МОНОГРАФИИ
НГТУ

Серия основана в 2001 году

В. И. ГУЖОВ

# МОДУЛЬНАЯ АРИФМЕТИКА ДЛЯ ИНЖЕНЕРОВ

НОВОСИБИРСК
2023

УДК 511:51
Г936

Рецензенты:

д-р техн. наук, профессор НГТУ *А. В. Чехонадских*
д-р физ.-мат. наук, профессор ОмГУ *В. И. Струнин*

**Гужов В. И.**

Г936 Модульная арифметика для инженеров : монография / В. И. Гужов. – Новосибирск : Изд-во НГТУ, 2023. – 160 с. – (Монографии НГТУ).

ISBN 978-5-7782-5106-9

В монографии излагаются основы модульной арифметики. Основное внимание уделяется вопросам перевода чисел из модульного представления в позиционное, сравнению чисел и определению переполнения при арифметических операциях. Приводится история развития компьютерных систем на основе модульной арифметики и примеры решения некоторых инженерных вопросов: расширения диапазона измерений интенсивности и устранения фазовой неоднозначности в задачах интерферометрии, голографии и структурированного освещения.

Монография будет интересна научным работникам, инженерам и студентам, специализирующимся в области прикладной математики, криптографических средств защиты информации, информационных систем, оптики и голографии.

**УДК 511:51**

**DOI 10.17212/978-5-7782-5106-9**
**ISBN 978-5-7782-5106-9**

V. I. GUZHOV

# MODULAR ARITHMETIC FOR ENGINEERS

NOVOSIBIRSK
2023

УДК 511:51
Г936

Reviewers:

Professor *A. V. Chekhonadskikh*, D.Sc. (Eng.), NSTU
Professor *V. I. Strunin*, D.Sc. (Phys. & Math.), OSU

**Guzhov V. I.**

Г936 Modular arithmetic for engineers: monograph / V. I. Guzhov. – Novosibirsk : NSTU Publisher, 2023. – 160 p. – (NSTU Monographs)

ISBN 978-5-7782-5106-9

The monograph outlines the fundamentals of modular arithmetic. The main attention is paid to the issues of converting numbers from the modular representation to the positional representation, comparing numbers and determining overflow in arithmetic operations. The history of the development of computer systems based on modular arithmetic and examples of solving some engineering issues are given: expanding the range of intensity measurements and eliminating phase ambiguity in problems of interferometry, holography and structured lighting.

The monograph will be of interest to scientists, engineers and students specializing in the field of applied mathematics, cryptographic information security tools, information systems, optics and holography.

**УДК 511:51**

**DOI 10.17212/978-5-7782-5106-9**
**ISBN 978-5-7782-5106-9**

## ПРЕДИСЛОВИЕ

Модульная арифметика имеет большое значение при выполнении операций сложения, вычитания и умножения. Применение таких модульных вычислений позволяет создавать системы параллельных вычислений, которые имеют значительные преимущества перед системами, используемыми в настоящее время.

Однако существует ряд нерешенных проблем, которые ограничивают развитие систем модульных вычислений:

1) медленное преобразование чисел из модульного представления в позиционное;

2) сложность проверки, какое число больше другого или меньше;

3) невозможность определения, возникло ли переполнение в результате математических операций;

4) неоднозначность при определении операции деления модульных чисел.

Попытка решения этих проблем показана в первом разделе настоящей работы.

Компьютеры с традиционной архитектурой фон Неймана работают с ограниченным размером машинного слова, который определяет максимальный размер числа. В 1960-х годах были предложены системы кодирования целых чисел в машинах для распараллеливания математических расчетов, использующие модульную арифметику, или систему остаточных классов. История развития компьютерных систем на базе модульных вычислений представлена в разделе 2.

Модульные вычисления хорошо подходят для задач шифрования. Некоторые проблемы при использовании модульных вычислений в создании современных систем шифрования рассмотрены в разделе 3.

Использование модульных вычислений в инженерных расчетах встречается очень редко. Одной из причин является то, что модульная арифметика работает с целыми числами. При переводе чисел из множества вещественных в целые возникают проблемы, связанные с устойчивостью решений. Пример использования модульных вычислений при расширении числа уровней квантования в проекционных системах и для увеличения динамического диапазона в проекционных и интерференционных системах приведен в разделах 4 и 5.

Модульная арифметика недостаточно известна. Математические курсы для подготовки специалистов в большинстве высших учебных заведениях не включают этот

раздел математики. Анализ специальной литературы самостоятельно связан с потерей времени на изучение математической терминологии. В настоящей работе сделана попытка изложения основных проблем модульной арифметики на языке, близком к инженерной практике. Основные положения приведены без доказательств, подробное описание которых есть в специальной литературе. Отсутствие доказательств компенсируется примерами, из которых становятся понятны приведенные алгоритмы модульной арифметики.

Каждая глава этой книги написана так, чтобы ее можно было изучить без чтения предыдущих разделов, поэтому имеются повторения некоторых рисунков и выражений. Это не случайно, поскольку таким образом значительно облегчается процесс чтения.

Автор надеется, что изложенные в этой работе проблемы модульной арифметики привлекут широкие слои специалистов, создающих инженерные системы, для практического использования в своей работе.

В заключение хочется поблагодарить д-ра техн. наук С. П. Ильиных и канд. техн. наук Д. С. Хайдукова за помощь и полезные обсуждения при написании книги. Исследования, связанные с устранением нелинейности в разделе 4, выполнены Е. Е. Трубилиной.

## ВВЕДЕНИЕ

Древнегреческие математики активно разрабатывали теорию чисел. Многие результаты дошли до нас в VII–IX книгах «Начал» Евклида (III век до н. э.), которые содержали понятия теории делимости: деления нацело, деления с остатком, простого числа, делителя, кратного и алгоритм Евклида для нахождения наибольшего общего делителя двух чисел.

Одним из важных результатов теории чисел является так называемая китайская теорема об остатках (Chinese Remainder Theorem). Эта теорема утверждает, что можно восстановить целое число по множеству его остатков от деления из некоторого набора попарно взаимно простых чисел. Теорема в ее арифметической формулировке была описана в трактате «Сунь Цзы Суань Цзин» китайского математика Сунь Цзы. Время написания его работы точно не установлено, это может быть 200 год до н. э. или 200 год н. э. В общем виде теорема была доказана Цинь Цзюшао (Чин Чжу-Шао, Qín Jiǔsháo) в его книге «Математические рассуждения в 9 главах», датируемой 1247 годом (*Dickson L. E.* History of the theory of numbers. Vol. II. Washington: Carnegie Institution, 1920).

Примерно в это же самое время (100 год н. э.) греческий математик Никомах сформулировал частный случай этой теоремы. Подобную задачу решал и индийский математик Брахмагупта (588–660).

Дальнейшее развитие теории чисел возобновилось только спустя пятьсот лет. Автором новых идей стал Пьер Ферма (XVII век). В числе прочих он открыл неизвестное древним свойство делимости (малая теорема Ферма), имеющее фундаментальный характер.

Французский математик. По профессии юрист, состоял на государственной службе: с 1631 по 1648 год был уполномоченным по приему прошений, а с 1648 года и до конца жизни – советником парламента Тулузы. Знаток классической литературы, лингвист и поэт.

Математика всегда была для Ферма лишь увлечением, и тем не менее он заложил основы многих ее областей – аналитической геометрии, исчисления бесконечно малых, теории вероятностей.

С именем Ферма связаны две знаменитые теоремы из области теории чисел: малая теорема Ферма и «великая» теорема Ферма, о которой на полях трудов Диофанта он написал: «Я нашел этому поистине чудесное доказательство, но эти поля слишком малы для него». Согласно этой теореме уравнение $a^n + b^n = c^n$ при $n > 2$ не имеет целых положительных корней. Доказательство теоремы в общем виде было получено лишь в 1994 году.

Однажды ученый получил письмо с вопросом: «Является ли простым число 100895598169?». Ферма незамедлительно ответил, что это двенадцатизначное число является произведением двух простых чисел: 898423 и 112303.

Пьер де Ферма
(Pierre de Fermat)
(1601–1665)

Исследования Ферма были продолжены и углублены Леонардом Эйлером, который основал теорию квадратичных и других степенных вычетов, открыл «тождество Эйлера», в общем виде сформулировал и доказал теорему об остатках в 1734 году.

Леонард Эйлер
(нем. Leonhard Euler)
(1707–1783)

Швейцарский ученый, внесший фундаментальный вклад в развитие математики и механики, а также физики, астрономии и ряда прикладных наук. Автор более 850 работ (включая 20 фундаментальных монографий) по математическому анализу, дифференциальной геометрии, теории чисел, приближенным вычислениям, небесной механике, математической физике, оптике, баллистике, кораблестроению, теории музыки и др.

Почти полжизни провел в России и внес существенный вклад в становление российской науки. В 1726 году был приглашен работать в Санкт-Петербург, куда переехал годом позже. С 1726 по 1741 год, а также с 1766 года был академиком Петербургской академии наук; в 1741–1766 годах работал в Берлине (оставаясь одновременно почетным членом Петербургской академии). Уже через год пребывания в России хорошо знал русский язык и часть своих сочинений (особенно учебники) публиковал на русском.

Первые русские академики, математик С. К. Котельников и астроном С. Я. Румовский, были учениками Эйлера.

