Отчёт по лабораторной работе №4.  
Вычисление наибольшего общего делителя

Студент: Аронова Юлия Вадимовна, 1032212303

Группа: НФИмд-01-21

Преподаватель: Кулябов Дмитрий Сергеевич,

д-р.ф.-м.н., проф.

Москва 2021

Содержание

[1 Цель работы 1](#_Toc89343742)

[2 Задание 2](#_Toc89343743)

[3 Теоретическое введение 2](#_Toc89343744)

[3.1 Основные понятия теории чисел 2](#_Toc89343745)

[3.2 Алгоритм Евклида 3](#_Toc89343746)

[3.3 Бинарный алгоритм Евклида 3](#_Toc89343747)

[3.4 Расширенный алгоритм Евклида 4](#_Toc89343748)

[3.5 Расширенный бинарный алгоритм Евклида 5](#_Toc89343749)

[4 Выполнение лабораторной работы 5](#_Toc89343750)

[4.1 Алгоритм Евклида 5](#_Toc89343751)

[4.2 Бинарный алгоритм Евклида 6](#_Toc89343752)

[4.3 Расширенный алгоритм Евклида 7](#_Toc89343753)

[4.4 Расширенный бинарный алгоритм Евклида 9](#_Toc89343754)

[5 Выводы 10](#_Toc89343755)

[Список литературы 10](#_Toc89343756)

# 1 Цель работы

Целью данной лабораторной работы является ознакомление с двумя алгоритмами нахождения наибольшего общего делителя: алгоритмом Евклида и бинарным алгоритмом Евклида, – и их расширениями для нахождения линейного представления наибольшего общего делителя, а также их последующая программная реализация.

# 2 Задание

Рассмотреть и реализовать на языке программирования Python:

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

# 3 Теоретическое введение

## 3.1 Основные понятия теории чисел

Опр. 3.1.

Пусть и – целые числа. Говорят, что *делит* (или *делится на* ), если существует такое целое число , что . Обозначение: [1].

Опр. 3.2.

Если и , говорят, что числа и *ассоциированы*. По сути это означает, что или .

Теорема 3.1.

(О делении с остатком). *Пусть . Тогда существуют единственные целые числа (неполное частное) и (остаток) такие, что и .*

Опр. 3.3.

Пусть . Говорят, что целое число является *общим делителем* и , если и .

Опр. 3.4.

Пусть . Целое число называется *наибольшим общим делителем (НОД)* чисел и , если:

* – общий делитель и ;
* если – общий делитель и , то .

Пример.

НОД(12345, 24690) = 12345;

НОД(12345, 54321) = 3;

НОД(12345, 12541) = 1.

Теорема 3.2.

*Наибольший общий делитель двух целых чисел существует и представляется в виде для некоторых целых* .

Опр. 3.5.

Выражение из теоремы 3.2 называется *линейным представлением НОД*.

**Свойства НОД**:

1. .
2. .
3. .

Опр. 3.6.

Числа называются *взаимно простыми*, если .

## 3.2 Алгоритм Евклида

Алгоритм Евклида нахождения наибольшего общего делителя основывается на следующем простом результате [2]:

*Если , то НОД() = НОД().* (1)

**Алгоритм 1. Алгоритм Евклида** [3]

*Вход.* Целые числа ; .

*Выход.*

1. Положить .
2. Найти остаток от деления на .
3. Если , то положить . В противном случае положить и вернуться на шаг 2.
4. Результат: .

Теорема 3.3.

Для любых алгоритм Евклида останавливается, и выдаваемое им число является наибольшим общим делителем чисел и .

Доказательство

По теореме 3.1 для любого имеем , где . Получаем монотонно убывающую последовательность неотрицательных целых чисел , ограниченную снизу. Такая последовательность не может быть бесконечной, следовательно, алгоритм Евклида останавливается. Тогда, по результату (1), получаем .

## 3.3 Бинарный алгоритм Евклида

Бинарный алгоритм Евклида вычисления НОД оказывается более быстрым при реализации этого алгоритма на компьютере, поскольку использует двоичное представление чисел и . Бинарный алгоритм Евклида основан на следующих свойствах наибольшего общего делителя (считаем, что ) [3]: $\vspace{2pt}$

* если оба числа и чётные, то ; $\vspace{2pt}$
* если число – нечётное, число – чётное, то ; $\vspace{2pt}$
* если оба числа и нечётные, то $\vspace{2pt}$
* если , то $\vspace{-3pt}$

**Алгоритм 2. Бинарный алгоритм Евклида** [3]

*Вход.* Целые числа ; .

