

# **Информационная безопасность. Отчет по лабораторной работе № 4**

**Вычисление наибольшего общего делителя**

Мухамеджанов Исматулло Иззатуллоевич

# Содержание

1	Цель работы	5
2	Указание к работе	6
3	Выполнение лабораторной работы	7
4	Выводы	11
5	Список литературы	12

# List of Figures

3.1	Программа (1)	. . . . .	8
3.2	Программа (2)	. . . . .	9
3.3	Программа (3)	. . . . .	9
3.4	Программа (4)	. . . . .	10
3.5	Программа (5)	. . . . .	10

## List of Tables

# 1 Цель работы

Освоить на практике применение вычисление Наибольшего Общего Делителя(НОД).

## 2 Указание к работе

Алгоритм Евклида

Бинарный алгоритм Евклида

Расширенный алгоритм Евклида

Расширенный бинарный алгоритм Евклида

### 3 Выполнение лабораторной работы

#### 1. Алгоритм Евклида.

Вход. Целые числа  $n, b$ ;  $0 < b \leq a$ . Выход.  $d$  — НОД( $n, b$ ). ПОЛОЖИТЬ  $P \leftarrow 0$ ,  $q \leftarrow b$ ,  $i \leftarrow 1$ . Найти остаток от деления  $r_i \leftarrow na_i$ . Если  $ru = 0$ , то положить  $d \leftarrow r_i$ . В противном случае положить  $i \leftarrow i + 1$  и вернуться на шаг 2. Результат:  $d$ . Бинарный алгоритм Евклида является более быстрым при реализации на компьютере, поскольку использует двоичное представление чисел  $n$  и  $b$ . Бинарный алгоритм Евклида основан на следующих свойствах наибольшего общего делителя (считаем, что  $0 < b \leq n$ ): если оба числа  $n$  и  $b$  четные, то  $\text{НОД}(n, b) = 2 \text{НОД}(J, )$ ; если число  $n$  — нечетное, число  $b$  — четное, то  $\text{НОД}(n, b) = \text{НОД}(n, 2)$  если оба числа  $n$  и  $b$  нечетные,  $n > b$ , то  $\text{НОД}(n, b) = \text{НОД}(n - b, b)$ , если  $n = b$ , то  $\text{НОД}(n, b) = n$ .

#### 2. Бинарный алгоритм Евклида.

Бинарный алгоритм Евклида — метод нахождения наибольшего общего делителя двух целых чисел. Данный алгоритм «быстрее» обычного алгоритма Евклида, так как вместо медленных операций деления и умножения используются сдвиги[1]. Но это преимущество в скорости теряется с увеличением разницы между целыми числами более чем на несколько порядков, в результате чего число итераций вычитания (см. шаги 6, 7 в разделе Алгоритм) может многократно превышать число итераций обычного алгоритма, использующего сравнение по модулю. То есть скорость бинарных сдвигов дает эффект только для чисел близких друг другу.

Возможно, алгоритм был известен еще в Китае 1-го века[2], но опубликован был лишь в 1967 году израильским физиком и программистом Джоозефом Стайном. Он основан на использовании следующих свойств НОД:

$\text{НОД}(2m, 2n) = 2 \text{НОД}(m, n)$ ,  $\text{НОД}(2m, 2n+1) = \text{НОД}(m, 2n+1)$ ,  $\text{НОД}(-m, n) = \text{НОД}(m, n)$

3. Расширенный алгоритм Евклида. Целые числа  $a, b$ :  $0 < b \leq a$  Выход  $\text{НОД}(a, b)$ ; такие числа  $x, y$ , что  $ax + by = d$

