Документ

# Цель работы

Цель данной лабораторной работы — изучить различные алгоритмы для вычисления наибольшего общего делителя (НОД) двух целых чисел, а также освоить их программную реализацию на языке Julia . В рамках лабораторной работы необходимо реализовать алгоритм Евклида, бинарный алгоритм Евклида, расширенный алгоритм Евклида, а также расширенный бинарный алгоритм Евклида.

# Задание

1. **Реализовать алгоритм Евклида для нахождения наибольшего общего делителя.**
2. **Реализовать бинарный алгоритм Евклида.**
3. **Реализовать расширенный алгоритм Евклида.**
4. **Реализовать расширенный бинарный алгоритм Евклида.**

# Теоретическое введение

### Определение наибольшего общего делителя

Наибольший общий делитель (НОД) двух или более целых чисел — это наибольшее целое число ( d ), которое делит эти числа без остатка. Если числа ( a\_1, a\_2, …, a\_k ) имеют НОД равный ( d ), то выполняются следующие условия: 1. Каждое из чисел ( a\_1, a\_2, …, a\_k ) делится на ( d ). 2. ( d ) является наибольшим числом, которое удовлетворяет первому условию.

### Алгоритм Евклида

Алгоритм Евклида используется для нахождения наибольшего общего делителя (НОД) двух целых чисел

#### Пошаговый алгоритм Евклида

1. **Инициализация**:
   * Задаем начальные значения:
2. **Деление с остатком**:
   * Находим остаток от деления:
3. **Проверка на завершение**:
4. **Результат**:

### Бинарный алгоритм Евклида (поиск наибольшего общего делителя)

Бинарный алгоритм Евклида — это улучшенная версия классического алгоритма Евклида, которая быстрее выполняется на компьютерах благодаря использованию операций сдвига для работы с двоичными представлениями чисел.

#### Пошаговое описание алгоритма

1. **Инициализация множителя**:
2. **Удаление общих множителей 2 из ( a ) и ( b )**:
   * Пока оба числа ( a ) и ( b ) четные, делим их на 2 и умножаем ( g ) на 2:
   * Этот шаг выполняется до тех пор, пока одно из чисел не станет нечетным.
3. **Инициализация переменных для основного цикла**:
4. **Основной цикл**:
   * Пока

* **Шаг 4.1**:
* **Шаг 4.2**:

$$
\text{Если \( v \) четное, то делим его на 2: } v\leftarrow \frac{v}{2}
$$

* **Шаг 4.3**:

1. **Вычисление НОД**:
2. **Результат**:
   * Итоговое значение d является наибольшим общим делителем чисел a и b.

Бинарный алгоритм Евклида эффективен для выполнения на компьютере, поскольку использует операции деления и вычитания, которые хорошо работают с двоичными числами, минимизируя количество арифметических операций.

**Пример выполнения:** Пусть даны числа ( a = 12345 ) и ( b = 24690 ). В ходе выполнения алгоритма Евклида последовательно находим остатки от деления, пока одно из чисел не станет нулём.

### Расширенный алгоритм Евклида

Расширенный алгоритм Евклида позволяет не только найти НОД двух чисел, но и найти такие целые числа ( x ) и ( y ), что ( ax + by = НОД(a, b) ). Это уравнение называется линейной комбинацией. Расширенный алгоритм полезен для задач, где требуется найти коэффициенты линейной комбинации, например, в криптографии.

### Расширенный бинарный алгоритм Евклида

### Расширенный алгоритм Евклида (поиск НОД и коэффициентов линейного представления)

Расширенный алгоритм Евклида не только находит наибольший общий делитель (НОД) чисел ( a ) и ( b ), но и определяет целые числа ( x ) и ( y ) такие, что выполняется равенство:

#### Пошаговое описание алгоритма

1. **Инициализация**:
   * Задаем начальные значения:
2. **Деление с остатком**:
   * Разделяем:
3. **Проверка на завершение**: $$ r\_{i+1} = 0, \ d r\_i, x x\_i, y y\_i \

r\_{i+1} , \ x\_{i+1} x\_{i-1} - q\_i x\_i, y\_{i+1} y\_{i-1} - q\_i y\_i $$ - Затем увеличиваем ( i ) на 1 и возвращаемся к шагу 2.

