**НАО «Восточно-Казахстанский технический университет имени Д. Серикбаева»**

**Дисциплина «Теоретико-числовые методы в криптографии»**

# ПРАКТИЧЕСКАЯ РАБОТА 2

# НОД, НОК И РАСШИРЕННЫЙ АЛГОРИТМ ЕВКЛИДА

*Методические указания и задание к выполнению*

**Аудиторная нагрузка:** 3 академических часа. Вначале изучите методические сведения и разобранные примеры, затем выполните задания своего варианта. Ручные вычисления сопровождаются программной проверкой.

## 1 Цель работы

Освоить вычисление НОД и НОК целых чисел, построение расширенной таблицы Евклида и проверку коэффициентов Безу. Применить результат к линейному уравнению и поиску обратного элемента по модулю.

## 2 Планируемые результаты

- записывать полную цепочку делений с остатком и правильно считать шаги;
- получать коэффициенты Безу прямым табличным методом;
- объяснять условия существования целочисленного решения и обратного элемента;
- реализовывать вычисления целыми числами в Python;
- проверять результаты с помощью независимых равенств и граничных примеров.

## 3 Программное обеспечение

Достаточно Python 3 и любой среды для его запуска. Дополнительные пакеты не нужны. Все листинги раздела 5 разместите последовательно в одном файле `practical02.py`. Сначала запустите готовый пример, затем подставьте данные своего варианта. Обозначения `gcd`, `lcm` в коде соответствуют НОД и НОК.

## 4 Методические сведения и разобранные примеры

### 4.1 Обычный алгоритм Евклида

Основное равенство: $\gcd(a, b) = \gcd(b, a \bmod b)$ при $b > 0$. Перед началом вычислений берут модули аргументов. Для подсчёта шагов большее число ставят первым. На каждом шаге деление должно иметь вид $a = qb + r$, $0 \le r < b$.

**Образец.** Найдём НОД чисел 240 и 46.

| Шаг | Делимое | Делитель | Частное | Остаток |
|---|---|---|---|---|
| 1 | 240 | 46 | 5 | 10 |
| 2 | 46 | 10 | 4 | 6 |
| 3 | 10 | 6 | 1 | 4 |
| 4 | 6 | 4 | 1 | 2 |
| 5 | 4 | 2 | 2 | 0 |

Ответ: $g = 2$, число делений $n = 5$. Последнее деление включается в счёт. Число 2 — последний ненулевой остаток; ноль означает остановку.

Для нулевого второго аргумента деления не выполняются: $\gcd(a, 0) = |a|$. В программе принято $\gcd(0,0) = 0$; это соглашение, а не наибольший положительный делитель двух нулей.

### 4.2 Таблица расширенного алгоритма

К каждому остатку добавляем коэффициенты $s, t$, чтобы сохранялось $r = as + bt$. Для положительных исходных чисел начальные строки имеют вид $(r, s, t) = (a, 1, 0)$ и $(b, 0, 1)$. Новую строку получаем из двух предыдущих:

$$r_{\text{new}} = r_0 - qr_1, \quad s_{\text{new}} = s_0 - qs_1, \quad t_{\text{new}} = t_0 - qt_1.$$

Здесь индекс 0 обозначает более раннюю из двух текущих строк, индекс 1 — последнюю строку. После вычисления новой строки окно сдвигается на одну строку вперёд.

| Строка | Частное $q$ | Остаток $r$ | $s$ | $t$ |
|---|---|---|---|---|
| Начальная 1 | — | 240 | 1 | 0 |
| Начальная 2 | — | 46 | 0 | 1 |
| 1 | 5 | 10 | 1 | −5 |
| 2 | 4 | 6 | −4 | 21 |
| 3 | 1 | 4 | 5 | −26 |
| 4 | 1 | 2 | −9 | 47 |
| 5 | 2 | 0 | 23 | −120 |

Например, строка 2 получается так: $6 = 46 - 4 \cdot 10$, $s = 0 - 4 \cdot 1 = -4$, $t = 1 - 4(-5) = 21$. Проверка этой строки: $240(-4) + 46 \cdot 21 = 6$.

Коэффициенты Безу берём из последней ненулевой строки:

$$240(-9) + 46 \cdot 47 = 2.$$

При отрицательных аргументах можно сначала выполнить таблицу для модулей, затем изменить знак коэффициента при том аргументе, чей знак был изменён. Так, $(-240) \cdot 9 + 46 \cdot 47 = 2$.