Несколько крупных открытий сделал Ж. Л. Лагранж, а А.-М. Лежандр опубликовал монографию «Опыт теории чисел» (1798), первое в истории обстоятельное изложение этого раздела математики. К концу XVIII века был достигнут прогресс в изучении непрерывных дробей, решении различных типов уравнений в целых числах (Валлис, Эйлер, Лагранж), а также положено начало исследованию распределения простых чисел (Лежандр).

Современный подход к модульной арифметике был разработан Карлом Фридрихом Гауссом в его книге «Disquisitiones Arithmeticae», опубликованной в 1801 году. Гаусс начал работу над своей книгой «Арифметические исследования» (лат. disquisitiones arithmeticae) еще в 20-летнем возрасте (1797). Из-за неспешности работы местной типографии работа над книгой растянулась на четыре года, кроме того, Гаусс стремился публиковать только завершенные исследования, пригодные для непосредственного практического применения. В отличие от Лежандра, Гаусс предложил не просто перечень теорем, а систематическое изложение теории на основе единых идей и принципов. Все рассмотренные проблемы были доведены до уровня алгоритма; книга содержит множество численных примеров, таблиц и пояснений.

Гаусс считается одним из величайших математиков всех времен, «королем математиков». Родился в немецком герцогстве Брауншвейг. Дед Гаусса был бедным крестьянином; отец, Гебхард Дитрих Гаусс, – садовником, каменщиком, смотрителем каналов; мать, Доротея Бенц, – дочерью каменщика. Будучи неграмотной, мать не записала дату рождения сына, запомнив только, что он родился в среду, за восемь дней до праздника Вознесения, который отмечается спустя 40 дней после Пасхи. В 1799 году Гаусс вычислил точную дату своего рождения, разработав метод определения даты Пасхи на любой год.

Уже в двухлетнем возрасте мальчик показал себя вундеркиндом. В три года он умел читать и писать, даже исправлял арифметические ошибки отца. Известна история, в которой юный Гаусс определил сумму чисел от 1 до 100 быстрее своих одноклассников. До самой старости большую часть вычислений он производил в уме.

В 1832 году создал абсолютную систему мер, введя три основные единицы: единицу времени – 1 с, единицу длины – 1 мм, единицу массы – 1 мг.

Гаусс писал: «Математика – царица наук, а теория чисел – царица математики».

Иоганн Карл Фридрих Гаусс (нем. Johann Carl Friedrich Gauß) (1777–1855)

В традиционной позиционной системе счисления значение каждого числового знака (цифры) при обозначении числа зависит от его позиции (или разряда) при записи числа. К позиционным системам счисления относятся такие известные системы, как десятичная и двоичная; название определяется основаниями этих систем. Используются также позиционные системы с основаниями 60, 16, 12 и 8. Основанием системы, в принципе, может быть выбрано любое число.

Кроме позиционных существуют также непозиционные системы счисления, где запись чисел основана на других принципах. Один из примеров таких систем – известные римские цифры, которые записываются в виде символов, означающих значение цифры: I – единица, V – пять, X – десять, L – пятьдесят, C – сто. Например, число 176 запишется как CLXXVI.

Другой пример непозиционных систем – это система счисления остаточных классов (СОК), или модульная (модулярная) система, которая является системой представления данных в вычислительной арифметике. В модульной системе счисления многоразрядное целое число (являющееся таковым в позиционной системе счисления) представляется в виде последовательности нескольких позиционных чисел. Эти числа суть остатки (вычеты) от деления исходного числа на некоторые модули – взаимно простые числа.

Взаимно однозначное соответствие между некоторым числом и набором его остатков, определяемым набором взаимно простых чисел, на практике помогает

работать не с длинными числами, а с наборами их коротких по длине остатков. Кроме того, вычисления по каждому из модулей можно выполнять параллельно.

Такое представление числа под названием «системы остаточных классов» использовалось в специализированных компьютерах, так как позволяло выполнять арифметические операции над числами с большим числом битов путем параллельного вычисления операций над остатками. Поскольку модули можно было взять с небольшим числом битов, то таблицы операций над остатками по каждому модулю можно было просто запомнить в памяти как таблицу, что позволяло производить арифметическую операцию практически за один такт.

Первые успешные попытки применения модульной арифметики в микроэлектронике были предприняты еще в 1950-х годах, но из-за сложностей работы с модульными операциями интерес к ним постепенно снизился.

В настоящее время заметно увеличилось количество публикаций по модульной арифметике. Более чем 150 работ в этом направлении было опубликовано в период с середины 1970-х до середины 1980-х годов [1]. С середины 1970-х теоретические разработки начали применяться в технологических разработках. Первоначально основной областью применения модульных вычислений являлась цифровая обработка сигнала. Хуан (Huang C. H.) построил и испытал согласованный фильтр в двумерном СОК [2]. В этот же период Смит (Smith W.) разработал быстрое преобразование Фурье (БПФ) с использованием модульной арифметики [3].

В настоящее время системы счисления остаточных классов применяются при разработке эффективных и высокопроизводительных процессоров специального назначения. Использование модульной арифметики для аппаратных средств со сверхвысокой степенью интеграции в системах СОК описана в работе Тейлора (Taylor F. J.) [4].

СОК широко используется и в криптографии. Например, модульная арифметика позволяет создавать эффективную аппаратную реализацию криптографических систем [5]. Применение непозиционной системы счисления позволяет ускорить медленные вычисления в асимметричных алгоритмах шифрования и повысить их надежность [6–9]. Модульная арифметика широко используется при аппаратной реализации алгоритмов RSA (аббревиатура от фамилий Rivest, Shamir и Adleman) и ECC (Elliptic Curve Cryptography).

Цель настоящей монографии – показать основные возможности вычислений при использовании модульной арифметики в инженерной практике и представить некоторые пути преодоления основных проблем при использовании вычислительных операций в системе остаточных классов.

# 1. МОДУЛЬНАЯ АРИФМЕТИКА

Основная идея модульной арифметики состоит в том, чтобы работать не с самими числами, а с остатками их деления на какое-либо число. Арифметические операции с остатками чисел по фиксированному модулю образуют модульную (модулярную) арифметику.

## 1.1. ОСНОВНЫЕ ПОНЯТИЯ МОДУЛЬНОЙ АРИФМЕТИКИ

Если разделить одно целое число на другое, то получим следующее выражение:

$$\frac{L}{m} = N + r, \tag{1.1.1}$$

где $L$ – делимое; $m$ – делитель; $N$ – целый делитель; $r$ – остаток.

Пусть каждому целому числу $L$ отвечает определенный остаток $r$ от деления на целое положительное $m$, который называется модулем. Если двум целым $a$ и $b$ отвечает один и тот же остаток *r*, то эти числа называются равноостаточными по модулю $m$.

Числа, имеющие одинаковые остатки $r$ от деления на целое $m$, называются сравнимыми по модулю $m$. Числа, сравнимые по модулю, образуют класс чисел по модулю $m$. Все числа класса можно записать в виде $mq + r + n_0$. Можно получить все числа класса, если $q$ будет пробегать все целые числа. Например: $\frac{13}{5} = 2 + \text{остаток } 3 \rightarrow 13 \,(\text{mod } 5) = 3$ или $13 \equiv 3 \,(\text{mod } 5)$.

Сравнимость чисел $a$ и $b$ записывается как

$$a \equiv b \,(\text{mod } m), \tag{1.1.2}$$

где знак «≡» обозначает операцию сравнения.

### 1.1.1. СРАВНЕНИЕ ПО МОДУЛЮ

Предпосылкой к созданию теории сравнений стало восстановление сочинений Диофанта, которые были выпущены в подлиннике и с латинским переводом в 1621 году. Их изучение привело Ферма к открытиям, которые существенно опередили свое время. Понятие и символьное обозначение сравнений было введено Гауссом как важный инструмент для обоснования его арифметической теории, работа над которой была начата им в 1797 году. В начале этого периода Гауссу еще не были известны труды предшественников, поэтому результаты его работы, изложенные в первых трех главах его книги «Арифметические исследования» (1801), были в основном уже известны. Однако методы, которые он использовал для доказательств, оказались абсолютно новыми, имеющими высшую важность для развития теории чисел.

Модульная арифметика часто используется в реальной жизни. В качестве примера можно привести 12-часовые циферблаты, в которых для определения времени суток используется модуль 12 (рис. 1.1.1), т. е. часы «переключаются» каждые 12 часов.


| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |

*Рис. 1.1.1.* Соответствие показаний на циферблате и времени суток

Когда мы говорим, что сейчас 14 часов, мы можем также сказать, что сейчас 2 часа дня, т. е.

$$14 \equiv 2 \pmod{12}. \tag{1.1.3}$$

Тот же принцип применяется и для измерения углов. Угол в 370 градусов равен углу в 10 градусов. Угол в 750 градусов равен углу 30 градусов:

$$750 \equiv 30 \pmod{360}. \tag{1.1.4}$$

Арифметика по модулю 7 используется в алгоритмах, определяющих день недели для заданной даты.

### 1.1.2. КЛАССЫ ВЫЧЕТОВ

Множество всех чисел, сравнимых с $a$ по модулю $m$, называется классом вычетов $a$ по модулю $m$. Полная система вычетов по модулю $m$ – это любой набор из $m$ попарно несравнимых по модулю $m$ целых чисел [10, 11].

