Лабораторная работа №4

Вычисление наибольшего общего делителя

Тазаева Анастасия Анатольевна

2025-10-04

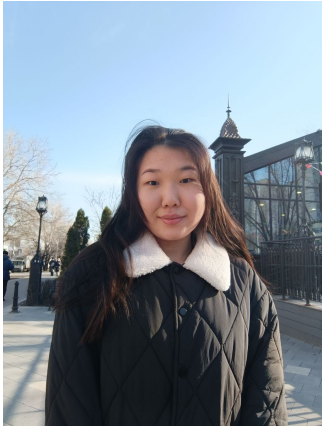
Содержание I

Раздел 1

1. Информация

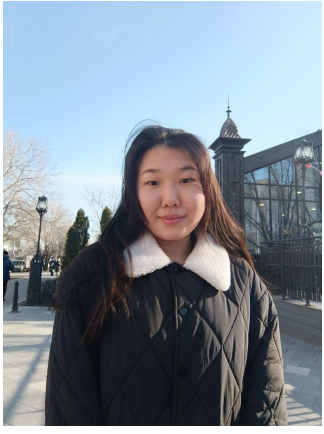
1.1 Докладчик

- Тазаева Анастасия Анатольевна



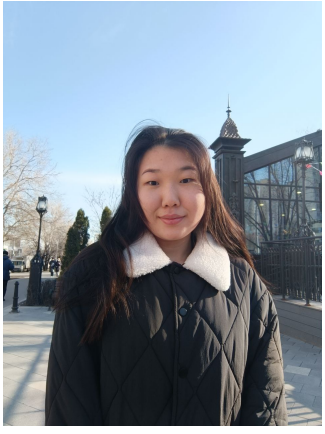
1.1 Докладчик

- Тазаева Анастасия Анатольевна
- студент группы НФИмд-02-25



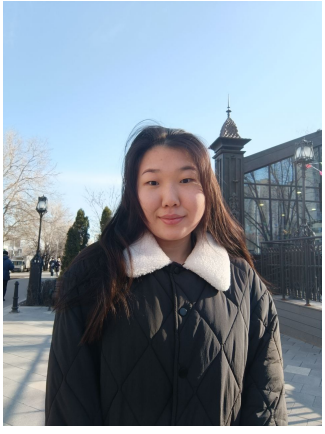
1.1 Докладчик

- Тазаева Анастасия Анатольевна
- студент группы НФИмд-02-25
- Российский университет дружбы народов им. П. Лумумбы



1.1 Докладчик

- Тазаева Анастасия Анатольевна
- студент группы НФИмд-02-25
- Российский университет дружбы народов им. П. Лумумбы
- 1032259385@pfur.ru



Раздел 2

2. Введение

2.1 Цель работы

Ознакомиться с алгоритмами вычисления наибольшего общего делителя.
Реализовать их.

2.2 Задачи

Реализовать на языке программирования Julia:

1. Алгоритм Евклида

2.2 Задачи

Реализовать на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида

2.2 Задачи

Реализовать на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида
3. Расширенный алгоритм Евклида

2.2 Задачи

Реализовать на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида
3. Расширенный алгоритм Евклида
4. Расширенный бинарный алгоритм Евклида

2.3 Теоретическое введение

Целое число $d \neq 0$ называется **наибольшим общим делителем** чисел $a_1, a_2, ..., a_k$, если:

1. Все числа делятся на d

2.3 Теоретическое введение

Целое число $d \neq 0$ называется **наибольшим общим делителем** чисел $a_1, a_2, ..., a_k$, если:

1. Все числа делятся на d
2. Любой другой общий делитель делится на d

Раздел 3

3. Программный код

3.1 Алгоритм Евклида

```
function euclidean_gcd(a::Int, b::Int)
    if b <= 0 || a < b
        error("error a, b, please use 0 < b <= a")
    end
    r0 = a
    r1 = b
    while r1 != 0
        r1_plus_1 = r0 % r1 #vy4islenie ostatka
        r0 = r1 #obnovlenie zna4enii
        r1 = r1_plus_1
    end
    return r0 #GCD
end
```

3.2 Алгоритм Евклида

```
println("a=100, b=2, gcd=",euclidean_gcd(100,2))
println("a=99, b=10, gcd=",euclidean_gcd(99,10))
println("a=99, b=99, gcd=",euclidean_gcd(99,99))
println("a=99, b=51, gcd=",euclidean_gcd(99,51))
println("a=7, b=3, gcd=",euclidean_gcd(7,3))
```

```
a=100, b=2, gcd=2
a=99, b=10, gcd=1
a=99, b=99, gcd=99
a=99, b=51, gcd=3
a=7, b=3, gcd=1
```

Рисунок 1: Алгоритм Евклида. Пример отработки

3.3 Бинарный алгоритм Евклида

```
function binary_gcd(a::Int, b::Int)
    if b <= 0 || a < b
        error("error a, b, please use 0 < b <= a")
    end
    g = 1 #mnojitel dlya u4eta obshih 2
    while a % 2 == 0 && b % 2 == 0
        a /= 2
        b /= 2
        g *= 2
    end
    u = a #first num
    v = b # second num
    while u != 0
        # udalyaem mnojiteli 2 is u
```