4. Расширенный бинарный алгоритм Евклида. Целые числа  $a, b$ :  $0 < b \leq a$  Выход  $\text{НОД}(a, b)$   $g = 1$  а и  $b$  нечетные,  $a = a / 2$ ,  $b = b / 2$ ,  $g = 2 * g$  пока  $a$  или  $b$  один из них не станет нечётным  $u = a$ ,  $v = b$ ,  $A = 1$ ,  $B = 0$ ,  $C = 0$ ,  $D = 1$   $u \% 2 \neq 0$ :  $u = u / 2$ ,  $A = A / 2$ ,  $B = B / 2$ ,  $A = (A + b) / 2$ ,  $B = (B - a) / 2$   $v \% 2 \neq 0$ :  $v = v / 2$ ,  $C = C / 2$ ,  $D = D / 2$ ,  $C = (C + b) / 2$ ,  $D = (D - a) / 2$   $u \geq v$ ,  $u = u - v$ ,  $A = A - C$ ,  $B = B - D$ , else  $v = v - u$ ,  $C = C - A$ ,  $D = D - B$

$d = g * v$ ,  $x = C$ ,  $y = D$  Вывод  $d, x, y$

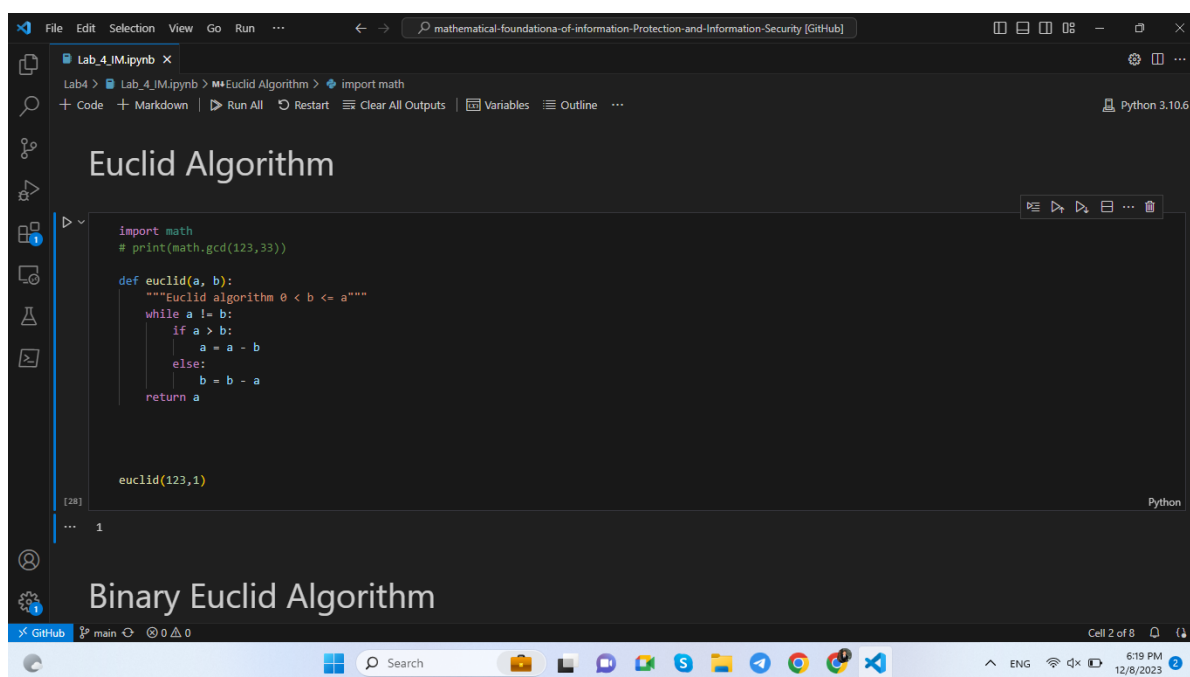


Figure 3.1: Программа (1)



```

def binary_Euclid(a,b):
    """Binary code for Euclid algorithm"""
    g = 1
    while a % 2 == 0 and b % 2 == 0:
        a = a // 2
        b = b // 2
        g = g * 2
    u = a
    v = b
    while u != 0:
        if u % 2 == 0:
            u = u // 2
        elif v % 2 == 0:
            v = v // 2
        if u >= v:
            u = u - v
        else:
            v = v - u
    d = g * v
    return d

binary_Euclid(8,4)