Однако обычно в качестве полной системы вычетов берется множество наименьших неотрицательных вычетов $0, 1, \ldots, m-1$. В этом случае число $a$ называется вычетом числа $b$ по модулю $m$, если разность $a - b$ делится на $m$ ($a$, $b$, $m > 0$ – целые числа).

Максимальный набор попарно несравнимых по модулю $m$ чисел, взаимно простых с $m$, называется приведенной системой вычетов по модулю $m$.

Всякая приведенная система вычетов по модулю $m$ содержит $\varphi(m)$ элементов, где $\varphi(m)$ – функция Эйлера [1].

Например, для числа $m = 42$ полная система вычетов может быть представлена числами

0, 1, 2, 3, …, 21, 22, 23, …, 39, 40, 41,

а приведенная

1, 5, 11, 13, 17, 19, 23, 25, 29, 31, 37, 41.

Система вычетов позволяет осуществлять арифметические операции над конечным набором чисел, не выходя за его пределы.

### 1.1.3. ПРЕДСТАВЛЕНИЕ ОТРИЦАТЕЛЬНЫХ ЧИСЕЛ

В качестве полной системы вычетов можно взять абсолютно наименьшие вычеты. В случае нечетного числа это будет ряд

$$-\frac{m-1}{2}, \ldots, -1, 0, 1, 2, \ldots, \frac{m-1}{2}\ , \qquad (1.1.5а)$$

а в случае четного – один из следующих рядов:

$$-\frac{m}{2}+1, \ldots, -1, 0, 1, 2, \ldots, \frac{m}{2} \qquad (1.1.5б)$$

или

$$-\frac{m}{2}, \ldots, -1, 0, 1, 2, \ldots, \frac{m}{2}-1. \qquad (1.1.5в)$$

Например, для $m = 5$ полная система вычетов может быть представлена числами

$$-2, -1, 0, 1, 2,$$

а для $m = 6$ – числами

$$-3, -2, -1, 0, 1, 2 \text{ или } -2, -1, 0, 1, 2, 3.$$

Схема отображения наименьших неотрицательных вычетов на систему вычетов наименьших по величине при $m = 11$ показана на рис. 1.1.2. После восстановления числа по набору остатков сравниваем результат с серединой, и если он больше, то выводим минус и инвертируем результат (т. е. отнимаем его от произведения всех простых и выводим уже его).

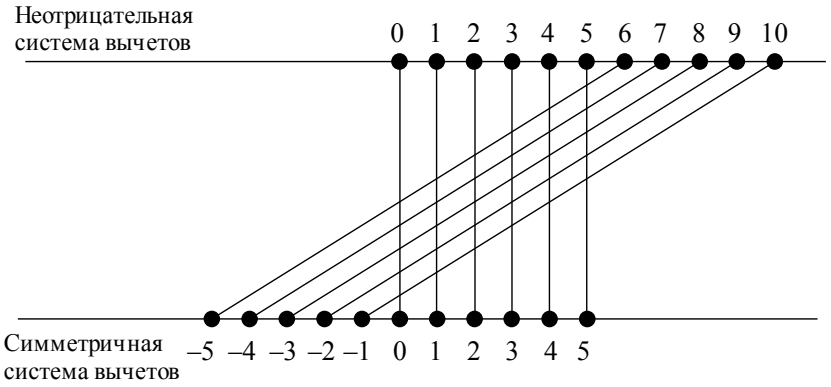



*Рис. 1.1.2.* Отображение наименьших неотрицательных вычетов на систему вычетов, наименьших по величине при нечетном $m$

Пример вычитания модульных чисел:

$$2 \pmod{11} - 5 \pmod{11} = -3 \pmod{11} = (-3 + 11) \pmod{11} = 8 \pmod{11}.$$

В случае четного значения $m$ система вычетов будет иметь вид

$$-\frac{m}{2}+1, \ldots, -1, 0, 1, \ldots, \frac{m}{2}, \tag{1.1.6}$$

или

$$-\frac{m}{2}, \ldots, -1, 0, 1, \ldots, \frac{m}{2}-1. \tag{1.1.7}$$

Для (1.1.6) схема отображения показана на рис. 1.1.3 ($m = 10$).

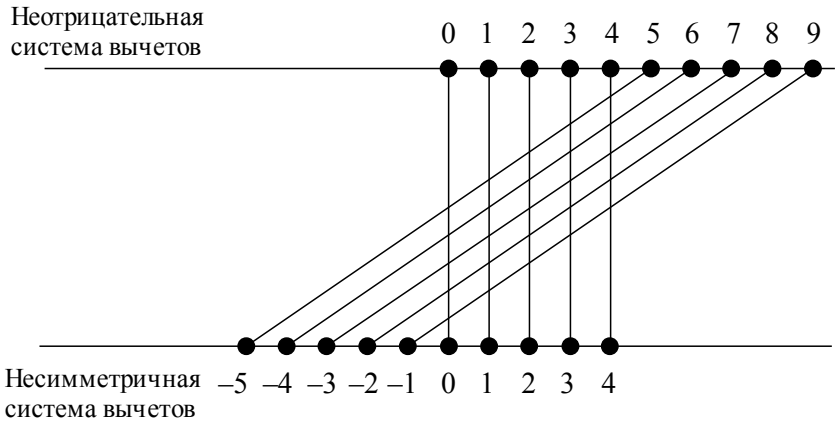



*Рис. 1.1.3.* Отображение наименьших неотрицательных вычетов на систему вычетов (1.1.6), наименьших по величине при четном $m$

Для (1.1.7) схема отображения показана на рис. 1.1.4 ($m = 10$).

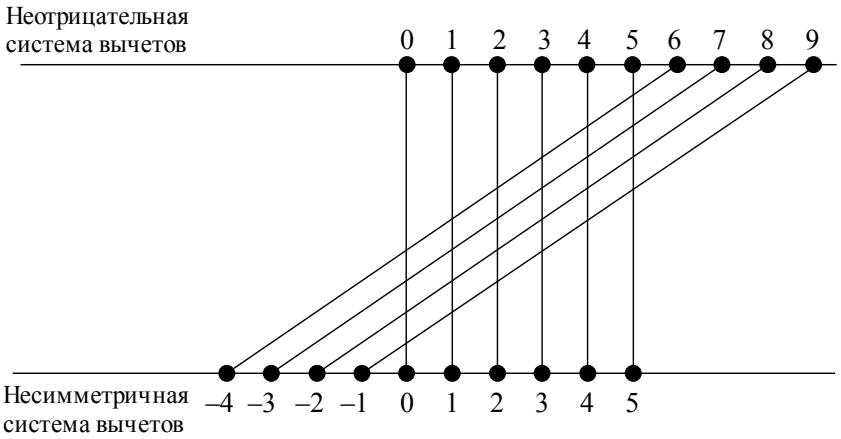



*Рис. 1.1.4.* Отображение наименьших неотрицательных вычетов на систему вычетов (1.1.7), наименьших по величине при четном $m$

Примеры вычитания модульных чисел:

$$2\,(\mathrm{mod}10) - 5\,(\mathrm{mod}10) = -3\,(\mathrm{mod}10) = (-3+10)\,(\mathrm{mod}10) = 7\,(\mathrm{mod}10);$$

$$2\,(\mathrm{mod}10) - 6\,(\mathrm{mod}10) = -4\,(\mathrm{mod}10) = (-4+10)\,(\mathrm{mod}10) = 6\,(\mathrm{mod}10);$$

$$1\,(\mathrm{mod}10) - 6\,(\mathrm{mod}10) = -5\,(\mathrm{mod}10) = (-5+10)\,(\mathrm{mod}10) = 5\,(\mathrm{mod}10).$$

Во многих языках программирования и калькуляторах есть операция взятия остатка (деления по модулю). В С-подобных языках она обозначается символом «%». Если вы делите по модулю отрицательное число, ответ может также получиться отрицательным. Например: $(-5)\%3 = -2$.

В модульной арифметике можно проводить операцию вычитания с наименьшими неотрицательными вычетами $0, 1, \ldots, m-1$. Перевод в отрицательные числа необходимо делать только при сравнении или восстановлении длинных чисел.

### 1.1.4. СВОЙСТВА СРАВНЕНИЙ

Для натурального числа $a$ отношение сравнимости по модулю $m$ обладает следующими свойствами:

- свойством ***рефлексивности***: для любого целого $a$ справедливо $a \equiv a \pmod{m}$;
- свойством ***симметричности***: если $a \equiv b \pmod{m}$, то $b \equiv a \pmod{m}$;
- свойством ***транзитивности***: если $a \equiv b \pmod{m}$ и $b \equiv c \pmod{m}$, то $a \equiv c \pmod{m}$.

Кроме вышеперечисленных свойств для сравнений справедливы следующие утверждения:

- любые два целых числа сравнимы по модулю 1;
- если числа $a$ и $b$ сравнимы по модулю $m$ и число $d$ является делителем $m$, то $a$ и $b$ сравнимы по модулю $d$.

Сравнения по одному и тому же модулю обладают многими свойствами обычных равенств.