3.4 Бинарный алгоритм Евклида

```
println("a=100, b=2, gcd=",binary_gcd(100,2))
println("a=99, b=10, gcd=",binary_gcd(99,10))
println("a=99, b=99, gcd=",binary_gcd(99,99))
println("a=99, b=51, gcd=",binary_gcd(99,51))
println("a=7, b=3, gcd=",binary_gcd(7,3))
```

```
a=100, b=2, gcd=2.0
a=99, b=10, gcd=1.0
a=99, b=99, gcd=99
a=99, b=51, gcd=3.0
a=7, b=3, gcd=1.0
```

Рисунок 2: Бинарный алгоритм Евклида. Пример отработки

3.5 Расширенный алгоритм Евклида

```
function extended_gcd(a::Int, b::Int)
    if b <= 0 || a < b
        error("error a, b, please use 0 < b <= a")
    end
    #xa+yb=gcd
    r0, r1 = a, b
    x0, x1 = 1, 0
    y0, y1 = 0, 1
    while r1 != 0
        q = div(r0, r1) #4astnoe
        r_next = r0 - q * r1 #sled.ostatok
        x_next = x0 - q * x1 #koeff x
        y_next = y0 - q * y1 #koeff y
        #sdvig
```

3.6 Расширенный алгоритм Евклида

```
extended_gcd (generic function with 1 method)
```

```
println("a=100, b=2, gcd, x, y=",extended_gcd(100,2))
println("a=99, b=10, gcd, x, y=",extended_gcd(99,10))
println("a=99, b=99, gcd, x, y=",extended_gcd(99,99))
println("a=99, b=51, gcd, x, y=",extended_gcd(99,51))
println("a=7, b=3, gcd, x, y=",extended_gcd(7,3))
```

```
a=100, b=2, gcd, x, y=(2, 0, 1)
a=99, b=10, gcd, x, y=(1, -1, 10)
a=99, b=99, gcd, x, y=(99, 0, 1)
a=99, b=51, gcd, x, y=(3, -1, 2)
a=7, b=3, gcd, x, y=(1, 1, -2)
```

Рисунок 3: Расширенный алгоритм Евклида. Пример отработки

3.7 Расширенный бинарный алгоритм Евклида

```
function extended_binary_gcd(a::Int, b::Int)
    if b <= 0 || a < b
        error("error a, b, please use 0 < b <= a")
    end
    g = 1 #mnojitel dlya u4eta obshih 2
    u = a #first num
    v = b # second num
    #koeff dlya lin.predstavleniya
    A = 1
    B = 0
    C = 0
    D = 1
    while u % 2 == 0 && v % 2 == 0
        u /= 2
```

3.8 Расширенный бинарный алгоритм Евклида

```
# obrabotka 4etnosti v
        while v % 2 == 0
            v ÷= 2  #
            # obnovlyaem koeff C D
            if C % 2 == 0 && D % 2 == 0
                C ÷= 2  # if oba 4et - to delim na 2
                D ÷= 2
            else
                C = (C + b) ÷ 2  # ina4e primenyam sled formu
                D = (D - a) ÷ 2
            end
        end
        if u >= v
            u = u - v  #
```

3.9 Расширенный бинарный алгоритм Евклида

```
extended_binary_gcd (generic function with 1 method)
```

```
println("a=100, b=2, gcd, x, y=",extended_binary_gcd(100,2))
println("a=99, b=10, gcd, x, y=",extended_binary_gcd(99,10))
println("a=99, b=99, gcd, x, y=",extended_binary_gcd(99,99))
println("a=99, b=51, gcd, x, y=",extended_binary_gcd(99,51))
println("a=7, b=3, gcd, x, y=",extended_binary_gcd(7,3))
```

```
a=100, b=2, gcd, x, y=(2.0, 0, 1)
a=99, b=10, gcd, x, y=(1, -1, 10)
a=99, b=99, gcd, x, y=(99, 0, 1)
a=99, b=51, gcd, x, y=(3, -1, 2)
a=7, b=3, gcd, x, y=(1, 1, -2)
```

Рисунок 4: Расширенный бинарный алгоритм Евклида. Пример отработки

Раздел 4

4. Заключение

4.1 Вывод

В ходе лабораторной работы реализованы на языке программирования Julia:

1. Алгоритм Евклида

4.1 Вывод

В ходе лабораторной работы реализованы на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида

4.1 Вывод

В ходе лабораторной работы реализованы на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида
3. Расширенный алгоритм Евклида

4.1 Вывод

В ходе лабораторной работы реализованы на языке программирования Julia:

1. Алгоритм Евклида
2. Бинарный алгоритм Евклида
3. Расширенный алгоритм Евклида
4. Расширенный бинарный алгоритм Евклида