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

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

# ПРАКТИЧЕСКАЯ РАБОТА 3

# ПРОСТЫЕ ЧИСЛА, ФАКТОРИЗАЦИЯ И РЕШЕТО ЭРАТОСФЕНА

*Методические указания и задание к выполнению*

**Аудиторная нагрузка:** 2 академических часа. Сначала изучите методические сведения и разобранные примеры, затем выполните свой вариант. Примеры кода можно подготовить до занятия.

## 1 Цель работы

Научиться доказывать простоту и составность небольших целых чисел, выполнять полное разложение на простые множители и строить решето Эратосфена. Проверить вычисления программно и объяснить, почему выбранные границы циклов достаточны.

## 2 Планируемые результаты

После выполнения работы студент умеет находить достаточную границу пробных делений, записывать каноническое разложение, проверять его полноту, получать все простые до заданной границы и отличать число операций решета от количества составных чисел. Дополнительное применение разложения — достраивание числа до полного квадрата.

## 3 Программное обеспечение

Python 3.8 или новее и любая среда его запуска. Используется только стандартный модуль `math`; дополнительные пакеты не нужны. Все блоки Python из раздела 5 разместите **последовательно в одном файле** `practical03.py`. В среде с ячейками выполняйте их сверху вниз. В командной строке файл запускается командой `python practical03.py` или `python3 practical03.py` в зависимости от настройки системы.

Входные данные функций — целые числа. Факторизация определена здесь для положительного аргумента, решето — для неотрицательной границы. Ручное обоснование нужно выполнить до сопоставления с программным ответом.

## 4 Методические сведения и разобранные примеры

### 4.1 Проверка простоты: пример 97 и 121

Простое число имеет ровно два положительных делителя. Числа 0 и 1 не являются простыми или составными. Если целое $n > 1$ составное, оно имеет простой делитель $p \le \sqrt{n}$. Поэтому достаточно проверить простые числа до **включённой** границы $\lfloor \sqrt{n} \rfloor$.

Для 97 граница равна 9. Проверим простые 2, 3, 5 и 7:

| **Делитель $p$** | **2** | **3** | **5** | **7** |
|---|---|---|---|---|
| Остаток $97 \bmod p$ | 1 | 1 | 2 | 6 |

Все остатки ненулевые, значит, 97 простое. Проверять 11 уже не требуется, поскольку $11^2 > 97$.

Для 121 граница равна 11. Проверки 2, 3, 5 и 7 не дают делителя, но $121 \bmod 11 = 0$. Следовательно, $121 = 11^2$ составное. Замена условия $d^2 \le n$ на $d^2 < n$ пропустила бы этот делитель.

В программе ниже перебираются 2 и **все нечётные кандидаты**, включая 9. Поэтому для 97 счётчик равен 5, а для 121 — 6. В ручном доказательстве достаточно соответственно четырёх и пяти простых кандидатов. Это разные способы перебора с одним математическим результатом.

### 4.2 Полная факторизация: пример 7560

Пусть $R$ — текущая неразложенная часть. Начните с $R = 7560$ и извлекайте найденный простой множитель столько раз, сколько возможно. После уменьшения $R$ пересматривайте границу: достаточно кандидатов с $p^2 \le R$.

| Действие | Деление или проверка | Результат |
|---|---|---|
| Извлечь степень 2 | $7560/2 = 3780, 3780/2 = 1890, 1890/2 = 945$ | $2^3, R = 945$ |
| Извлечь степень 3 | $945/3 = 315, 315/3 = 105, 105/3 = 35$ | $3^3, R = 35$ |
| Проверить 5 | $5^2 \le 35, 35/5 = 7$ | $5, R = 7$ |
| Проверить границу | Следующий кандидат 7; $7^2 > 7$ | Остаток 7 прост |

Каноническое разложение:

$$7560 = 2^3 \cdot 3^3 \cdot 5 \cdot 7.$$

Проверка полноты состоит из трёх частей: основания 2, 3, 5 и 7 простые; показатели положительные целые; произведение $8 \cdot 27 \cdot 5 \cdot 7$ равно 7560. При каждой замене $R$ выполняется точное целочисленное деление. Запись `/` в Python даёт вещественный результат; в алгоритме применяется `//`.

Оставшуюся часть можно признать простой только после достаточного перебора. Например, извлечение 2 из 242 даёт 121. Это ещё составное число, а полное разложение равно $242 = 2 \cdot 11^2$.