1. Если $a_i \equiv b_i \pmod{m}$, $i = 1, \ldots, n$, то

$$\begin{gathered}(a_1 + a_2 + \ldots + a_n) \equiv (b_1 + b_2 + \ldots + b_n) \pmod{m};\\ (a_1 a_2 \ldots a_n) \equiv (b_1 b_2 \ldots b_n) \pmod{m}.\end{gathered} \tag{1.1.8}$$

При этом нельзя выполнять операции со сравнениями, если их модули не совпадают.

2. К обеим частям сравнения можно прибавить одно и то же число $c$:

$$(a + c) \equiv (b + c) \pmod{m}.$$

3. Можно перенести число из одной части сравнения в другую со сменой знака:

$$a \equiv (b + c) \pmod{m};$$

$$(a - c) \equiv b \pmod{m}.$$

4. Если числа $a$ и $b$ сравнимы по модулю $m$, то их степени $a^k$ и $b^k$ тоже сравнимы по модулю $m$ при любом натуральном $k$:

$$a^k \equiv b^k \pmod{m}.$$

5. Если $a \equiv b \pmod{m}$ и $t_1$ и $t_2$ – произвольные целые числа, кратные $m$, то

$$(a + t_1) \equiv (b + t_2) \pmod{m}.$$

6. Обе части сравнения и модуль можно умножить на одно и то же число: если $a \equiv b \pmod{m}$ и $q$ – целое, то

$$aq \equiv bq \pmod{mq}.$$

7. Однако сравнения нельзя, вообще говоря, делить друг на друга или на другие числа. Правила сокращения для сравнений следующие:

- можно делить обе части сравнения на число, но только взаимно простое с модулем: если $ad \equiv bd \pmod{m}$ и НОД $(d, m) = 1$, то $a \equiv b \pmod{m}$. Если число $d$ не взаимно простое с модулем, то сокращать на $d$ нельзя;
- можно одновременно разделить обе части сравнения и модуль на их общий делитель: если $aq \equiv bq \pmod{mq}$, то $a \equiv b \pmod{m}$.

### 1.1.5. СИСТЕМЫ СРАВНЕНИЙ

Все целые числа можно разделить на классы эквивалентности: два числа находятся в одном классе, если они сравнимы по модулю $m$.

Рассмотрим систему сравнений первой степени с одним неизвестным:

$$\begin{cases} L \equiv b_1 \bmod(m_1); \\ \quad \cdots \\ L \equiv b_n \bmod(m_n), \end{cases} \tag{1.1.9}$$

где $b_i$ – остаток от деления $L$ на число $m_i$.

Решить систему сравнений (1.1.9) значит найти число $L$ по его остаткам $(b_1, b_2, \ldots, b_n)$.

Если модули – взаимно простые числа, то максимальное расширение диапазона определяется произведением этих модулей: $m_1 m_2 \ldots m_n$. Это следует из китайской

теоремы об остатках. В своей современной формулировке она записывается следующим образом.

**Теорема**

Пусть $M = m_1 m_2 \ldots m_n$, где $m_i$ – попарно взаимно простые числа.

Поставим в соответствие произвольному числу $L$ $(0 \le L < M)$ набор остатков $(b_1, b_2, \ldots, b_n)$, где $b_i \equiv L \pmod{m_i}$ : $L \Leftrightarrow (b_1, b_2, \ldots, b_n)$.

Тогда это соответствие (между числом и набором остатков) будет являться **взаимно однозначным**.

Целые числа взаимно просты, если их наибольший общий делитель (НОД) равен единице. Попарная простота модулей обозначает, что

$$\text{НОД}\,(m_i, m_j) = 1, \quad \forall\, i, j \in \{1, 2, \ldots, n\}, \quad i \ne j\,.$$

Из этой теоремы следует, что операции, выполняемые с числом *a*, можно эквивалентно выполнять с соответствующими наборами остатков путем независимого выполнения операций над каждым компонентом.

### 1.1.6. АРИФМЕТИЧЕСКИЕ ОПЕРАЦИИ В МОДУЛЬНОЙ АРИФМЕТИКЕ

Операции сложения, вычитания, умножения и деления, основанные на работе с наборами остатков, составляют модульную, или модулярную, арифметику (modular arithmetic). В модульной арифметике определяются основные арифметические операции сложения, вычитания, умножения и деления. Кроме этого, введение отношения порядка позволяет установить операцию сравнения по величине и связанные с ней действия (оценку переполнения арифметической суммы или произведения чисел).

Область чисел, над которыми можно выполнять операции модульной арифметики, определяется произведением модулей $(M = m_1 m_2 \ldots m_n)$.

Преимущество модульного представления состоит в том, что операции сложения, вычитания и умножения выполняются очень просто [12]:

$$
\begin{aligned}
&(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) \bmod (m_1, m_2, \ldots, m_n) = \\
&\quad = (a_1 + b_1) \bmod m_1, \ldots, (a_n + b_n) \bmod m_n; \\
&(a_1, a_2, \ldots, a_n) - (b_1, b_2, \ldots, b_n) \bmod (m_1, m_2, \ldots, m_n) = \\
&\quad = (a_1 - b_1) \bmod m_1, \ldots, (a_n - b_n) \bmod m_n; \\
&(a_1, a_2, \ldots, a_n)(b_1, b_2, \ldots, b_n) \bmod (m_1, m_2, \ldots, m_n) = \\
&\quad = (a_1 b_1) \bmod m_1, \ldots, (a_n b_n) \bmod m_n.
\end{aligned}
\tag{1.1.10}
$$

Если предусмотрена возможность параллельного выполнения операций, то применение модульной арифметики дает для таких операций значительное преимущество при вычислениях. Операции, связанные с разными модулями, могут выполняться одновременно, что приводит к сокращению времени их выполнения.

Однако существует ряд недостатков модульного представления, которые ограничивают его использование:

1) необходимость преобразования чисел из модульного представления в позиционное;

2) трудность проверки, какое число в модульном представлении больше или меньше, чем другое;

3) сложность определения переполнения в результате математических операций над модульными числами;

4) проблемы при определении операции деления модульных чисел.

Такие операции можно осуществлять только над числами в позиционной системе счисления, т. е. число из модульного представления необходимо перевести к «длинному» числу. В большинстве случаев при этом все преимущества модульного исчисления исчезают.

В следующих разделах рассмотрим некоторые попытки решения перечисленных недостатков модульных вычислений.

## 1.2. ПЕРЕХОД ОТ ПОЗИЦИОННОГО ПРЕДСТАВЛЕНИЯ ЧИСЕЛ К МОДУЛЬНОМУ

Если есть число $L$, то переход в модульное представление $(b_1, b_2, \ldots, b_n)$ можно получить, разделив $L$ на модули $(m_1, m_2, \ldots, m_n)$ и взяв остатки от деления.

При работе на компьютерах, использующих двоичную систему, желательно выбирать модули следующим образом: $m_j = 2^{k_j} - 1$, т. е. модуль на единицу меньше

двойки в некоторой степени. Такой выбор модулей часто упрощает выполнение основных арифметических операций.

Для работы с модулями вида $m_j = 2^{k_j} - 1$ надо знать, при каких условиях число $2^e - 1$ взаимно простое с числом $2^f - 1$. Существует очень простое правило [12]:

$$\text{НОД}\,(2^e - 1,\ 2^f - 1) = 2^{\text{НОД}\,(e,\,f)} - 1, \tag{1.2.1}$$

где $\text{НОД}\,(a, b)$ – наибольший общий делитель чисел $a$ и $b$. Это правило утверждает, в частности, что $2^e - 1$ и $2^f - 1$ являются простыми тогда и только тогда, когда $e$ и $f$ взаимно простые.

Поэтому на компьютере с длиной слова 32 бита можно выбрать $m_1 = 2^{32} - 1,\ m_2 = 2^{31} - 1,\ m_2 = 2^{29} - 1,\ \ m_2 = 2^{27} - 1,\ m_2 = 2^{25} - 1$, что обеспечивает эффективность сложения, вычитания и умножения целых чисел в интервале вплоть до $m_1, m_2, \ldots, m_5 > 2^{143}$.

При работе с числами в двоичной системе и модулями вида $m_j = 2^{k_j} - 1$ переход в модульное представление имеет очень простой вид.

Рассмотрим двоичное представление числа $L$ с блоками, состоящими из $k_j$ битов, сгруппированных вместе:

$$u = a_t A^t + a_{t-1} A^{t-1} + \ldots + a_1 A + a_0, \tag{1.2.2}$$

где $A = 2^{k_j}$ и $0 \le a_j < 2^{k_j}$ при $0 \le j \le t$. Тогда

$$u = a_t + a_{t-1} + \ldots + a_1 + a_0 (\bmod 2^{k_j} - 1), \tag{1.2.3}$$

поскольку $A \equiv 1$. Поэтому мы можем получить $u_j$, прибавляя состоящие из $k_j$ битов числа:

$$u_j = a_t \oplus a_{t-1} \oplus \ldots \oplus a_1 \oplus a_0 (\bmod 2^{k_j} - 1), \tag{1.2.4}$$

где операция $\oplus$ – сложение по модулю 2. Действие этой операции над двоичными числами показано в табл. 1.2.1.

Таблица 1.2.1

**Операция «сложение по модулю 2»**

| $a$ | $b$ | $a \oplus b$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Пусть $m_1 = 2^8 - 1, \quad m_2 = 2^7 - 1$.

В этом случае максимальный диапазон числа, которое можно представить в системе остаточных классов, – от нуля до $m_1 m_2 - 1$, т. е. до 32 767.