```

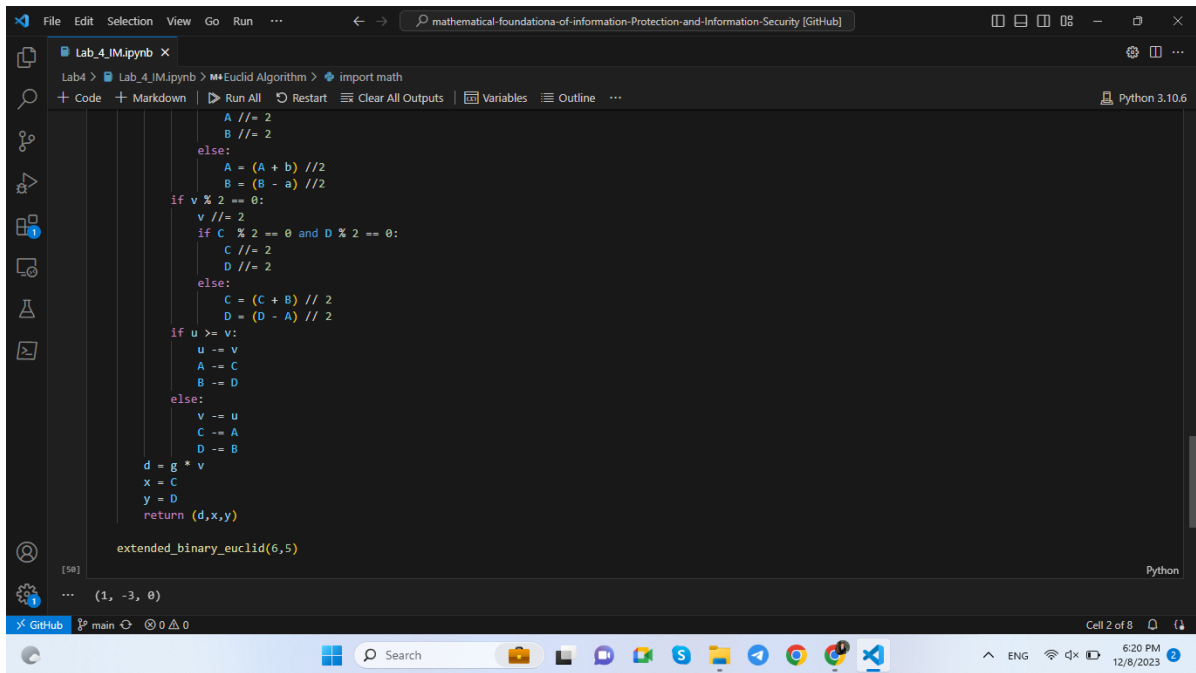
Figure 3.2: Программа (2)

```

def extended_binary_euclid(a, b):
    g = 1
    while a % 2 == 0 and b % 2 == 0:
        a //= 2
        b //= 2
        g = 2 * g
    u = a
    v = b
    A = 1
    B = 0
    C = 0
    D = 1
    while u != 0:
        if u % 2 == 0:
            u = u // 2
            if A % 2 == 0 and B % 2 == 0:
                A //= 2
                B //= 2
            else:
                A = (A + b) // 2
                B = (B - a) // 2
        if v % 2 == 0:
            v = v // 2
            if C % 2 == 0 and D % 2 == 0:
                C //= 2
                D //= 2
        if u >= v:
            u = u - v
            A = A - D
            B = B - C
        else:
            v = v - u
            C = C - B
            D = D - A
    return g, A, B, C, D

