= 1
            if iseven(C) && iseven(D)
                C >>= 1
                D >>= 1
            else
                C = (C + b) >> 1
                D = (D - a) >> 1
            end
        elseif v < u
            u, A, B, v, C, D = v, C, D, u, A, B
        else
            v -= u
            C -= A
            D -= B
        end
    end

    return (u << shift, A, B)
end

tests = [30 12; 4 0; 7 1; 125 25; 13 31; 450 45; 3140 720; 330 18]
for i in 1:size(tests)[1]
    t = bin_Euclid_alg(tests[i, 1], tests[i, 2])
    println("Тест ", i ,") Наибольший делитель для ", tests[i, 1]," и ", tests[i, 2],":  ", t[1])
end
```

Рис. 4: Бинарный алгоритм Евклида 2

Результат 2

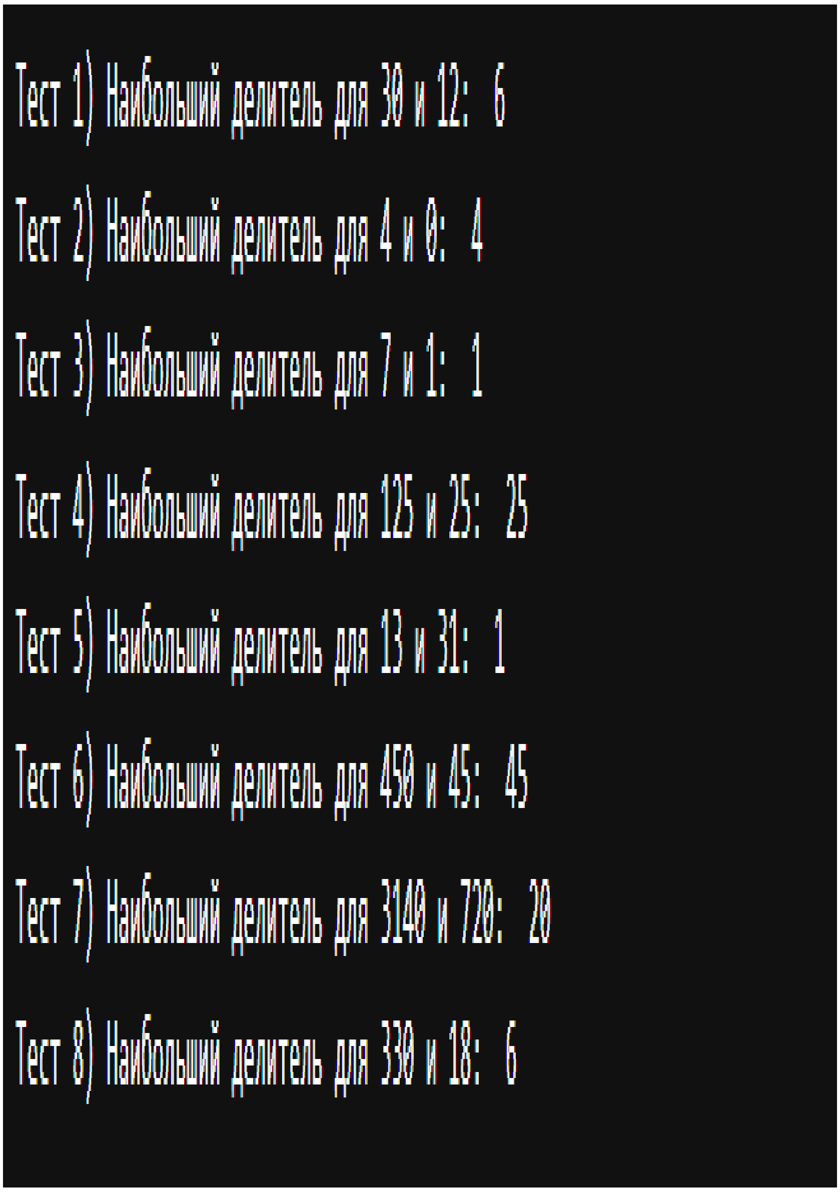

Рис. 5: Результат бинарный Евклида

Задание 3

```julia
# 3. Расширенный алгоритм Евклида
function ext_Euclid_alg(a::Int, b::Int)
    if b == 0
        return (a, 1, 0)
    else
        g, x1, y1 = ext_Euclid_alg(b, a % b)
        x = y1
        y = x1 - (a ÷ b) * y1
        return (g, x, y)
    end
end

tests = [30 12; 4 0; 7 1; 125 25; 13 31; 450 45; 3140 720; 330 18]
for i in 1:size(tests)[1]
    t = ext_Euclid_alg(tests[i, 1], tests[i, 2])
    println("Тест ", i ,") Наибольший делитель для ", tests[i, 1]," и ", tests[i, 2],": ", t)
end
```

Рис. 6: Расширенный алгоритм Евклида

Результат 3

```
Тест 1) Наибольший делитель для 30 и 12:  (6, 1, -2)
Тест 2) Наибольший делитель для 4 и 0:  (4, 1, 0)
Тест 3) Наибольший делитель для 7 и 1:  (1, 0, 1)
Тест 4) Наибольший делитель для 125 и 25:  (25, 0, 1)
Тест 5) Наибольший делитель для 13 и 31:  (1, 12, -5)
Тест 6) Наибольший делитель для 450 и 45:  (45, 0, 1)
Тест 7) Наибольший делитель для 3140 и 720:  (20, -11, 48)
Тест 8) Наибольший делитель для 330 и 18:  (6, 1, -18)
```

Рис. 7: Результат расширенного Евклида

Задание 4