В восьми битах можно записать число в диапазоне от нуля до 255, в семи битах – в диапазоне от нуля до 127, в 15 битах – от нуля до $m_1 m_2 - 1$, или от нуля до 32 767.

**Пример**

Пусть число $a = 1025_{10} = 10\,000\,000\,001_2$, представим его как остаток от деления на некоторый модуль.

По модулю $m_1 = 2^8 - 1 = 255$. Разбиваем число $a$ на блоки по 8 бит и складываем по модулю 2:

$$00000001 \oplus 00000100 = 00000101, \qquad a = 5 \pmod{255}.$$

По модулю $m_2 = 2^7 - 1 = 127$. Разбиваем на блоки по 7 бит:

$$0000001 \oplus 00001000 = 00001001, \qquad a = 9 \pmod{127}.$$

Перевод от чисел из модульной формы в позиционную выполняется несколько сложнее.

## 1.3. ПЕРЕХОД ОТ МОДУЛЬНОГО ПРЕДСТАВЛЕНИЯ ЧИСЕЛ К ПОЗИЦИОННОМУ

В модульном представлении каждому числу $L$ ставится в соответствие набор остатков $(b_1, b_2, \ldots, b_n)$. Метод модульной арифметики состоит в том, чтобы оперировать не непосредственно с числом $L$, а с его остатками от деления на некоторые

числа $m_i$. Нужно научиться восстанавливать число $L$ по набору остатков $(b_1, b_2, \ldots, b_n)$.

Рассмотрим систему сравнений первой степени с одним неизвестным:

$$\begin{cases} L \equiv b_1 \bmod (m_1); \\ \quad \cdots \\ L \equiv b_n \bmod (m_n), \end{cases} \tag{1.3.1}$$

где $b_i$ – это остаток от деления $L$ на число $m_i$.

Переход от модульного представления к позиционному может происходить, когда все вычисления закончены. В этом случае время выполнения не является значимым. Однако в случаях выполнения сравнения или определения переполнения необходимы быстрые алгоритмы перехода к позиционному представлению чисел.

Наиболее известным является метод, основанный на китайской теореме об остатках.

### 1.3.1. ПЕРЕХОД ОТ МОДУЛЬНОГО ПРЕДСТАВЛЕНИЯ К ПОЗИЦИОННОМУ НА ОСНОВЕ КИТАЙСКОЙ ТЕОРЕМЫ ОБ ОСТАТКАХ

Для нахождения числа $L$ по набору остатков можно использовать конструктивную формулировку теоремы об остатках [11]. Если модули – взаимно простые числа, то максимальное расширение диапазона определяется произведением этих модулей: $m_1 m_2 \ldots m_n$. Числа взаимно (попарно) простые, если НОД $(m_i, m_j) = 1$, $i \neq j$.

**Теорема**

Пусть числа $M_i$ и $N_i$ определены из условий:

$$m_1 m_2 \ldots m_n = M_i m_i,$$
$$M_i N_i \equiv 1 \pmod{m_i}$$

и пусть

$$X_0 \equiv M_1 N_1 b_1 + M_2 N_2 b_2 + \ldots + M_n N_n b_n (\bmod m_1\ m_2 \ldots m_n).$$

Тогда совокупность значений $L$, удовлетворяющих системе сравнений (1.3.1), определяется сравнением

$$L \equiv X_0 (\bmod m_1\ m_2 \ldots m_n).$$

Из теоремы об остатках имеем

$$L \equiv M_1 N_1 b_1 + M_2 N_2 b_2 + \ldots + M_n N_n b_n (\mathrm{mod}\, m_1\, m_2 \ldots m_n). \qquad (1.3.3)$$

Таким образом, для определения числа $L$ необходимо определить все $M_i$ и $N_i$.

Коэффициенты $M_i$ найти очень просто из равенства $m_1\, m_2 \ldots m_n = M_i m_i$, откуда

$$M_i = \frac{m_1\, m_2 \ldots m_n}{m_i}. \qquad (1.3.4)$$

Для нахождения $N_i$ необходимо найти решения сравнения $M_i N_i \equiv 1 \ (\mathrm{mod}\, m_i)$ относительно $N_i$.

Наиболее простым способом найти $N_i$ из сравнения является перебор чисел 1, 2, 3 и так далее до первого удовлетворяющего данному сравнению значения. Это бывает оправданно, когда операцию определения коэффициентов $M_i$ и $N_i$ делают только один раз перед началом работы. Однако удобнее вычислять значения $N_i$ в процессе работы.

Обозначив $a = M_i$ и $x = N_i$, находим решение сравнения $ax \equiv 1 (\mathrm{mod}\, m_i)$.

Пусть у нас есть система из двух сравнений:

$$\begin{cases} L \equiv b_1 \ (\mathrm{mod}\, 11); \\ L \equiv b_2 \ (\mathrm{mod}\, 17). \end{cases} \qquad (1.3.5)$$

Из формулы (1.3.4) имеем $M_1 = \dfrac{11 \cdot 17}{11} = 17,\ M_2 = \dfrac{11 \cdot 17}{17} = 11$.

Решения для сравнений $17x \equiv 1 \ (\mathrm{mod}11)$ и $11x \equiv 1 \ (\mathrm{mod}17)$ приведены в разделе 1.1.5 «Системы сравнений». В результате мы получили $N_1 = 2$ и $N_2 = 14$.

Таким образом, решение системы сравнений (1.3.5) будет иметь следующий вид:

$$L \equiv 17 \cdot 2\, b_1 + 11 \cdot 14\, b_2 \equiv 34 b_1 + 154 b_2 \ (\mathrm{mod}187). \qquad (1.3.6)$$

Просчитав значения для всех остатков $(b_1, b_2)$, получим таблицу, показанную на рис. 1.3.1.

| | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b1 | | | | | | | | | | | | | | | | | | |
| 0 | | 0 | 154 | 121 | 88 | 55 | 22 | 176 | 143 | 110 | 77 | 44 | 11 | 165 | 132 | 99 | 66 | 33 |
| 1 | | 34 | 1 | 155 | 122 | 89 | 56 | 23 | 177 | 144 | 111 | 78 | 45 | 12 | 166 | 133 | 100 | 67 |
| 2 | | 68 | 35 | 2 | 156 | 123 | 90 | 57 | 24 | 178 | 145 | 112 | 79 | 46 | 13 | 167 | 134 | 101 |
| 3 | | 102 | 69 | 36 | 3 | 157 | 124 | 91 | 58 | 25 | 179 | 146 | 113 | 80 | 47 | 14 | 168 | 135 |
| 4 | | 136 | 103 | 70 | 37 | 4 | 158 | 125 | 92 | 59 | 26 | 180 | 147 | 114 | 81 | 48 | 15 | 169 |
| 5 | | 170 | 137 | 104 | 71 | 38 | 5 | 159 | 126 | 93 | 60 | 27 | 181 | 148 | 115 | 82 | 49 | 16 |
| 6 | | 17 | 171 | 138 | 105 | 72 | 39 | 6 | 160 | 127 | 94 | 61 | 28 | 182 | 149 | 116 | 83 | 50 |
| 7 | | 51 | 18 | 172 | 139 | 106 | 73 | 40 | 7 | 161 | 128 | 95 | 62 | 29 | 183 | 150 | 117 | 84 |
| 8 | | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 | 8 | 162 | 129 | 96 | 63 | 30 | 184 | 151 | 118 |
| 9 | | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 | 9 | 163 | 130 | 97 | 64 | 31 | 185 | 152 |
| 10 | | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 | 10 | 164 | 131 | 98 | 65 | 32 | 186 |

*Рис. 1.3.1.* Таблица решений для модулей $m_1 = 11,\ m_2 = 17$

При нахождении коэффициентов $M_i N_i$ можно столкнуться с фактом, что эти числа слишком велики. Поэтому для определения $L$ придется проводить операции с большими (длинными) числами, чего хотелось бы избежать.

Для перехода от набора остатков $(b_1, b_2, \ldots, b_n)$ к самому числу $L$ нужен лучший конструктивный способ перехода от модульного к позиционному представлению.

Рассмотрим способ, предложенный в 1958 году Х. Л. Гарднером [12].

### 1.3.2. АЛГОРИТМ ГАРДНЕРА

Находится решение системы сравнений (1.3.1).

Алгоритм Гарднера основан на использовании констант $c_{ij}$, которые определяются как решения сравнения

$$c_{ij} m_i \equiv 1 \pmod{m_j}. \qquad (1.3.7)$$

Константы $c_{ij}$ легко вычисляются с помощью алгоритма Евклида [11]. Процесс решений показан в разделе 1.6.1 «Нахождение решений сравнений первой степени на основе теории непрерывных дробей».

После их нахождения получим

$$
\begin{aligned}
&v_1 \leftarrow b_1 \bmod m_1;\\
&v_2 \leftarrow (b_2 - v_1)\, c_{12} \bmod m_2;\\
&v_3 \leftarrow \big((b_3 - v_1)\, c_{13} - v_2\big) c_{23} \bmod m_3;\\
&\ldots\\
&v_n \leftarrow \big(\ldots\big((b_n - v_1)\, c_{1n} - v_2\big) c_{2n} - \ldots - v_{n-1}\big) c_{(n-1)n} \bmod m_n.
\end{aligned}
\tag{1.3.8}
$$

В этом случае число в позиционном виде определяется как

$$L = v_n m_{n-1} \ldots m_2 + \ldots + v_3 m_2 m_1 + v_2 m_1 + v_1. \tag{1.3.9}$$

Стоит заметить, что на практике вычислять ответ нужно с помощью длинной арифметики, но при этом сами коэффициенты $v_j$ вычисляются с помощью модульной арифметики, и поэтому алгоритм Гарднера является достаточно эффективным.