### 4.3 Достраивание до квадрата

В квадрате показатели всех простых множителей чётные. Чтобы найти наименьшее положительное $h$, для которого $Mh$ — квадрат, перемножьте простые основания с нечётными показателями:

$$h = \prod_{\alpha_i \text{ нечётно}} p_i, \qquad M = \prod_i p_i^{\alpha_i}.$$

Для $M = 7560$ все четыре показателя нечётные, поэтому $h = 2 \cdot 3 \cdot 5 \cdot 7 = 210$. Тогда

$$7560 \cdot 210 = 2^4 \cdot 3^4 \cdot 5^2 \cdot 7^2 = 1260^2.$$

Минимальность следует из единственности разложения: любой подходящий положительный множитель обязан содержать по крайней мере по одному экземпляру каждого простого с нечётным показателем и потому делится на 210. Если исходное число уже квадрат, ответ $h = 1$.

### 4.4 Решето: все простые до 60

Создайте таблицу чисел от 0 до $N = 60$. Исключите 0 и 1. Для каждого ещё не вычеркнутого $p$, удовлетворяющего $p^2 \le 60$, вычёркивайте кратные, начиная с $p^2$. Для этой границы базовые простые — 2, 3, 5 и 7.

| Базовое простое $p$ | Первое отмечаемое число $p^2$ | Число отметок $\lfloor 60/p \rfloor - p + 1$ |
|---|---|---|
| 2 | 4 | 29 |
| 3 | 9 | 18 |
| 5 | 25 | 8 |

| Базовое простое $p$ | Первое отмечаемое число $p^2$ | Число отметок $\lfloor 60/p \rfloor - p + 1$ |
|---|---|---|
| 7 | 49 | 2 |

Оставшиеся простые: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59. Их число обозначают $\pi(60) = 17$.

В работе счётчик $C(N)$ учитывает **каждое** выполнение присваивания «вычеркнуто», в том числе для уже исключённого числа. Поэтому $C(60) = 29 + 18 + 8 + 2 = 57$. Начальные присваивания для 0 и 1 не считаются. Различных составных чисел в $[2,60]$ меньше: $60 - 1 - 17 = 42$.

Для произвольной границы проверьте программный счётчик по формуле

$$C(N) = \sum_{\substack{p \text{ простое} \\ p^2 \le N}} (\lfloor N/p \rfloor - p + 1).$$

Начало с $p^2$ достаточно: кратные $2p, 3p, \ldots, (p-1)p$ уже имеют меньший простой делитель и были исключены раньше. Любое составное не больше $N$ имеет простой делитель не больше $\sqrt{N}$, поэтому после обработки этих базовых простых таблица готова.

## 5 Программные примеры с пояснениями

### 5.1 Проверка одного числа

Функция возвращает тройку `(prime, divisor, checks)`: признак простоты, первый найденный делитель или `None`, число выполненных операций `%`. При $n < 2$ возвращается `False`, но это не означает составность: такие числа находятся вне обоих классов. Для 2 деления не выполняются.

```
def trial_test(n):
    if n < 2:
        return False, None, 0
    if n == 2:
        return True, None, 0
    checks = 1
    if n % 2 == 0:
        return False, 2, checks
    d = 3
    while d * d <= n:
        checks += 1
        if n % d == 0:
            return False, d, checks
        d += 2
    return True, None, checks
```

Условие `d * d <= n` сравнивает целые числа без округления. В счётчик входит первоначальная проверка делимости на 2, если она действительно выполняется. Для найденного делителя полезно дополнительно проверить $1 < d < n$ и $n \bmod d = 0$.

### 5.2 Разложение с показателями

Результат — список пар `(простое, показатель)` в возрастающем порядке оснований. Числу 1 соответствует пустой список. Внутренний цикл извлекает всю степень текущего кандидата; внешний цикл использует уменьшающийся `rest`.

```
def factor_trial(n):
    if n < 1:
        raise ValueError("Ожидается положительное целое")
    factors = []
    rest, p = n, 2
    while p * p <= rest:
        exponent = 0
        while rest % p == 0:
            rest //= p
            exponent += 1
        if exponent > 0:
            factors.append((p, exponent))
        p = 3 if p == 2 else p + 2
    if rest > 1:
        factors.append((rest, 1))
    return factors
```

Хотя кандидаты включают 9, 15 и другие составные числа, положительный показатель может появиться только у простого: все меньшие простые делители текущего остатка уже извлечены. Условие выхода позволяет добавить оставшийся `rest > 1` как простой множитель.

### 5.3 Решето и счётчик отметок

Массив `flags` содержит единицу у ещё не исключённого числа и ноль у исключённого. Верхняя граница `range` не включается, поэтому используется `N + 1`. Счётчик увеличивается без проверки прежнего значения ячейки: именно это соответствует формуле $C(N)$.

```
def sieve(N):
    if N < 0:
        raise ValueError("Граница должна быть неотрицательной")
    flags = bytearray([1]) * (N + 1)
    flags[0] = 0
    if N >= 1:
        flags[1] = 0
    marks, p = 0, 2
    while p * p <= N:
        if flags[p]:
            for j in range(p * p, N + 1, p):
                flags[j] = 0
                marks += 1
        p += 1
    primes = [i for i in range(2, N + 1) if flags[i]]
    return primes, marks
```

Решето хранит сведения обо всём диапазоне, поэтому требует память порядка $N$. Пробные деления проверяют одно число. Сравнивать их счётчики как время выполнения нельзя: задачи и учитываемые операции различаются.