```
# 4. Расширенный бинарный алгоритм Евклида
function ext_bin_Euclid_alg(a::Int, b::Int)
    if a == 0
        return (abs(b), 0, sign(b))
    elseif b == 0
        return (abs(a), sign(a), 0)
    end
    a, b, shift = abs(a), abs(b), 0

    while iseven(a) && iseven(b)
        a >>= 1
        b >>= 1
        shift += 1
    end

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

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

```

Рис. 8: Расширенный бинарный алгоритм Евклида 1

```
    while u != v
        if iseven(v)
            v >>= 1
            if iseven(C) && iseven(D)
                C >>= 1
                D >>= 1
            else
                C = (C + b) >> 1
                D = (D - a) >> 1
            end
        elseif v < u
            u, A, B, v, C, D = v, C, D, u, A, B
        else
            v -= u
            C -= A
            D -= B
        end
    end

    return (u << shift, A, B)
end

tests = [30 12; 4 0; 7 1; 125 25; 13 31; 450 45; 3140 720; 330 18]
for i in 1:size(tests)[1]
    t = ext_bin_Euclid_alg(tests[i, 1], tests[i, 2])
    println("Тест ", i ,") Наибольший делитель для ", tests[i, 1]," и ", tests[i, 2],":  ", t)
end
```

Рис. 9: Расширенный бинарный алгоритм Евклида 2

Результат 4

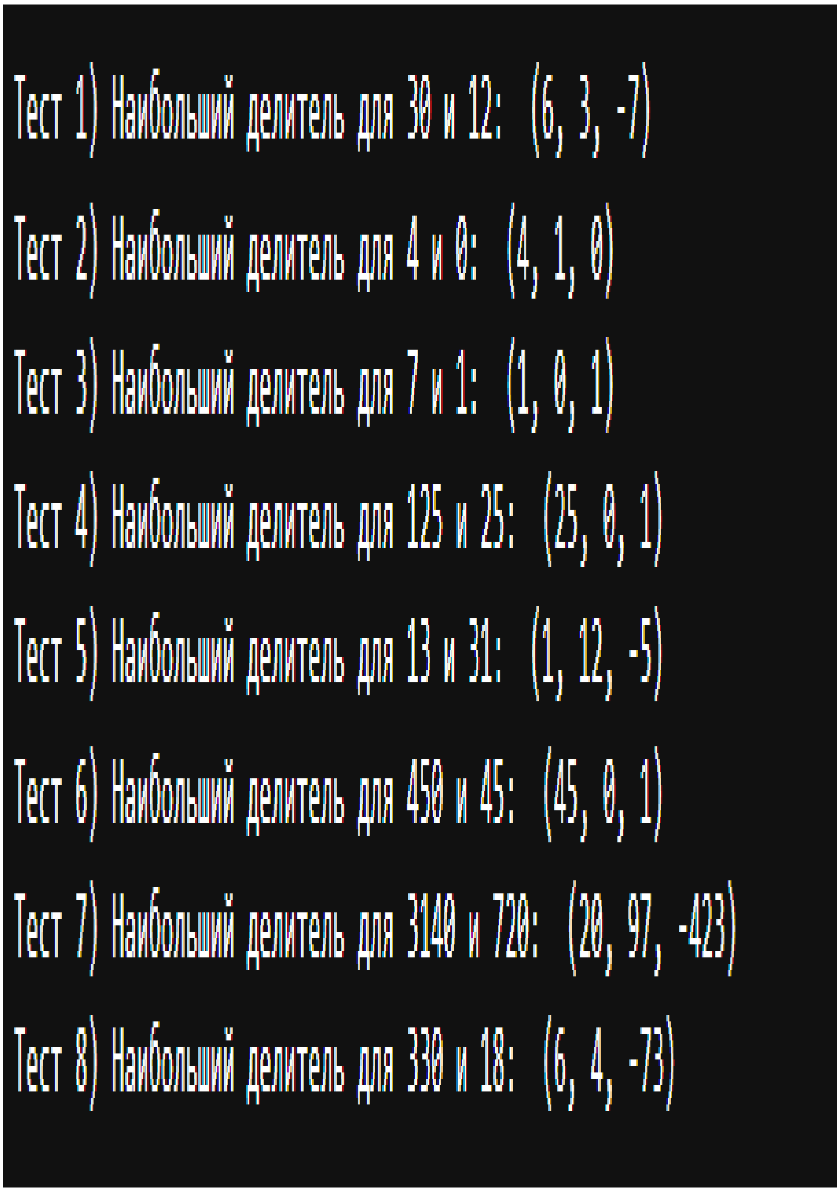

Рис. 10: Результат расширенного бинарного Евклида

Выводы по проделанной работе

Вывод

В ходе выполнения лабораторной работы я ознакомился и реализовал разные варианты алгоритма Евклида для нахождения наибольшего общего делителя. И в результате был сделан очевидный вывод:

- Классический алгоритм - простой и исторически первый.
- Бинарный алгоритм - быстрее на больших числах.
- Расширенный алгоритм - находит коэффициенты x, y.
- Расширенный бинарный алгоритм - сочетает в себе преимущества всех вышеперечисленных.

И есть другие более гибкие и универсальные способы которые часто используют в своей основе методы связанные с алгоритмом Евклида.

Список литературы

1. В очередной раз о НОД, алгоритме Евклида и немного об истории алгоритмов вообще
2. Евклидовы алгоритмы (базовые и расширенные)
3. 8 способов нахождения наибольшего общего делителя
4. Вычисление НОД — ошибка, которой не замечают