### 4.3 Вычисление НОК

Для ненулевых аргументов $\operatorname{lcm}(a, b) = |ab|/g$. В образце

$$L = (240/2) \cdot 46 = 5520.$$

Проверяем три соотношения: $L \bmod 240 = 0$, $L \bmod 46 = 0$, $gL = |240 \cdot 46|$. При наличии нулевого аргумента принимаем НОК равным нулю и не выполняем деление на НОД.

### 4.4 Линейное уравнение: когда есть решение

Для положительных $a, b$ уравнение $ax + by = c$ разрешимо в целых числах тогда и только тогда, когда $g \mid c$. При выполнении этого условия умножаем коэффициенты Безу на $c/g$:

$$x_0 = s(c/g), \quad y_0 = t(c/g), \qquad x = x_0 + (b/g)k, \quad y = y_0 - (a/g)k.$$

Здесь $k$ — любое целое число. Для $240x + 46y = 14$ получаем $(x_0, y_0) = (-63, 329)$. Все решения:

$$x = -63 + 23k, \qquad y = 329 - 120k, \qquad k \in \mathbb{Z}.$$

При $k = 3$ получается $(6, -31)$: $240 \cdot 6 + 46(-31) = 14$. Для правой части 15 решений нет: $15 \bmod 2 = 1$. В этом случае нужно обосновать отсутствие решения, а не подбирать приблизительные значения.

### 4.5 Обратный элемент по модулю

Обратным к $a$ по модулю $m > 1$ называется число $u$, для которого $au \bmod m = 1$. Обратный существует при $\gcd(a, m) = 1$; тогда из $as + mt = 1$ получаем $u = s \bmod m$. Выбираем представителя $0 \le u < m$.

Например, $17(-367) + 3120 \cdot 2 = 1$, поэтому $u = 2753$. Проверка: $17 \cdot 2753 \bmod 3120 = 1$. Для 240 по модулю 46 обратного нет, поскольку НОД равен 2. Это применение тождества Безу; теория сравнений будет изучаться в следующих темах.

## 5 Программные примеры

### 5.1 НОД с записью делений

`divmod(a, b)` возвращает частное и остаток. В список `rows` записываются четвёрки «делимое, делитель, частное, остаток». Его длина равна числу делений; при двух нулях возвращаются НОД 0 и пустой список.

```
def gcd_trace(a, b):
    a, b = sorted((abs(a), abs(b)), reverse=True)
    rows = []
    while b != 0:
        q, r = divmod(a, b)
        rows.append((a, b, q, r))
        a, b = b, r
    return a, rows
```

### 5.2 Коэффициенты Безу и НОК

Переменные `r0, r1` хранят два последних остатка, `s0, s1` и `t0, t1` — их коэффициенты. Python сначала вычисляет всю правую часть множественного присваивания и только затем заменяет значения. Поэтому строки обновления не теряют старые коэффициенты. После цикла знак возвращается коэффициентам исходных аргументов.

```
def extended_gcd(a, b):
    if a == 0 and b == 0:
        return 0, 0, 0
    r0, r1 = abs(a), abs(b)
    s0, s1 = 1, 0
    t0, t1 = 0, 1
    while r1 != 0:
        q = r0 // r1
        r0, r1 = r1, r0 - q * r1
        s0, s1 = s1, s0 - q * s1
        t0, t1 = t1, t0 - q * t1
    s = s0 if a >= 0 else -s0
    t = t0 if b >= 0 else -t0
    return r0, s, t
```

Для НОК используем найденный НОД. Нулевые аргументы проверяем до деления.

```
def lcm_euclid(a, b):
    if a == 0 or b == 0:
        return 0
    g, _, _ = extended_gcd(a, b)
    return abs((a // g) * b)
```

Оператор `//` выполняет целочисленное деление. В формуле НОК деление на $g$ точное. Оператор `/` здесь не нужен: он создал бы число с плавающей точкой.

### 5.3 Два приложения

`inverse_mod` возвращает обратный элемент или `None`, если его нет. Для некорректного модуля $m \le 1$ возникает `ValueError`. Функция `solve_linear` предназначена для **положительных** $a, b$ и

любого целого $c$; результат — четвёрка $(x_0, y_0, \Delta x, \Delta y)$, задающая все решения $x = x_0 + \Delta x\, k, y = y_0 + \Delta y\, k$. Значение `None` означает отсутствие решения.

```
def inverse_mod(a, m):
    if m <= 1:
        raise ValueError("m must be greater than 1")
    g, s, _ = extended_gcd(a, m)
    return s % m if g == 1 else None


def solve_linear(a, b, c):
    if a <= 0 or b <= 0:
        raise ValueError("a and b must be positive")
    g, s, t = extended_gcd(a, b)
    if c % g != 0:
        return None
    scale = c // g
    x0, y0 = s * scale, t * scale
    return x0, y0, b // g, -(a // g)
```