*Выход.*

1. Положить .
2. Пока оба числа и чётные, выполнять до получения хотя бы одного нечётного значения или .
3. Положить .
4. Пока выполнять следующие действия:

* 4.1. Пока чётное, полагать .
* 4.2. Пока чётное, полагать .
* 4.3. При положить . В противном случае положить .

1. Положить .
2. Результат: .

## 3.4 Расширенный алгоритм Евклида

Помимо наибольшего общего делителя чисел и расширенный алгоритм Евклида также находит его линейное представление, т.е. целые числа и , для которых .

**Алгоритм 3. Расширенный алгоритм Евклида** [3]

*Вход.* Целые числа ; .

*Выход.* ; такие целые числа , что .

1. Положить , .
2. Разделить с остатком на : .
3. Если , то положить , . В противном случае положить , и вернуться на шаг 2.
4. Результат: .

Теорема 3.4.

На каждой итерации алгоритма выполняется равенство при .

Доказательство

Воспользуемся методом математической индукции. Для и требуемое равенство имеет место в силу шага 1. Допустим, что оно справедливо для и для . Тогда на шаге 3 получаем: и . Следовательно, .

## 3.5 Расширенный бинарный алгоритм Евклида

**Алгоритм 4. Расширенный бинарный алгоритм Евклида** [3]

*Вход.* Целые числа ; .

*Выход.* ; такие целые числа , что .

1. Положить .
2. Пока числа и чётные, выполнять до получения хотя бы одного нечётного значения или .
3. Положить .
4. Пока выполнять следующие действия:

* 4.1. Пока чётное:
* 4.1.1. Положить .
* 4.1.2. Если оба числа и чётные, то положить , . В противном случае положить , .
* 4.2. Пока чётное:
* 4.2.1. Положить .
* 4.2.2. Если оба числа и чётные, то положить , . В противном случае положить , .
* 4.3. При положить . В противном случае положить .

1. Положить .
2. Результат: .

# 4 Выполнение лабораторной работы

## 4.1 Алгоритм Евклида

Реализуем вышеописанные алгоритмы на языке **Python** в среде Jupyter Notebook. Начнём с алгоритма Евклида: создадим функцию euclidean\_algorithm(a, b) следующего вида:

def euclidean\_algorithm(a, b):  
 """  
 Находит НОД чисел a и b с помощью алгоритма Евклида  
 """  
 # убеждаемся, что числа - целые и положительные  
 (a, b) = (abs(int(a)), abs(int(b)))  
  
 # по условнию 0 < b <= a, поэтому  
 if b > a: # если оно не выполняется  
 (a, b) = (b, a) # меняем a и b местами  
  
 # поскольку на каждом шаге мы используем только два значения,  
 # сохранять будем только их  
 r = [a, b] # шаг 1; задаем r0 и r1  
  
 # шаги 2-3  
 while r[1] != 0: # пока r\_{i+1} не равно нулю  
 # находим очередной остаток от деления и  
 # имитируем увеличение индекса, сдвигая значения  
 (r[0], r[1]) = (r[1], r[0] % r[1])  
  
 return r[0] # шаг 4; d = r\_i

Теперь с помощью данной функции найдём наибольший общий делитель для нескольких пар чисел (см. Рис. 1).

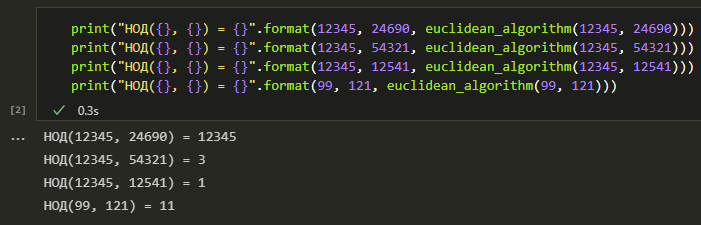


Figure 1: Примеры нахождения НОД двух чисел с помощью программной реализации алгоритма Евклида

## 4.2 Бинарный алгоритм Евклида

Реализуем бинарный алгоритм Евклида: создадим две функции – is\_even(a) для проверки чётности числа и euclidean\_algorithm\_binary(a, b):

def is\_even(a):  
 """  
 Проверяет чётность числа a  
 """  
 return (True if a % 2 == 0 else False)

def euclidean\_algorithm\_binary(a, b):  
 """  
 Находит НОД чисел a и b с помощью бинарного алгоритма Евклида  
 """  
 # убеждаемся, что числа - целые и положительные  
 (a, b) = (abs(int(a)), abs(int(b)))  
  