1. **Результат**:
   * После завершения алгоритма ( d ), ( x ), и ( y ) будут такими, что:

### Расширенный бинарный алгоритм Евклида

Расширенный бинарный алгоритм Евклида позволяет не только находить наибольший общий делитель (НОД) двух чисел ( a ) и ( b ), но также определяет коэффициенты ( x ) и ( y ) для линейного представления:

где

#### Пошаговое описание алгоритма

1. **Инициализация множителя**:
   * Устанавливаем ( g ⟵ 1 ). Этот множитель будет использоваться для учета общего множителя 2, если оба числа ( a ) и ( b ) четные.
2. **Удаление общих множителей 2 из ( a ) и ( b )**:
   * Пока оба числа ( a ) и ( b ) четные, делим их на 2 и удваиваем ( g ):
   * Этот процесс продолжается до тех пор, пока одно из чисел не станет нечетным.
3. **Инициализация переменных для расширенного алгоритма**:
   * Задаем начальные значения:
4. **Основной цикл**:

$$
\begin{align\*}
&\text{- Пока } u \neq v, \text{ выполняем следующие действия:} \\
&\quad \text{- Шаг 4.1: Если } u \text{ четное, то:} \\
&\quad\quad \text{4.1.1 Если } u \text{ четное, делим его на 2:} \\
&\quad\quad u \leftarrow \frac{u}{2} \\
&\quad\quad \text{4.1.2 Если оба числа } A \text{ и } B \text{ четные, делим их на 2:} \\
&\quad\quad A \leftarrow \frac{A}{2}, \quad B \leftarrow \frac{B}{2} \\
&\quad\quad \text{В противном случае:} \\
&\quad\quad A \leftarrow \frac{A + b}{2}, \quad B \leftarrow \frac{B - a}{2}
\end{aligned}
$$

* $$
* $$

1. **Вывод результата**:
   * После завершения цикла устанавливаем:
2. **Результат**:
   * Итоговое значение ( d ), ( x ), и ( y ) такое, что:

# Выполнение лабораторной работы

### Реализация алгоритма Евклида

Алгоритм Евклида начинается с инициализации чисел ( r\_0 = a ) и ( r\_1 = b ). Затем производится деление с нахождением остатка, пока остаток не станет равным нулю. На каждом шаге: - Делим большее число на меньшее. - Запоминаем остаток. - Если остаток равен нулю, то текущее меньшее число является НОД.

**Пример кода:**

a = 4567890  
b = 462  
  
function euclidian\_algorithm(a,b)  
 if a<b  
 a,b=b,a  
 end  
 r\_0 = a  
 r\_1 = b   
 while true  
 r\_next= r\_0 % r\_1  
 if r\_next == 0  
 return r\_1  
 end  
 r\_0 = r\_1  
 r\_1 = r\_next  
 end  
end  
  
println(euclidian\_algorithm(a,b))

6

### Реализация бинарного алгоритма Евклида

Бинарный алгоритм Евклида проверяет чётность обоих чисел. На каждом шаге: 1. Если оба числа чётные, делим их пополам и умножаем результат на 2. 2. Если одно из чисел чётное, делим только его. 3. Если оба нечётные, из большего вычитаем меньшее, делим результат на 2 и повторяем.

**Пример кода:**

a = 678908890  
b = 2937  
  
function euclidian\_binary(a,b)  
 if a<b  
 a,b=b,a  
 end  
 g = 1  
 while a%2==0 && b%2==0  
 a = div(a,2)  
 b = div(a,2)  
 g\*= 2  
 end  
 u = a   
 v = b  
 while u!=0  
 while u%2==0  
 u = div(u,2)  
 end  
 while v%2==0  
 v = div(v,2)  
 end  
 if u>=v  
 u -= v   
 else  
 v -= u  
 end   
 end  
 d = g\*v  
 return d  
end  
  
println(euclidian\_binary(a,b))

11

### Реализация расширенного алгоритма Евклида

В расширенном алгоритме Евклида на каждом шаге вычисления остатка также обновляются коэффициенты ( x ) и ( y ) для линейной комбинации. Процесс продолжается до тех пор, пока один из остатков не станет нулём. На этом этапе значения ( x ) и ( y ) дают линейное представление НОД.