### 5.4 Минимальный множитель до квадрата

Функция использует уже определённую `factor_trial`. Точная целочисленная функция `isqrt` возвращает $\lfloor\sqrt{x}\rfloor$; после построения $h$ значение $Mh$ обязано быть полным квадратом.

```
from math import isqrt

def square_completion(n):
    h = 1
    for p, exponent in factor_trial(n):
        if exponent % 2 == 1:
            h *= p
    root = isqrt(n * h)
    return h, root
```

### 5.5 Запуск разобранного примера

После определения всех функций добавьте следующий блок. Для своего варианта затем замените только значения `N`, `a`, `b`, `M` в первой строке этого блока. Числа в блоке контрольных проверок раздела 5.6 сохраняйте: это общие проверочные примеры.

```
N, a, b, M = 60, 97, 121, 7560
for n in (a, b):
    prime, divisor, checks = trial_test(n)
    print(n, "->", prime, divisor, checks)
factors = factor_trial(M)
print("Разложение:", factors)
primes, marks = sieve(N)
print("Простые:", primes)
print("Количество:", len(primes), "Отметки:", marks)
print("Различные составные:", N - 1 - len(primes))
h, root = square_completion(M)
print("Множитель:", h, "Корень:", root)
```

**Ожидаемый вывод**

```
97 -> True None 5
121 -> False 11 6
Разложение: [(2, 3), (3, 3), (5, 1), (7, 1)]
Простые: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59]
Количество: 17 Отметки: 57
Различные составные: 42
Множитель: 210 Корень: 1260
```

В строках проверки чисел после стрелки идут признак простоты, делитель и счётчик. Например, `121 -> False 11 6` означает составность 121, найденный делитель 11 и шесть операций `%`.

### 5.6 Проверки перед использованием

Добавьте проверки ниже после демонстрационного блока. Они проверяют случаи 0 и 1, единственное чётное простое, квадраты простых, полное извлечение степеней и малые границы решета. Если условие `assert` ложно, Python остановится с `AssertionError`, и ошибку нужно исправить.

```
assert trial_test(0) == (False, None, 0)
assert trial_test(1) == (False, None, 0)
assert trial_test(2) == (True, None, 0)
assert trial_test(3) == (True, None, 1)
assert trial_test(4) == (False, 2, 1)
assert trial_test(49) == (False, 7, 4)
assert trial_test(121) == (False, 11, 6)
assert factor_trial(1) == []
assert factor_trial(242) == [(2, 1), (11, 2)]
assert factor_trial(7560) == [(2, 3), (3, 3), (5, 1), (7, 1)]
assert sieve(0) == ([], 0)
assert sieve(1) == ([], 0)
assert sieve(2) == ([2], 0)
assert sieve(10) == ([2, 3, 5, 7], 5)
assert square_completion(1) == (1, 1)
assert square_completion(144) == (1, 12)
assert square_completion(7560) == (210, 1260)
print("Все проверки пройдены")
```

При успешном выполнении появится строка `Все проверки пройдены`. Для `factor_trial(0)` и `sieve(-1)` ожидается `ValueError`: это выход за оговорённую область входных данных, а не корректный числовой результат. Проверяйте эти вызовы отдельно, чтобы намеренная ошибка не прервала основной пример.

## 6 Порядок выполнения работы

1. Разобрать примеры раздела 4 и объяснить границы циклов без обращения к коду.
2. Получить номер варианта, выписать $N, a, b, M$ из раздела 8.
3. Выполнить задания 1–4 вручную и записать обоснования; таблицу решета допускается вести на отдельном листе.
4. Собрать и запустить программу, выполнить задание 5, сопоставить результаты.
5. Оформить отчёт и подготовиться к защите. Ориентир для занятия: 15 минут на разбор, 45 минут на свой вариант, 20 минут на проверку и 10 минут на обсуждение.

## 7 Индивидуальные задания

**Задание 1. Проверка простоты двух чисел.**

Для каждого из $a$ и $b$ найдите $\lfloor\sqrt{n}\rfloor$ и определите, простое оно или составное. Для простого предъявите полный список простых кандидатов до этой границы и ненулевые остатки. Для составного предъявите нетривиальный делитель и соответствующее произведение. Укажите отдельно число проверок `%` в `trial_test`; учтите перебор всех нечётных кандидатов.