### 5.4 Запуск готового примера

Добавьте следующие строки после определений функций.

```
a, b, c = 240, 46, 14
g, rows = gcd_trace(a, b)
for step, (dividend, divisor, q, r) in enumerate(rows, 1):
    print(f"{step}: {dividend} = {q} * {divisor} + {r}")

g, s, t = extended_gcd(a, b)
print("gcd, steps:", g, len(rows))
print("Bezout s, t:", s, t)
print("Check:", a * s + b * t)
print("lcm:", lcm_euclid(a, b))
print("inverse of a mod b:", inverse_mod(a, b))
print("linear (x0, y0, dx, dy):", solve_linear(a, b, c))
print("inverse of 17 mod 3120:", inverse_mod(17, 3120))
```

**Ожидаемый вывод**

```
1: 240 = 5 * 46 + 10
2: 46 = 4 * 10 + 6
3: 10 = 1 * 6 + 4
4: 6 = 1 * 4 + 2
5: 4 = 2 * 2 + 0
gcd, steps: 2 5
Bezout s, t: -9 47
Check: 2
lcm: 5520
inverse of a mod b: None
linear (x0, y0, dx, dy): (-63, 329, 23, -120)
inverse of 17 mod 3120: 2753
```

Сначала сравните каждое деление с таблицей раздела 4.1. Затем проверьте равенство Безу, НОК и решение уравнения. Английские подписи вывода соответствуют именам функций; число после `Check` должно совпадать с НОД.

### 5.5 Проверка нулей, знаков и приложений

В следующем фрагменте стандартная функция `math.gcd` используется как независимая проверка. Оператор `assert` прерывает программу, если указанное условие ложно; проверки выполняются при обычном запуске Python без режима оптимизации.

```
from math import gcd

cases = [(240, 46), (-240, 46), (240, -46),
         (-240, -46), (0, 46), (240, 0), (0, 0), (46, 240)]
for a, b in cases:
    g, s, t = extended_gcd(a, b)
    assert g == gcd(a, b)
    assert a * s + b * t == g
    assert gcd_trace(a, b)[0] == g
    assert g * lcm_euclid(a, b) == abs(a * b)

assert inverse_mod(17, 3120) == 2753
assert inverse_mod(240, 46) is None
assert solve_linear(240, 46, 15) is None
x0, y0, dx, dy = solve_linear(240, 46, 14)
for k in (-2, 0, 3):
    assert 240 * (x0 + dx * k) + 46 * (y0 + dy * k) == 14
print("All checks passed")
```

Ожидаемый вывод этого фрагмента: `All checks passed`. Затем добавьте в список `cases` пару из своего варианта и проверьте собственные результаты. Успешный запуск примеров подтверждает проверенные случаи, а математическое обоснование алгоритма даётся его инвариантом и завершением.

## 6 Порядок выполнения работы

1. **Первый академический час:** разобрать методичку; вручную вычислить НОД и НОК своего варианта; записать все деления.
2. **Второй академический час:** заполнить расширенную таблицу; проверить тождество Безу; разобрать уравнение и существование обратного элемента.
3. **Третий академический час:** запустить код, сопоставить результаты, выполнить проверки знаков и нулей, оформить выводы и защитить работу.

Теоретическое обоснование и ручная таблица включаются в отчёт вместе с программными результатами.

## 7 Индивидуальные задания

Исходные данные $a, b, c$ выберите из раздела 8. Во всех вариантах $a > b > 1$.

**Задание 1. Обычный алгоритм и число делений.**

Запишите полную цепочку делений для $\gcd(a, b)$. Для каждого шага проверьте $a_i = q_i b_i + r_i$ и $0 \le r_i < b_i$. Укажите НОД $g$, число делений $n$ и число десятичных цифр $d$ в исходном $b$. Сопоставьте $n$ с границей $5d$.

**Задание 2. Расширенная таблица и НОК.**

Постройте таблицу со столбцами $q, r, s, t$, включая две начальные строки и последнюю нулевую строку. Выпишите коэффициенты Безу из строки НОД и проверьте $as + bt = g$. Вычислите НОК $L$, проверьте его делимость на $a, b$ и равенство $gL = ab$. Объясните, почему $(a/g)$ и $(b/g)$ взаимно просты.

