Отчёт по лабораторной работе 4

Супонина Анастасия Павловна

Содержание

Список иллюстраций

Список таблиц

Элементы списка иллюстраций не найдены.

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

Теоретическое введение

Для вычисления наибольшего общего делителя двух целых чисел применяется способ повторного деления с остатком называемый алгоритмом Евклида. Рассмотрим действия всех алгоритмов на практике.

Выполнение лабораторной работы

1. **Алгоритм Евклида.**

1. Записываю значения a, b в r_array.

```
r_array = [81, 23]
```

Значения

2. Нахожу остаток от деления.

```
push!(r_array, r_array[i]%r_array[i+1])
```

Остаток

3. При полученном значении равном нулю вывожу НОД, в противном случае повторяю ещё раз шаг 2.

```
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
```

Остаток

```
4. Вывожу результат.
```

```
НОД:1
```

результат

Бинарный алгоритм Евклида является более быстрым при реализации на компьютере, поскольку использует двоичное представление чисел a и b. Бинарныи алгоритм Евклида основан на следующих свойствах наибольшего общего делителя (считаем, что $ 0 < b <=a $):

```
1. если оба числа a и b четные
2. если число a – нечетное, а число b – четное
3. если оба числа a и b нечетные
4. если a = b
```

```
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
```

все условия

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
```

Общий вид программы

2. **Бинарный алгоритм Евклида.**

    1. Записываю g = 1, a, b

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

Записываю переменные

2. Пока оба числа a и b четные, делю их на 2 и умножаю на 2 g при каждой иттерации, до получения хотя бы одного нечетного значения a или b.

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

получения хотя бы одного нечетного значения

3. Записываю u и v.

```
u = r_array[1]
v = r_array[2]
```

Записываю u и v

4. Провожу преобразования пока u не будет равно 0.

```
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
```

преобразования пока u не будет равно 0

5. Записываю результат в d.

```
d = Int(g*v)
println("НОД:", d)
```

Записываю результат в d

6. Вывожу результат

```
НОД:4

julia>
```

результат

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)
```

Общий вид программы

3. **Расширенный алгоритм Евклида.**

    1. Записываю значения a, b, а также значения для x и y.

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

Записываю значения

2. Вычисляю остаток и целочисленное частное от деления.

```
global q = div(r_array[i], r_array[i+1])
push!(r_array, r_array[i]%r_array[i+1])
```

Вычисляю остаток и целочисленное частное от деления

3. Провожу преобразования пока не получу остаток равный 0.

```
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
```

Провожу преобразования

4. Вывожу результат.

```
println("d(НОД):", r_array[i+1])
println("x:", x_array[i+1])
println("y:", y_array[i+1])
```

результат

Общий вид программы

```
1   r_array = [20, 8]
2   x_array = Float64[1, 0]
3   y_array = Float64[0, 1]
4   i = 1
5
6   while r_array[i]%r_array[i+1] != 0
7       global q = div(r_array[i], r_array[i+1])
8       push!(r_array, r_array[i]%r_array[i+1])
9       push!(x_array, x_array[i]-q*x_array[i+1])
10      push!(y_array, y_array[i]-q*y_array[i+1])
11      global i += 1
12  end
13
14  println("d(НОД):", r_array[i+1])
15  println("x:", x_array[i+1])
16  println("y:", y_array[i+1])
```

Общий вид программы

4. **Расширенный бинарный алгоритм Евклида.**

    1. Записываю g = 1, a, b

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

Записываю переменные

2. Пока оба числа a и b четные, делю их на 2 и умножаю на 2 g при каждой иттерации, до получения хотя бы одного нечетного значения a или b.

```
4   while (r_array[1] %2 == 0) && (r_array[2] % 2 == 0)
5       r_array[1] /= 2
6       r_array[2] /= 2
7       global  g *= 2
8   end
```

получения хотя бы одного нечетного значения

3. Записываю значения u, v, а также отдельно значения A, B, C и D связанные с u и v.

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

Записываю значения

4. Провожу преобразования пока u не будет равно 0.

    1. Пока u четное.
    2. Пока v четное.

```
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
```

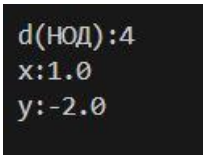
Провожу преобразования пока и не будет равно 0

5. Записываю полученные значения

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

Записываю полученные значения

6. Вывожу результат



результат

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.

Список литературы

::: Пособие по лабораторной работе 3 {file:///C:/Users/bermu/Downloads/lab03.pdf}