**Задание 2. Полная факторизация.**

Разложите $M$ на простые множители и запишите результат канонически. Покажите уменьшение текущего остатка после извлечения каждой степени и объясните остановку перебора. Проверьте простоту всех оснований, положительность показателей и точное восстановление $M$.

**Задание 3. Решето и число операций.**

Постройте решето до $N$ включительно. Выпишите базовые простые $p$ с $p^2 \le N$, итоговый список простых и $\pi(N)$. Для каждого базового простого вычислите число отметок, найдите $C(N)$ и количество различных составных чисел $N - 1 - \pi(N)$. Объясните разницу двух последних величин.

**Задание 4. Достраивание до квадрата.**

По разложению $M$ найдите наименьшее положительное $h$, при котором $Mh = t^2$ для целого $t > 0$. Укажите $h$ и $t$, проверьте равенство и обоснуйте минимальность $h$ через чётность показателей.

**Задание 5. Программная проверка и границы применимости.**

Запустите код раздела 5 со своими данными и общими проверками. Сравните ответы с ручными результатами. Объясните на примере 121 ошибку строгой границы `d * d < n`, на примере 242 — ошибку преждевременного объявления остатка простым. Кратко поясните, почему 0 и 1 получают признак `False`, но не называются составными.

## 8 Варианты индивидуальных заданий

В столбце $N$ дана включённая верхняя граница решета, $a, b$ — числа для проверки простоты, $M$ — число для факторизации и достраивания до квадрата. Демонстрационный пример раздела 4 не заменяет выполнение своего варианта.

| **Вариант** | $N$ | $a$ | $b$ | $M$ |
|---|---|---|---|---|
| 1 | 60 | 103 | 187 | 10584 |
| 2 | 65 | 143 | 101 | 9240 |
| 3 | 70 | 127 | 169 | 13860 |
| 4 | 75 | 131 | 137 | 12600 |
| 5 | 80 | 221 | 289 | 16632 |
| 6 | 85 | 157 | 247 | 17640 |
| 7 | 90 | 341 | 173 | 19800 |
| 8 | 95 | 181 | 299 | 27720 |
| 9 | 100 | 193 | 361 | 29400 |
| 10 | 105 | 437 | 211 | 30800 |
| 11 | 110 | 223 | 227 | 32760 |
| 12 | 115 | 391 | 403 | 37800 |
| 13 | 120 | 239 | 529 | 41580 |
| 14 | 125 | 667 | 251 | 45360 |
| 15 | 130 | 257 | 561 | 48048 |
| 16 | 135 | 263 | 625 | 55440 |
| 17 | 140 | 527 | 271 | 60060 |
| 18 | 145 | 277 | 509 | 65520 |
| 19 | 150 | 589 | 703 | 69300 |
| 20 | 155 | 293 | 899 | 83160 |

## 9 Требования к отчёту

Отчёт должен содержать название работы, Ф. И. О. и группу, номер варианта и исходные данные; ручные решения заданий 1–4; таблицу или изображение собственного решета; исходный код и его вывод; объяснения из задания 5; краткий вывод о проверках, подтвердивших результат. У каждого математического утверждения укажите достаточное обоснование: одного программного `True` для доказательства простоты недостаточно.

## 10 Критерии оценивания

| Составляющая работы | Баллы |
|---|---|
| Проверка $a$, $b$: границы, остатки или делитель, верный подсчёт операций | 20 |
| Полное разложение $M$, ход вычислений и проверка полноты | 25 |
| Решето, список простых, $\pi(N)$, отметки и различные составные | 25 |
| Минимальный множитель $h$, корень $t$ и доказательство | 15 |
| Работающий код, граничные случаи, сопоставление и оформление | 15 |
| Итого | 100 |

## 11 Контрольные вопросы

1. Почему в определении простого требуется $p > 1$?
2. Почему составное число имеет простой делитель не больше своего квадратного корня?
3. Почему проверка только нечётных чисел без отдельной проверки 2 недостаточна?
4. Почему ручной и программный счётчики для 97 могут различаться?
5. Зачем извлекать всю степень найденного множителя?
6. При каком условии текущий остаток факторизации можно объявить простым?
7. Почему решето начинает отметки с $p^2$?
8. Почему при $N = 60$ получаются 57 отметок, но 42 различных составных числа?
9. Как найти минимальный множитель до полного квадрата и доказать минимальность?
10. Почему пробные деления неудобны для числа с большой двоичной длиной?

## Литература

Мартынов Л. М. **Алгебра и теория чисел для криптографии**. Лань, 2020. § 3.3, с. 91–96. Панкратова И. А. **Теоретико-числовые методы в криптографии**. 2009. § 3.1, с. 73–74. Дополнительно: Trappe W., Washington L. C. **Introduction to Cryptography with Coding Theory**. 2nd ed., 2006. § 3.1.2.