Математические основы защиты информации и информационной безопасности

Супонина Анастасия Павловна

5 Октября 2024

РУДН, Москва, Россия

Лабораторная работа 4

Цель работы

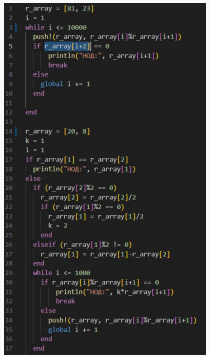
Для нахождения наибольшего общего делителя ознакомиться 4 различными методами и написать программу для каждого из них, а именно для Алгоритма Евклида, Бинарного алгоритма Евклида, Расширенного алгоритма Евклида и Расширенного Бинарного алгоритма Евклида.

Задание

Программно реализовать на языке Julia следующие алгоритмы:

1. Алгоритма Евклида
2. Бинарный алгоритма Евклида
3. Расширенный алгоритм Евклида
4. Расширенный Бинарный алгоритм Евклида

Общий вид программы



```
r_array = [81, 23]
i = 1
while i <= 10000
  push!(r_array, r_array[i]%r_array[i+1])
  if r_array[i+2] == 0
    println("НОД:", r_array[i+1])
    break
  else
    global i += 1
  end

end

r_array = [20, 8]
k = 1
i = 1
if r_array[1] == r_array[2]
  println("НОД:", r_array[1])
else
  if (r_array[2]%2 == 0)
    r_array[2] = r_array[2]/2
    if (r_array[1]%2 == 0)
      r_array[1] = r_array[1]/2
      k = 2
    end
  elseif (r_array[1]%2 != 0)
    r_array[1] = r_array[1]-r_array[2]
  end
  while i <= 1000
    if r_array[i]%r_array[i+1] == 0
      println("НОД:", k*r_array[i+1])
      break
    else
      push!(r_array, r_array[i]%r_array[i+1])
      global i += 1
    end
  end
```

Рис. 1: Общий вид программы

Общий вид программы

```
r_array = [20, 8]
g = 1

while (r_array[1] %2 == 0) && (r_array[2] % 2 == 0)
    r_array[1] /= 2
    r_array[2] /= 2
    global  g *= 2
end

u = r_array[1]
v = r_array[2]
while u != 0
    while (u%2 != 1) && (v%2 != 1)
        if u%2 == 0
            global u /= 2
        else
            global v /= 2
        end
    end
    if u >= v
        global u -= v
    else
        global v -= u
    end
end

d = Int(g*v)
println("НОД:", d)
```

Рис. 2: Общий вид программы

Общий вид программы

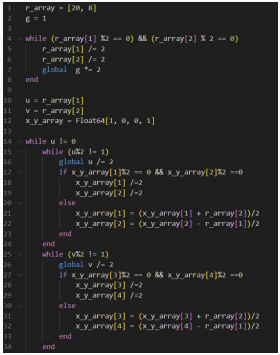
```
r_array = [20, 8]
x_array = Float64[1, 0]
y_array = Float64[0, 1]
i = 1

while r_array[i]%r_array[i+1] != 0
    global q = div(r_array[i], r_array[i+1])
    push!(r_array, r_array[i]%r_array[i+1])
    push!(x_array, x_array[i]-q*x_array[i+1])
    push!(y_array, y_array[i]-q*y_array[i+1])
    global i += 1
end
💡
println("d(НОД):", r_array[i+1])
println("x:", x_array[i+1])
println("y:", y_array[i+1])
```

Рис. 3: Общий вид программы

Расширенный бинарный алгоритм Евклида.

Общий вид программы

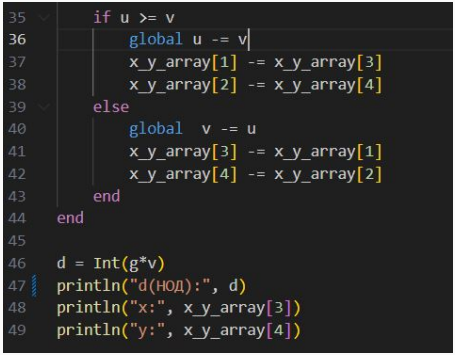


```
r_array = [20, 8]
g = 1

while (r_array[1] %2 == 0) && (r_array[2] % 2 == 0)
    r_array[1] /= 2
    r_array[2] /= 2
    global  g *= 2
end

u = r_array[1]
v = r_array[2]
x_y_array = Float64[1, 0, 0, 1]

while u != 0
    while (u%2 != 1)
        global u /= 2
        if x_y_array[1]%2 == 0 && x_y_array[2]%2 ==0
            x_y_array[1] /=2
            x_y_array[2] /=2
        else
            x_y_array[1] = (x_y_array[1] + r_array[2])/2
            x_y_array[2] = (x_y_array[2] - r_array[1])/2
        end
    end
    while (v%2 != 1)
        global v /= 2
        if x_y_array[3]%2 == 0 && x_y_array[4]%2 ==0
            x_y_array[3] /=2
            x_y_array[4] /=2
        else
            x_y_array[3] = (x_y_array[3] + r_array[2])/2
            x_y_array[4] = (x_y_array[4] - r_array[1])/2
        end
    end
```



```
    if u >= v
        global u -= v
        x_y_array[1] -= x_y_array[3]
        x_y_array[2] -= x_y_array[4]
    else
        global  v -= u
        x_y_array[3] -= x_y_array[1]
        x_y_array[4] -= x_y_array[2]
    end
end

d = Int(g*v)
println("d(НОД):", d)
println("x:", x_y_array[3])
println("y:", x_y_array[4])
```

Выводы

В процессе выполнения работы, я разобралась с принципом работы алгоритмов Евклида.

Реализовала разные виды алгоритмов на языке программирования Julia.

Спасибо за внимание!