```

Figure 3.3: Программа (3)



```

import math

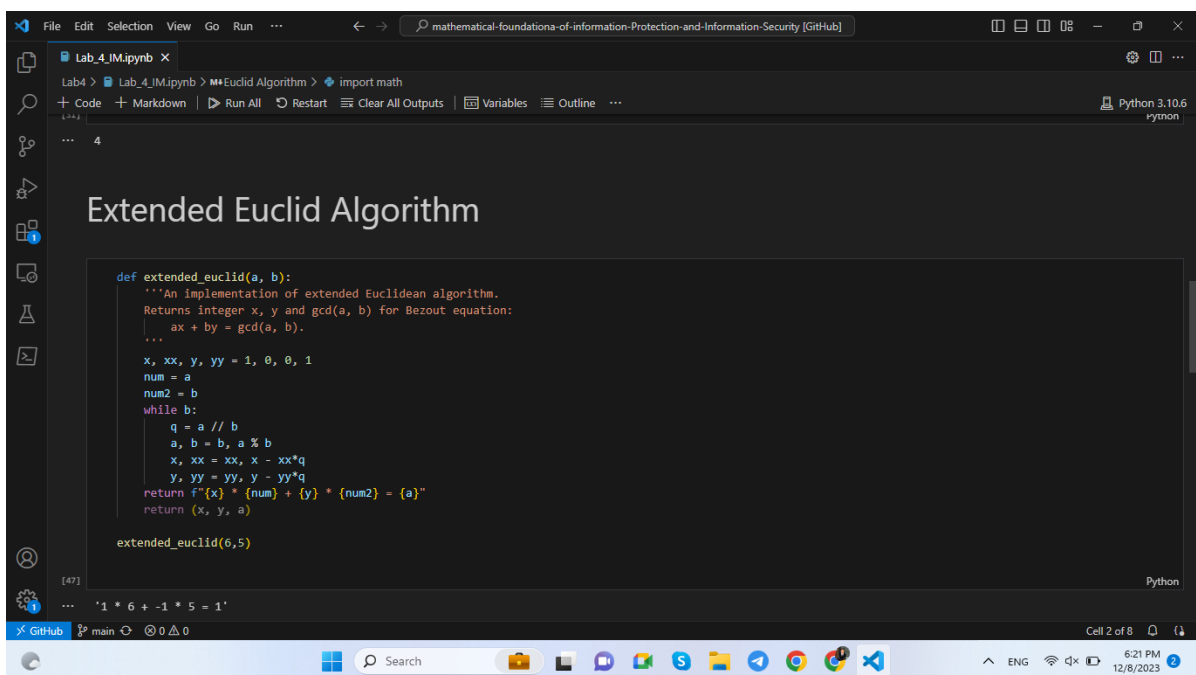
def extended_binary_euclid(a, b):
    if a == 0:
        return b, 0, 1
    else:
        A = (a + b) // 2
        B = (a - b) // 2
        if v % 2 == 0:
            v //= 2
            if C % 2 == 0 and D % 2 == 0:
                C //= 2
                D //= 2
            else:
                C = (C + B) // 2
                D = (D - A) // 2
        if u >= v:
            u -= v
            A -= B
            B -= D
        else:
            v -= u
            C -= A
            D -= B
    d = g * v
    x = C
    y = D
    return (d, x, y)

extended_binary_euclid(6, 5)

```

Output: (1, -3, 0)

Figure 3.4: Программа (4)



### Extended Euclid Algorithm

```

def extended_euclid(a, b):
    """An implementation of extended Euclidean algorithm.
    Returns integer x, y and gcd(a, b) for Bezout equation:
    ax + by = gcd(a, b).
    """
    x, xx, y, yy = 1, 0, 0, 1
    num = a
    num2 = b
    while b:
        q = a // b
        a, b = b, a % b
        x, xx = xx, x - qx*q
        y, yy = yy, y - yy*q
    return f"{x} * {num} + {y} * {num2} = {a}"
    return (x, y, a)

extended_euclid(6, 5)

```

Output: 1 \* 6 + -1 \* 5 = 1

Figure 3.5: Программа (5)

## **4 Выводы**

Освоены методы определения НОД

## 5 Список литературы

1. Методические материалы курса
2. Википедия