**Пример**

Найдем решения системы двух сравнений с модулями $m_1 = 11,\ m_2 = 17$ (1.3.5):

$$\begin{cases} L \equiv b_1 \pmod{11};\\ L \equiv b_2 \pmod{17}. \end{cases}$$

Решения этой системы показаны в таблице на рис. 1.3.1.

Получим решения по методу Гарднера для остатков $(b_1, b_2) = (4, 4)$ и $(b_1, b_2) = (4, 9)$.

Для остатков $(b_1, b_2) = (4, 4)$:

$c_{ij} m_i \equiv 1 \pmod{m_j} \rightarrow c_{12} m_1 \equiv 1 \pmod{m_2} \rightarrow c_{12} 11 \equiv 1 \pmod{17} \rightarrow c_{12} = 14;$

$v_1 \leftarrow b_1 \bmod m_1 = 4;$

$v_2 \leftarrow (b_2 - v_1)\, c_{12} \bmod m_2 = 0;$

$L = v_2 m_1 + v_1 = 0 \cdot 11 + 4 = 4.$

Для остатков $(b_1, b_2) = (4, 9)$:

$v_1 \leftarrow b_1 \bmod m_1 = 4;$

$v_2 \leftarrow (b_2 - v_1)\, c_{12} \bmod m_2 = 5 \cdot 14 \bmod 17 = 2;$

$L = v_2 m_1 + v_1 = 2 \cdot 11 + 4 = 26.$

К сожалению вычисления нельзя выполнять параллельно, поскольку прежде, чем вычислить $v_j$, необходимо определить $v_{j-1}$.

Рассмотрим геометрический способ перехода от модульного представления к позиционному [13].

### 1.3.3. ГЕОМЕТРИЧЕСКИЙ СПОСОБ ПЕРЕВОДА ЧИСЕЛ ОТ МОДУЛЬНОГО ПРЕДСТАВЛЕНИЯ К ПОЗИЦИОННОМУ

Если выбрать небольшие модули, то эффективно использовать табличные методы реализации. В ряде случаев бывает удобно внести результат операции в таблицу, строки и столбцы в которой определяются операндами. В таких случаях операция поиска по таблице эффективнее непосредственно арифметических операций с остатками. Однако хранить полную таблицу не обязательно.

Рассмотрим восстановление чисел на основе подсчета количества витков. Это позволит значительно сократить объем табличных данных.

Можно представить геометрическое истолкование одного сравнения. Пусть $m$ для определенности равно 7. Представим себе, что числовая прямая целых чисел накручена на окружность длиной 7 так, что числа 7, 14, 21 и так далее попадают на нулевую точку; числа 1, 8, 15, …, а также –6, –13, –20 совмещаются со следующей целочисленной точкой окружности и т. д. [14] (рис. 1.3.2).

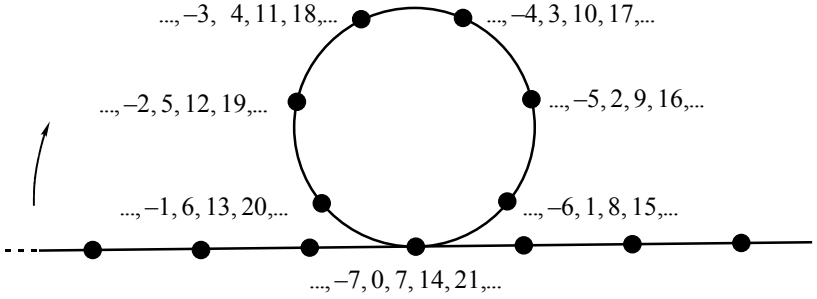



*Рис. 1.3.2.* Геометрическое представление сравнения $a \equiv b \pmod 7$

При одном сравнении числовая прямая накручивается на окружность. Рассмотрим, как выглядит геометрическое представление решения системы сравнений при двух модулях [15].

### РЕШЕНИЕ СИСТЕМЫ СРАВНЕНИЙ ПРИ ДВУХ МОДУЛЯХ

Рассмотрим случай ***для двух взаимно простых модулей***. Пусть $m_1 = 11$, $m_2 = 17$. Решение можно представить в виде (1.3.6):

$$L \equiv 17 \cdot 2\, b_1 + 11 \cdot 14\, b_2 \pmod{11 \cdot 17} \equiv 34 b_1 + 154 b_2 \pmod{187}.$$

На рис. 1.3.3 показана таблица всех возможных решений системы из двух сравнений с модулями, равными $m_1 = 11$ и $m_2 = 17$. Первая строка и первый столбец – значения остатков, на пересечении – результаты решения системы сравнений.

На рис. 1.3.3 стрелками показано последовательное возрастание чисел в таблице решений. Видно, что числа возрастают от нуля до $m_1 - 1$ последовательно по главной диагонали, а затем по диагоналям, показанным на рисунке стрелками.

| | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b1 | 0 | 14 | 11 | 8 | 5 | 2 | 16 | 13 | 10 | 7 | 4 | 1 | 15 | 12 | 9 | 6 | 3 | |
| 0 | 3 | 0 | 154 | 121 | 88 | 55 | 22 | 176 | 143 | 110 | 77 | 44 | 11 | 165 | 132 | 99 | 66 | 33 |
| 1 | 6 | 34 | 1 | 155 | 122 | 89 | 56 | 23 | 177 | 144 | 111 | 78 | 45 | 12 | 166 | 133 | 100 | 67 |
| 2 | 9 | 68 | 35 | 2 | 156 | 123 | 90 | 57 | 24 | 178 | 145 | 112 | 79 | 46 | 13 | 167 | 134 | 101 |
| 3 | 12 | 102 | 69 | 36 | 3 | 157 | 124 | 91 | 58 | 25 | 179 | 146 | 113 | 80 | 47 | 14 | 168 | 135 |
| 4 | 15 | 136 | 103 | 70 | 37 | 4 | 158 | 125 | 92 | 59 | 26 | 180 | 147 | 114 | 81 | 48 | 15 | 169 |
| 5 | 1 | 170 | 137 | 104 | 71 | 38 | 5 | 159 | 126 | 93 | 60 | 27 | 181 | 148 | 115 | 82 | 49 | 16 |
| 6 | 4 | 17 | 171 | 138 | 105 | 72 | 39 | 6 | 160 | 127 | 94 | 61 | 28 | 182 | 149 | 116 | 83 | 50 |
| 7 | 7 | 51 | 18 | 172 | 139 | 106 | 73 | 40 | 7 | 161 | 128 | 95 | 62 | 29 | 183 | 150 | 117 | 84 |
| 8 | 10 | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 | 8 | 162 | 129 | 96 | 63 | 30 | 184 | 151 | 118 |
| 9 | 13 | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 | 9 | 163 | 130 | 97 | 64 | 31 | 185 | 152 |
| 10 | | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 | 10 | 164 | 131 | 98 | 65 | 32 | 186 |

Число витков

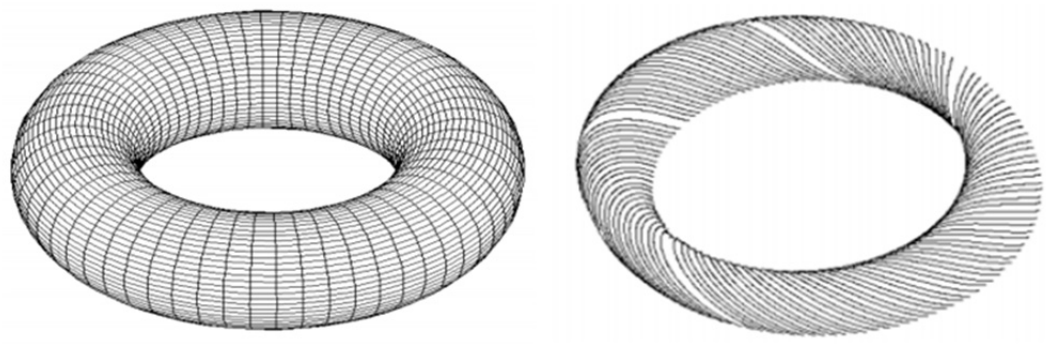
Число витков

*Рис. 1.3.3.* Последовательное изменение чисел в таблице решений системы сравнений с $m_1 = 11$ и $m_2 = 17$ (значения в столбцах $b_1$ меняются от нуля до 10, в строках $b_2$ – от нуля до 16)

Если непрерывно соединить продолжения диагоналей, то при последовательном возрастании чисел можно заметить, что после склейки верхней и нижней горизонтальной строки и левого и правого столбца таблицы решений образуется тор (рис. 1.3.4).

Для удобства мы можем вычислить начальные значения на витках тора. Номера витков показаны во второй строке и втором столбце таблицы на рис. 1.3.3. Размер витка равен минимальному из значений модулей (для нашего случая $m_1 = 11$).

