|  |  |  |
| --- | --- | --- |
| **Дисциплина** | **Лабораторная** | **ФИО** |
| Математические основы защиты информации и информационной безопасности | №4 | Александра Миличевич |

# Цель работы

Познакомиться с алгоритмами вычисления наибольшего общего делителя (Алгоритмами Евклида).

# Задание

1.Реализовать следующие алгоритмы программно

1.1 Алгоритм Евклида 1.2 Расширенный алгоритм Евклида 1.3 Бинарный алгоритм Евклида 1.4 Расширенный бинарный алгоритм Евклида

# Выполнение лабораторной работы

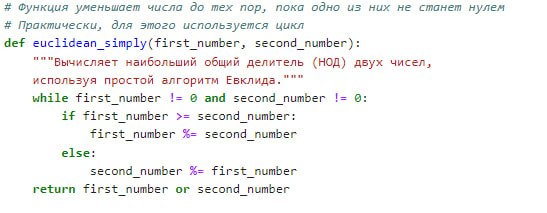
## Объяснение алгоритмов Евклида для нахождения НОД

Этот документ описывает три функции, реализующие различные алгоритмы Евклида для вычисления наибольшего общего делителя (НОД) двух чисел.

### 1. euclidean\_simply(first\_number, second\_number)

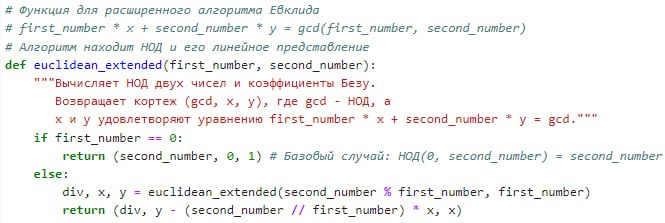
Эта функция реализует базовый итеративный алгоритм Евклида.

#### Как работает:

1. Функция использует цикл while, который выполняется до тех пор, пока оба числа (first\_number и second\_number) не станут равны нулю.
2. Внутри цикла, если first\_number больше или равно second\_number, то first\_number заменяется на остаток от деления first\_number на second\_number (first\_number %= second\_number).
3. Иначе, second\_number заменяется на остаток от деления second\_number на first\_number (second\_number %= first\_number).
4. Когда один из чисел становится равным нулю, функция возвращает второе число (которое и есть НОД). Эта функция реализует расширенный алгоритм Евклида. 

### 2. euclidean\_extended(first\_number, second\_number)

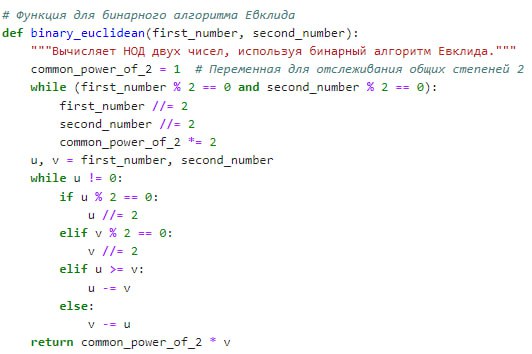
#### Как работает:

1. Функция использует рекурсию.
2. Базовый случай: если first\_number равно 0, то возвращается кортеж `(second\_number, 0, 1).
3. В противном случае, функция вызывает себя рекурсивно с параметрами second\_number % first\_number и first\_number, получает результаты div, x, y.
4. Затем вычисляет и возвращает новый кортеж (div, y - (second\_number // first\_number) \* x, x). Этот кортеж содержит НОД и коэффициенты Безу. 

### 3. binary\_euclidean(first\_number, second\_number)

#### Как работает:

1. Используется переменная common\_power\_of\_2 для отслеживания общих степеней двойки.
2. Сначала числа делятся на 2, пока оба четные, при этом common\_power\_of\_2 умножается на 2.
3. Далее, используются u и v для хранения текущих значений.
4. Пока u не равно нулю, выполняется цикл:
   * Если u четное, то u делится на 2.
   * Иначе, если v четное, то v делится на 2.
   * Иначе, если u больше или равно v, то u вычитается v.
   * Иначе, v вычитается u.
5. Функция возвращает результат, умноженный на накопленную степень двойки.



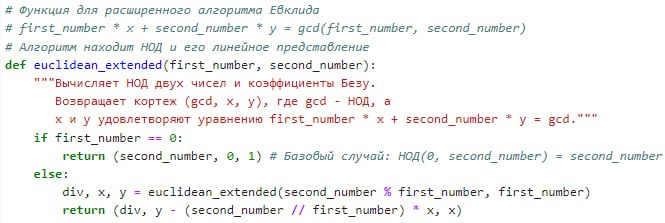
бинарний алгоритм

### 4. binary\_euclidean\_extended(first\_number, second\_number)

Эта функция реализует расширенный бинарный алгоритм Евклида.

#### Как работает:

1. Так же как и в ‘binary\_euclidean’, выделяется общая степень двойки.
2. Используются ‘u’ и ‘v’ для хранения текущих значений. ‘A, B, C, D’ — это коэффициенты для расширенного алгоритма.
3. Пока ‘u’ не равно нулю, выполняется цикл:
   * Если ‘u’ четное, то ‘u’ делится на 2. Если ‘A’ и ‘B’ четные, то они тоже делятся на 2. Иначе, ‘A’ и ‘B’ обновляются с учетом четности и делителя.
   * Аналогично для ‘v’ и ‘C, D’.
   * Если ‘u >= v’, то u и ‘A, B’ уменьшаются.
   * Иначе, ‘v’ и ‘C, D’ уменьшаются.
4. Функция возвращает НОД и коэффициенты Безу, умноженные на общую степень двойки. ### Заключение



расширенный бинарный алгоритм

Эти функции предоставляют три разных способа для вычисления наибольшего общего делителя (НОД). euclidean\_simply — это простой итеративный подход, euclidean\_extended — это рекурсивный метод, который также находит коэффициенты Безу, а binary\_euclidean использует битовые операции для повышения эффективности