 # убеждаемся, что выполняется условие 0 < b <= a  
 if b > a:  
 (a, b) = (b, a)  
  
 g = 1 # шаг 1  
  
 # шаг 2; пока числа a и b чётные  
 while is\_even(a) and is\_even(b):  
 (a, b, g) = (int(a / 2), int(b / 2), 2 \* g)  
  
 (u, v) = (a, b) # шаг 3  
  
 # шаг 4  
 while u != 0:  
 # шаг 4.1  
 while is\_even(u): # пока u - чётное  
 u = int(u / 2)  
  
 # шаг 4.2  
 while is\_even(v): # пока v - чётное  
 v = int(v / 2)  
  
 # шаг 4.3  
 if u >= v:  
 u -= v  
 else:  
 v -= u  
  
 return g \* v # шаги 5-6

Теперь с помощью данной функции найдём наибольший общий делитель для нескольких пар чисел (см. Рис. 2).

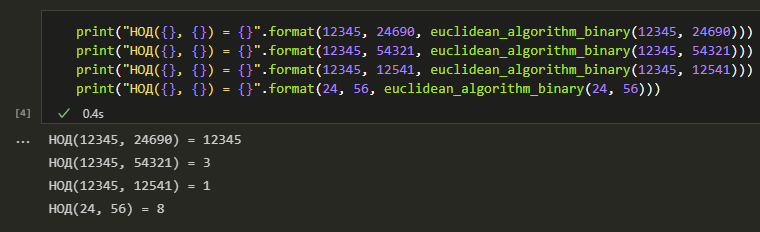


Figure 2: Примеры нахождения НОД двух чисел с помощью программной реализации бинарного алгоритма Евклида

## 4.3 Расширенный алгоритм Евклида

Создадим функцию euclidean\_algorithm\_extended(a, b), реализующую расширенный алгоритм Евклида:

def euclidean\_algorithm\_extended(a, b):  
 """  
 Находит d = НОД(a, b), а также такие целые числа x и y,  
 что ax + by = d, с помощью расширенного алгоритма Евклида  
 """  
 # убеждаемся, что числа - целые и положительные  
 (a, b) = (abs(int(a)), abs(int(b)))  
  
 # убеждаемся, что выполняется условие 0 < b <= a  
 reversed = True if b > a else False # флаг  
 (a, b) = (b, a) if reversed else (a, b) # меняем местами a и b, если нужно  
  
 (r, x, y) = ([a, b], [1, 0], [0, 1]) # шаг 1  
  
 # шаги 2-3  
 # r[0] ~ r\_{i}, r[1] ~ r\_{i+1}  
 while r[1] != 0:  
 # r\_{i-1} = qi \* ri + r\_{i+1}  
 (r[0], r[1], q) = (r[1], r[0] % r[1], r[0] // r[1])  
  
 if r[1] != 0: # если остаток ещё не нулевой..  
 (x[0], x[1]) = (x[1], x[0] - q \* x[1])  
 (y[0], y[1]) = (y[1], y[0] - q \* y[1])  
  
 (d, x\_r, y\_r) = (r[0], x[1], y[1])  
  
 if reversed: # если a и b были в неправильном порядке  
 (x\_r, y\_r) = (y\_r, x\_r) # меняем найденные коэффициенты местами  
  
 return (d, x\_r, y\_r)

Теперь с помощью данной функции найдём наибольший общий делитель и его линейное представление для нескольких пар чисел (см. Рис. 3).

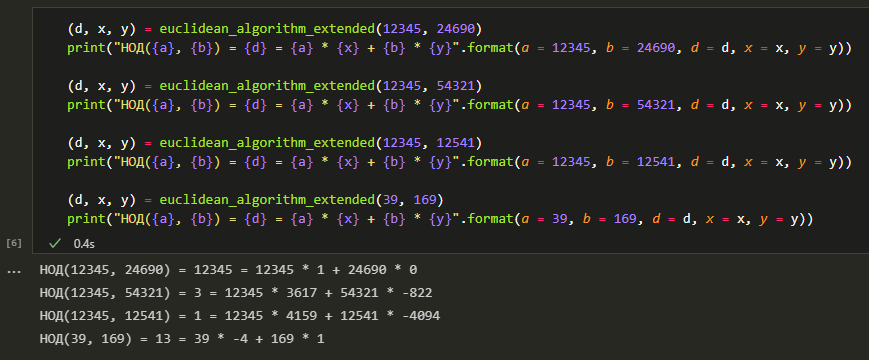


Figure 3: Примеры нахождения НОД двух чисел и его линейного представления с помощью программной реализации расширенного алгоритма Евклида