**Пример кода:**

a = 3984759347  
b = 47584  
  
function euclidian\_ext(a,b)  
 if a<b  
 a,b=b,a  
 end  
 r\_0 = a  
 r\_1 = b  
 x\_0=1 # a\*x + b\*y = d (НОД(a,b))  
 x\_1=0   
 y\_0=0  
 y\_1=1  
 while r\_1!=0  
 q = div(r\_0,r\_1)  
 r\_next = r\_0 - q\*r\_1 # НОД(r\_0,r\_1)  
 r\_0 = r\_1  
 r\_1 = r\_next  
  
 x\_next = x\_0 - q\*x\_1  
 x\_0 = x\_1  
 x\_1 = x\_next  
  
 y\_next = y\_0-q\*y\_1  
 y\_0 = y\_1  
 y\_1 = y\_next  
 end   
 d = r\_0  
 x = x\_0  
 y = y\_0  
 return d, x, y  
end  
println(euclidian\_ext(a,b))

(1, 18011, -1508269599)

**Проверка**

d, x, y = euclidian\_ext(a,b)  
check = a\*x+b\*y  
println(check)  
  
  
 1

### Реализация расширенного бинарного алгоритма Евклида

В расширенном бинарном алгоритме Евклида используется тот же процесс, что и в бинарном алгоритме, но также включены вычисления для коэффициентов линейной комбинации. Алгоритм проходит следующие шаги: 1. Инициализируем коэффициенты и выполняем деление пополам, если числа чётные. 2. На каждом шаге вычисления коэффициентов учитываются изменения в значениях чисел. 3. Процесс продолжается до тех пор, пока одно из чисел не станет равным нулю.

**Пример кода:**

a = 1024  
b = 512  
  
function euclidean\_ext\_bin(a, b)  
 if a < b  
 a, b = b, a  
 end  
 g = 1  
 while a % 2 == 0 && b % 2 == 0  
 a = div(a, 2)  
 b = div(b, 2)  
 g \*= 2  
 end  
 u = a  
 v = b  
 A = 1  
 B = 0  
 C = 0  
 D = 1  
 while u != 0  
 while u % 2 == 0  
 u = div(u, 2)  
 if A % 2 == 0 && B % 2 == 0  
 A = div(A, 2)  
 B = div(B, 2)  
 else  
 A = div(A + b, 2)  
 B = div(B - a, 2)  
 end  
 end  
 while v % 2 == 0  
 v = div(v, 2)  
 if C % 2 == 0 && D % 2 == 0  
 C = div(C, 2)  
 D = div(D, 2)  
 else  
 C = div(C + b, 2)  
 D = div(D - a, 2)  
 end  
 end  
 if u >= v  
 u -= v  
 A -= C  
 B -= D  
 else  
 v -= u  
 C -= A  
 D -= B  
 end  
 end  
 d = g \* v  
 x = C  
 y = D  
 return d, x, y  
end  
println(euclidean\_ext\_bin(a,b))  
  
  
 (512, 0, 1)

**Проверка**

d, x, y = euclidean\_ext\_bin(a,b)  
check = a\*x+b\*y   
println(check)

512

# Выводы

В ходе выполнения лабораторной работы были изучены и реализованы 4 вариации *Алгоритма Евклида* для нахождения НОД, включая их расширенные версии. Каждый из алгоритмов имеет свои преимущества и применяется в зависимости от требований к вычислительной эффективности или необходимости в нахождении коэффициентов линейной комбинации. Бинарный алгоритм показал себя как более быстрый для реализации на компьютере, в то время как расширенные алгоритмы обеспечили возможность нахождения дополнительных параметров.

# Список литературы