Разрежем тор вдоль одного из меридианов. Тогда он превращается в круговой цилиндр с двумя краевыми окружностями [15]. Закрепим неподвижно одну окружность и станем закручивать цилиндр вокруг себя так, чтобы вторая окружность сделала $k$ оборотов. Всякая прямолинейная образующая цилиндра при этом обратится в винтовую линию, обходящую ось цилиндра $k$ раз. Если снова склеить оба края, то получим топологическое отображение тора на самого себя. При таком отображении тора его параллели превратились в винтообразные кривые, и наоборот (рис. 1.3.4, *б*). Решения системы сравнений последовательно будут возрастать по спирали на поверхности тора от нуля до $m_1 m_2$.



*а* *б*

*Рис. 1.3.4.* Тор, образующийся в результате склейки таблицы решений (*а*); последовательное возрастание чисел по диагоналям в таблице решений при отображении на поверхность двумерного тора (*б*)

Задачу определения числа по его остаткам можно свести к задаче определения начального значения витка тора и количества точек на этом витке. Этот подход определения числа $L$ по набору остатков мы назвали *геометрическим*.

Номер витков может быть найден следующим образом. Поскольку для двух остатков

$$L \equiv M_1 N_1 b_1 + M_2 N_2 b_2 (\mathrm{mod}\, m_1 m_2) \tag{1.3.10}$$

в нулевой строке $(b_1 = 0)$, то эти значения можно определить с помощью следующих выражений:

$$n[i] = M_2 N_2\ i \,(\mathrm{mod}\ m_1 m_2) = \frac{m_1 m_2}{m_2} N_2\, i \,(\mathrm{mod}\ m_1 m_2) = \\ = m_1 N_2\ i \,(\mathrm{mod}\ m_1 m_2) = N_2\, i \,(\mathrm{mod}\ m_2), \tag{1.3.11}$$

где $i = 0, \ldots, m_2 - 1$.

Тогда сразу же можно определить само число $L$ для случая $i = b_2 - b_1 \geq 0$ :

$$L = n[i] m_1 + b_1 . \tag{1.3.12}$$

К сожалению, по этой двумерной таблице трудно проследить полные витки. Только нулевой виток полностью помещается на одной диагонали, другие витки начинаются в нулевой строке и продолжаются на диагоналях, начинающихся с соответствующих значений на столбцах. Поэтому придется хранить два массива количества витков, по строкам и по столбцам (на рис. 1.3.3 – вторая строка и второй столбец). Чтобы избежать этого, представим таблицу в виде, показанном на рис. 1.3.5.

Когда значения остатков $b_2 - b_1 < 0,$ в таблице решений они соответствуют числам ниже нулевой диагонали. Такие числа симметрично отображаются, как показано на рис. 1.3.5.

| | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | ← Число витков |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b1 | 0 | 14 | 11 | 8 | 5 | 2 | 16 | 13 | 10 | 7 | 4 | 1 | 15 | 12 | 9 | 6 | 3 | | |
| 0 | 3 | 0 | 154 | 121 | 88 | 55 | 22 | 176 | 143 | 110 | 77 | 44 | 11 | 165 | 132 | 99 | 66 | 33 | |
| 1 | 6 | 34 | 1 | 155 | 122 | 89 | 56 | 23 | 177 | 144 | 111 | 78 | 45 | 12 | 166 | 133 | 100 | 67 | 34 |
| 2 | 9 | 68 | 35 | 2 | 156 | 123 | 90 | 57 | 24 | 178 | 145 | 112 | 79 | 46 | 13 | 167 | 134 | 101 | 68 | 35 |
| 3 | 12 | 102 | 69 | 36 | 3 | 157 | 124 | 91 | 58 | 25 | 179 | 146 | 113 | 80 | 47 | 14 | 168 | 135 | 102 | 69 | 36 |
| 4 | 15 | 136 | 103 | 70 | 37 | 4 | 158 | 125 | 92 | 59 | 26 | 180 | 147 | 114 | 81 | 48 | 15 | 169 | 136 | 103 | 70 | 37 |
| 5 | 1 | 170 | 137 | 104 | 71 | 38 | 5 | 159 | 126 | 93 | 60 | 27 | 181 | 148 | 115 | 82 | 49 | 16 | 170 | 137 | 104 | 71 | 38 |
| 6 | 4 | 17 | 171 | 138 | 105 | 72 | 39 | 6 | 160 | 127 | 94 | 61 | 28 | 182 | 149 | 116 | 83 | 50 | 17 | 171 | 138 | 105 | 72 | 39 |
| 7 | 7 | 51 | 18 | 172 | 139 | 106 | 73 | 40 | 7 | 161 | 128 | 95 | 62 | 29 | 183 | 150 | 117 | 84 | 51 | 18 | 172 | 139 | 106 | 73 | 40 |
| 8 | 10 | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 | 8 | 162 | 129 | 96 | 63 | 30 | 184 | 151 | 118 | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 |
| 9 | 13 | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 | 9 | 163 | 130 | 97 | 64 | 31 | 185 | 152 | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 |
| 10 | | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 | 10 | 164 | 131 | 98 | 65 | 32 | 186 | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 |

↑ Число витков

*Рис. 1.3.5.* Определение количества витков на поверхности тора

Выражение (1.3.12) в этом случае будет выглядеть как

$$L = \left[N_2 (b_2 - b_1) \bmod m_2\right] m_1 + b_1, \qquad \text{если } b_2 - b_1 \geq 0 ; \tag{1.3.13а}$$

$$L = \left[N_2 (b_2 - b_1 + m_2) \bmod m_2\right] m_1 + b_1, \quad \text{если } b_2 - b_1 < 0 . \tag{1.3.13б}$$

Алгоритм нахождения $L$ для двух модулей можно записать в следующем виде.

1. Формируем массив витков $n\,[i]$ (вторая строка в таблице на рис. 1.3.2) и вычисляем $a[i]$:

$$a\,[i] = n\,[i]\,m_1 = \left[N_2 i \pmod{m_2}\right] m_1 \ , \quad i = 0, \ldots, m_2 - 1 \,. \tag{1.3.14}$$

2. Находим индекс в массиве:

$$i = b_2 - b_1 \,, \qquad \text{если} \quad b_2 - b_1 \geq 0 \,; \tag{1.3.15а}$$

$$i = b_2 - b_1 + m_2 \,, \quad \text{если} \quad b_2 - b_1 < 0 \,. \tag{1.3.15б}$$

3. Находим результирующее значение в таблице:

$$L = a\,[i] + b_1 \,. \tag{1.3.16}$$

**Пример**

Найти решение для остатков $(b_1, b_2) = (4, 4)$, $(b_1, b_2) = (4, 9)$ и $(b_1, b_2) = (9, 4)$ в системе из двух сравнений с модулями $m_1 = 11$ и $m_2 = 17$.

Вычислим

$(b_1, b_2) = (4, 4),$
$i = 4 - 4 = 0,$
$L = n\,[0] \cdot 11 + 4 = 0 + 4 = 4;$

$(b_1, b_2) = (4, 9),$
$i = 9 - 4 = 5,$
$L = n\,[5] \cdot 11 + 4 = 2 \cdot 11 + 4 = 26;$

$(b_1, b_2) = (9, 4),$
$i = 4 - 9 = -5 < 0 \quad \Rightarrow \quad i = -5 + 17 = 12,$
$L = n\,[12] \cdot 11 + 9 = 15 \cdot 11 + 4 = 165 + 9 = 174 \,.$

В алгоритме наиболее трудоемким является первый этап, но он выполняется только один раз перед началом измерений. Значения элементов массива можно запомнить в массиве из $m_2$ длинных чисел.

В процессе вычислений выполняются этапы 2 и 3 алгоритма, которые реализуются с помощью основных операций компьютера (вычитания, сложения и сравнения с нулем), поэтому они выполняются быстро.

Недостатком алгоритма является то, что операции на этапах 2 и 3 выполняются с помощью длинной арифметики. К достоинствам алгоритма можно отнести отсутствие длинных операций умножения.

### РЕШЕНИЕ СИСТЕМЫ СРАВНЕНИЙ ПРИ ТРЕХ МОДУЛЯХ

Рассмотрим случай ***для трех взаимно простых модулей*** [16].

Дана система сравнений с тремя модулями: $m_1 = 3$, $m_2 = 4$, $m_3 = 5$. При этом наборе модулей решение может быть записано в следующем виде:

$$L \equiv 40b_1 + 45b_2 + 36b_3 \pmod{60}. \qquad (1.3.17)$$

Как видно из рис. 1.3.6, значения чисел последовательно увеличиваются по главным диагоналям. Однако проследить поведение диагоналей на трехмерном торе достаточно сложно.