## 4.4 Расширенный бинарный алгоритм Евклида

Реализуем расширенный бинарный алгоритм Евклида, создав функцию euclidean\_algorithm\_binary\_extended(a, b) следующего вида:

def euclidean\_algorithm\_binary\_extended(a, b):  
 """  
 Находит d = НОД(a, b), а также такие целые числа x и y,  
 что ax + by = d, с помощью расширенного бинарного алгоритма Евклида  
 """  
 # убеждаемся, что числа - целые и положительные  
 (a, b) = (abs(int(a)), abs(int(b)))  
  
 # убеждаемся, что выполняется условие 0 < b <= a  
 reversed = True if b > a else False # флаг  
 (a, b) = (b, a) if reversed else (a, b) # меняем местами a и b, если нужно  
  
 g = 1 # шаг 1  
  
 # шаг 2; пока числа a и b чётные  
 while is\_even(a) and is\_even(b):  
 (a, b, g) = (int(a / 2), int(b / 2), 2 \* g)  
  
 # шаг 3; задаем начальные значения  
 (u, v, A, B, C, D) = (a, b, 1, 0, 0, 1)  
  
 # шаг 4; пока u не равно нулю  
 while u != 0:  
 # шаг 4.1  
 while is\_even(u):  
 u = int(u / 2) # шаг 4.1.1  
  
 # шаг 4.1.2  
 if is\_even(A) and is\_even(B):  
 (A, B) = (int(A / 2), int(B / 2))  
 else:  
 (A, B) = (int((A + b) / 2), int((B - a) / 2))  
  
 # шаг 4.2  
 while is\_even(v):  
 v = int(v / 2) # шаг 4.2.1  
  
 # шаг 4.2.2  
 if is\_even(C) and is\_even(D):  
 (C, D) = (int(C / 2), int(D / 2))  
 else:  
 (C, D) = (int((C + b) / 2), int((D - a) / 2))  
  
 # шаг 4.3  
 if u >= v:  
 (u, A, B) = (u - v, A - C, B - D)  
 else:  
 (v, C, D) = (v - u, C - A, D - B)  
  
 # шаг 5  
 (d, x, y) = (g \* v, C, D)  
  
 # если a и b были в неправильном порядке  
 if reversed:  
 (x, y) = (y, x) # меняем найденные коэффициенты местами  
  
 return (d, x, y)

Теперь с помощью данной функции найдём наибольший общий делитель и его линейное представление для нескольких пар чисел (см. Рис. 4).

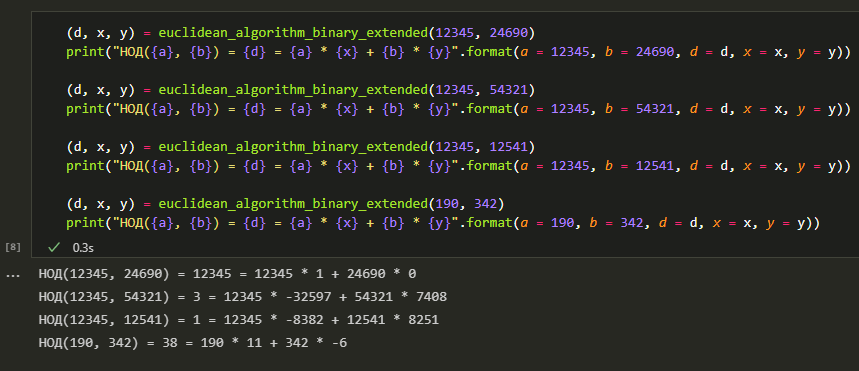


Figure 4: Примеры нахождения НОД двух чисел и его линейного представления с помощью программной реализации расширенного бинарного алгоритма Евклида

# 5 Выводы

Таким образом, была достигнута цель, поставленная в начале лабораторной работы: было проведено краткое знакомство с двумя алгоритмами нахождения наибольшего общего делителя – алгоритмом Евклида, бинарным алгоритмом Евклида, – и их расширенными версиями для нахождения линейного представления этого делителя, после чего все четыре алгоритма были успешно реализованы на языке программирования **Python**.

# Список литературы

1. Лузгарев А. Алгебра и теория чисел: конспекты лекций. <https://mahalex.net/151-153/algebra.pdf>, 2014-2016.

2. Илларионов А.А. Теория чисел: учебное пособие. <http://www.iam.khv.ru/articles/Illarionov/mainNumberTheory.pdf>, 2016.

3. Шитов Ю.А. Теоретико-численные методы в криптографии. Лекция 2. Вычисление наибольшего общего делителя. Институт космических и информационных технологий СФУ <http://ikit.edu.sfu-kras.ru/drupal/node/68>, 2014.