

# Отчёт по лабораторной работе №4

---

Аветисян Давид Артурович

26 октября 2024

РУДН, Москва, Россия

- Реализовать алгоритмы вычисления наибольшего общего делителя.

# Алгоритм Евклида на языке Julia

- 1-5 строки: задание чисел, НОД которых ищем.
- 6-12 строки: реализация самого алгоритма Евклида: делим наибольшее число на наименьшее и записываем остаток до тех пор, пока одно из них не обнулится.
- 14-15 строки: записываем НОД в переменную и выводим.

```
[11]: number1 = 32
      number2 = 26

[11]: 26

[12]: num1 = number1
      num2 = number2
      while num1 != 0 && num2 != 0
          if num1 >= num2
              num1 = num1 % num2
          else
              num2 = num2 % num1
          end
      end
      nod1 = num1 + num2
      println(nod1)

      2
```

Рис. 1: Алгоритм Евклида на языке Julia

# Бинарный алгоритм Евклида на языке Julia

- 1-3 строки: задание чисел, НОД которых ищем, и обнуление “сдвига”.
- 5-23 строки: реализация самого бинарного алгоритма Евклида: смотрим на четность получающихся значений и записываем, насколько нам необходимо “сдвинуть” число, чтобы получить НОД.
- 25 строка: сдвиг влево и вывод получившегося НОД.

```
[13]: num1 = number1
      num2 = number2
      shift = 0

      while (num1 | num2) & 1 == 0
          shift += 1
          num1 >>= 1
          num2 >>= 1
      end

      while num1 & 1 == 0
          num1 >>= 1
      end

      while num2 & 1 == 0
          while num2 & 1 == 0
              num2 >>= 1
          end
          if num1 > num2
              num1, num2 = num2, num1
          end
          num2 >>= num1
      end

      println(num1 << shift)
      2
```

Рис. 2: Бинарный алгоритм Евклида на языке Julia

# Расширенный алгоритм Евклида на языке Julia

- 1 строка: задание рекурсивной функции.
- 2-3 строки: если второе число равно нулю, возвращаем ответ из трёх чисел.
- 5-7 строки: в ином случае запускаем рекурсию, а затем выводим ответ согласно формуле на строке 7.
- 8 строка: возвращаем вывод в качестве получившегося НОД; числа, что нужно домножить на первую цифру и на вторую, чтобы получить НОД.
- 11 строка: вызов функции и сохранение данных в переменные.
- 12 строка: вывод на экран.

```
[14]: function ext_evk(a, b)
      if b == 0
          return a, 1, 0
      else
          mod, x1, y1 = ext_evk(b, a % b)
          x = y1
          y = x1 - div(a, b) * y1
          return mod, x, y
      end
end

mod, x, y = ext_evk(number1, number2)
println(x, " ", number3, " ", '{', y, '}', " ", number2, " ", mod)

-2*12*(1)*26=2
```

Рис. 3: Расширенный алгоритм Евклида на языке Julia

# Расширенный бинарный алгоритм Евклида на языке Julia

Данная программа работает рекурсивно, рассматривая 4 случая:

1. a четное
2. a нечетное и b четное
3. a нечетное и b нечетное,  $b > a$
4. a нечетное и b нечетное,  $a > b$

Каждая рекурсия сдвигает биты в цифрах, формирую окончательный ответ. В итоге выводит три значения: НОД; числа, которые нужно домножить на первую цифру и на вторую, чтобы получить НОД.

```
[25]: function ext_gcd(a, b)
    if a == 0
        return 0, 1, b
    elseif b == 0
        return 1, 0, a
    elseif (a & 1) == 0 && (b & 1) == 0
        x, y, g = ext_gcd(a >> 1, b >> 1)
        return x, y, g << 1
    elseif (a & 1) == 0
        x, y, g = ext_gcd(a >> 1, b)
        if (x & 1) == 1
            return x - (b >> 1), y + (a >> 1), g
        else
            return x >> 1, y, g
        end
    elseif (b & 1) == 0
        x, y, g = ext_gcd(a, b >> 1)
        if (y & 1) == 1
            return x + (b >> 1), y - (a >> 1), g
        else
            return x, y >> 1, g
        end
    elseif b > a
        x, y, g = ext_gcd(a, b - a)
        return x - y, y, g
    else
        x, y, g = ext_gcd(a - b, b)
        return x, y + x, g
    end
end
```

- Я реализовал алгоритмы вычисления наибольшего общего делителя.