**Задание 3. Линейное уравнение.**

Исследуйте уравнение $ax + by = c$. Если оно разрешимо, найдите одно решение, запишите все решения и проверьте две разные пары. Дополнительно выберите решение с $0 \le x < b/g$. Если решений нет, приведите остаток $c \bmod g$, доказывающий это.

**Задание 4. Обратный элемент.**

Выясните, существует ли обратный к $a$ по модулю $b$. Если существует, найдите $0 \le u < b$ и проверьте $au \bmod b = 1$. Если не существует, объясните причину через НОД. Коэффициент Безу допускается использовать для обратного элемента только при $g = 1$.

**Задание 5. Программа и граничные случаи.**

Запустите код с данными варианта и сравните с ручными расчётами. Проверьте также пары $(-a, b)$, $(a, -b)$, $(0, b)$, $(a, 0)$, $(0,0)$ и $(b, a)$. Для каждой пары укажите НОД, НОК и корректное равенство Безу. Объясните изменение знаков коэффициентов и отсутствие деления на ноль.

**Задание 6. Общее задание на оценку.**

Подсчитайте деления для пар (89,55) и (144,89). Сопоставьте частные, остатки и число шагов. Объясните связь с числами Фибоначчи. Время выполнения программы измерять не требуется: сравнивается именно число делений.

## 8 Варианты индивидуальных заданий

| **Вариант** | $a$ | $b$ | $c$ |
|---|---|---|---|
| 1 | 240 | 46 | 14 |
| 2 | 391 | 299 | 5 |
| 3 | 221 | 85 | 51 |
| 4 | 527 | 341 | 62 |
| 5 | 655 | 264 | 7 |
| 6 | 899 | 493 | 56 |
| 7 | 1001 | 143 | 286 |
| 8 | 748 | 306 | 34 |
| 9 | 1235 | 494 | 12 |
| 10 | 987 | 610 | 3 |
| 11 | 119 | 84 | 35 |
| 12 | 437 | 247 | 38 |
| 13 | 771 | 513 | 12 |
| 14 | 609 | 357 | 5 |
| 15 | 1365 | 1001 | 182 |
| 16 | 2027 | 975 | 11 |
| 17 | 1597 | 987 | 8 |
| 18 | 765 | 432 | 10 |
| 19 | 1729 | 703 | 57 |
| 20 | 1009 | 433 | 17 |

## 9 Требования к отчёту

Укажите дисциплину, номер работы, Ф. И. О., группу, вариант и исходные данные. Далее приведите: цель; ручную цепочку делений; расширенную таблицу; НОД, НОК и коэффициенты Безу с проверками; решение уравнения и проверку обратимости; текст программы и её вывод; результаты граничных случаев; сравнение пар Фибоначчи; краткий вывод о выполненных задачах.

Код оформляйте моноширинным шрифтом с сохранением отступов. Основной текст выравнивайте по ширине. Таблицы должны содержать подписи столбцов, формулы — обозначения переменных. Скриншот запуска дополняет, но не заменяет текст программы и расчёты.

## 10 Критерии оценивания

| Составляющая работы | Баллы |
|---|---|
| Полная цепочка Евклида, НОД, число шагов и оценка | 20 |
| Расширенная таблица, тождество Безу и НОК | 25 |
| Линейное уравнение и обратный элемент | 20 |
| Рабочий код, сравнение результатов и граничные случаи | 20 |
| Пары Фибоначчи, выводы и ответы при защите | 15 |
| Итого | 100 |

## 11 Контрольные вопросы

1. Почему остаток обязан быть меньше делителя?
2. Какие две причины обосновывают завершение и корректность алгоритма Евклида?
3. Как проверяется произвольная строка расширенной таблицы?
4. Почему коэффициенты при нулевом остатке не являются ответом для тождества Безу?
5. Можно ли считать ошибкой коэффициенты Безу, отличающиеся от ответа преподавателя?
6. Почему в вычислении НОК сначала нужно проверить нулевые аргументы?
7. Что возвращают функции, если уравнение неразрешимо или обратного элемента нет?
8. Чем верхняя оценка Ламе отличается от фактического числа делений?
9. Почему выражение `a / g` не подходит для точных целочисленных вычислений?
10. Как меняются коэффициенты Безу при изменении знака одного исходного числа?

## Литература

1. Мартынов Л. М. **Алгебра и теория чисел для криптографии**. СПб.: Лань, 2020. §§ 3.1–3.2, с. 85–91.
2. Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory**. 2nd ed. Pearson Education, 2006. §§ 3.1.3–3.2.
3. Лекция 2 **«НОД, НОК и алгоритм Евклида»** настоящего курса, разделы 2–7.