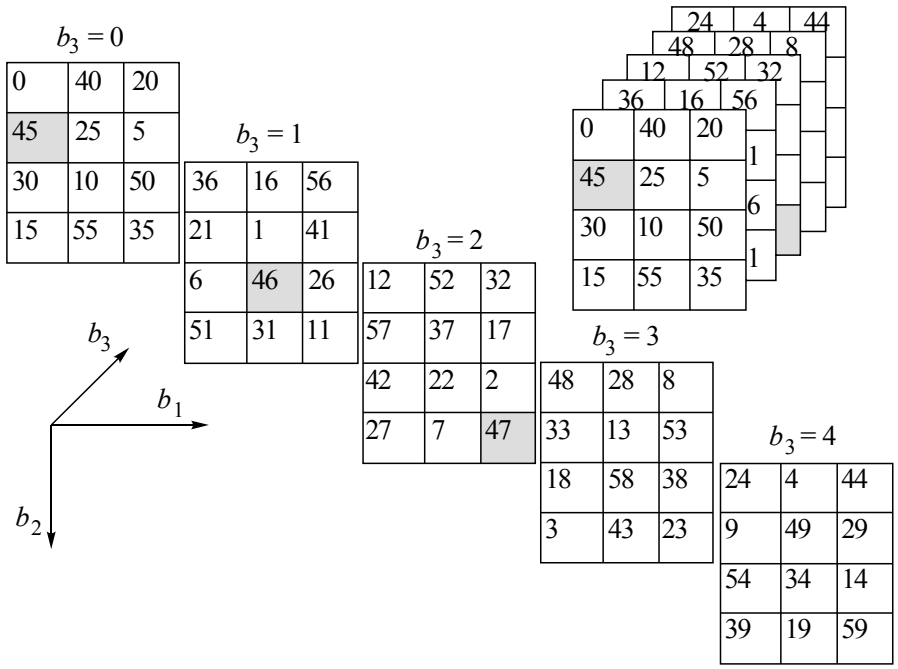



*Рис. 1.3.6.* Таблица решений с тремя модулями $m_1 = 3$, $m_2 = 4$, $m_3 = 5$

Необходимо построить алгоритм, аналогичный решению сравнений при двух модулях, который сводится к выбору из одномерного массива.

Для модулей $m_1 = 11$, $m_2 = 13$, $m_3 = 17$ имеем

$$M_1 = 221,\ M_2 = 187,\ M_3 = 143,$$

$$N_1 = 1,\ N_2 = 8,\ N_3 = 5.$$

Решение системы сравнений будет иметь вид:

$$L \equiv M_1 N_1 b_1 + M_2 N_2 b_2 + M_3 N_3 b_3 (\mathrm{mod}\, m_1 m_2 m_3)\,; \qquad (1.3.18\text{а})$$

$$L \equiv 221 \cdot 1\, b_1 + 187 \cdot 8\, b_2 + 143 \cdot 5\, b_3 (\mathrm{mod} 11 \cdot 13 \cdot 17)\,; \qquad (1.3.18\text{б})$$

$$L \equiv 221\, b_1 + 1496\, b_2 + 715\, b_3 (\mathrm{mod}\, 2431)\,. \qquad (1.3.18\text{в})$$

Таблица решений для случая трех модулей будет выглядеть следующим образом (рис. 1.3.7).

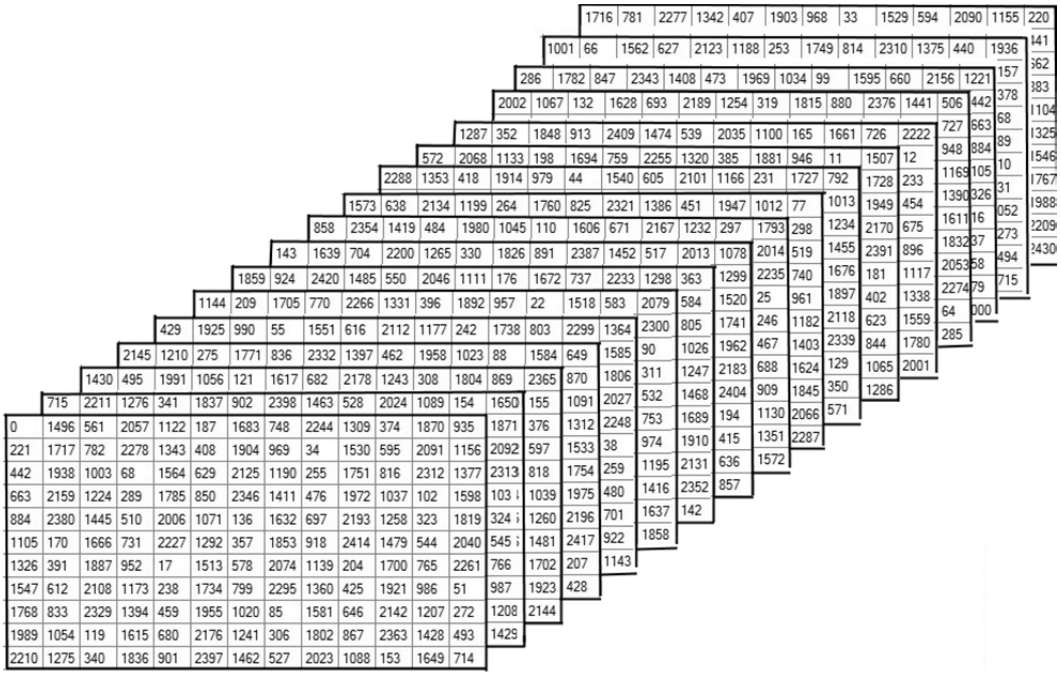


*Рис. 1.3.7.* Таблица решений с тремя модулями: $m_1 = 11$, $m_2 = 13$, $m_3 = 17$

Решение этой трехмерной задачи можно свести к двумерному случаю. Для этого представим решения для $L$ в виде двумерной таблицы.

Представим число в модульном представлении для двух модулей, $m_1$ и $m_2^3$, где $m_2^3 = m_2 m_3$. Тогда выражение (1.3.18а) примет вид

$$L \equiv M_1 N_1 b_1 + (M_2 N_2 + M_3 N_3) b_2^3 \ (\mathrm{mod}\, m_1 m_2 m_3). \tag{1.3.19}$$

Пусть $M_2 N_2 + M_3 N_3 = M_2^2 N_2^2$, тогда

$$L \equiv M_1 N_1 b_1 + M_2^2 N_2^2 b_2^3 \ (\mathrm{mod}\, m_1 m_2 m_3), \tag{1.3.20}$$

или для модулей $m_1$, $m_2^3 = m_2 m_3$ решение можно представить в двумерном виде:

$$L \equiv M_1 N_1 b_1 + M_2^2 N_2^2 b_2^3 \ (\mathrm{mod}\, m_1 m_2^3). \tag{1.3.21}$$

Для модулей $m_1 = 11$, $m_2^3 = 13 \cdot 17 = 221$ получим $M_1 = 221$, $M_2^2 = 11$, $N_1 = 1$, $N_2^2 = 201$:

$$\begin{aligned} L &\equiv 221 \cdot 1 \ b_1 + 11 \cdot 201 \, b_2^3 \ (\mathrm{mod}\, 11 \cdot 221) \equiv \\ &\equiv 221 b_1 + 2211 b_2^3 \ (\mathrm{mod}\ 2431). \end{aligned} \tag{1.3.22}$$

Видно, что коэффициенты сравнения (1.3.21) $M_2^2 N_2^2$ при $b_2^3$ можно найти простым суммированием коэффициентов для трех модулей, как показано в (1.3.19) – $M_2^2 N_2^2 = M_2 N_2 + M_3 N_3$, или прямо из китайской теоремы об остатках:

$$M_2^2 = \frac{m_1 m_2^3}{m_2^3} = m_1, \quad M_2^2 N_2^2 \equiv 1 \left(\mathrm{mod}\ m_2^3\right).$$

Заметим, что и в этом случае мы тоже будем работать в системе остаточных классов по модулю $m_1 m_2 m_3 = 2431$. Значения $b_1$ меняются от нуля до 11, значения $b_2^3$ – от нуля до 220. Начальная часть таблицы решений (1.3.22) показана на рис. 1.3.8.

| $b_2^3$ | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| b1 | b3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 0 | | 0 | 2211 | 1991 | 1771 | 1551 | 1331 | 1111 | 891 | 671 | 451 | 231 | 11 | 2222 | 2002 | 1782 | 1562 | 1342 | 1122 | 902 | 682 | 462 | 242 | 22 | 2233 | 2013 | 1793 | 1573 | 1353 | 1133 | 913 | 693 | 473 | 253 |
| 1 | | 221 | 1 | 2212 | 1992 | 1772 | 1552 | 1332 | 1112 | 892 | 672 | 452 | 232 | 12 | 2223 | 2003 | 1783 | 1563 | 1343 | 1123 | 903 | 683 | 463 | 243 | 23 | 2234 | 2014 | 1794 | 1574 | 1354 | 1134 | 914 | 694 | 474 |
| 2 | | 442 | 222 | 2 | 2213 | 1993 | 1773 | 1553 | 1333 | 1113 | 893 | 673 | 453 | 233 | 13 | 2224 | 2004 | 1784 | 1564 | 1344 | 1124 | 904 | 684 | 464 | 244 | 24 | 2235 | 2015 | 1795 | 1575 | 1355 | 1135 | 915 | 695 |
| 3 | | 663 | 443 | 223 | 3 | 2214 | 1994 | 1774 | 1554 | 1334 | 1114 | 894 | 674 | 454 | 234 | 14 | 2225 | 2005 | 1785 | 1565 | 1345 | 1125 | 905 | 685 | 465 | 245 | 25 | 2236 | 2016 | 1796 | 1576 | 1356 | 1136 | 916 |
| 4 | | 884 | 664 | 444 | 224 | 4 | 2215 | 1995 | 1775 | 1555 | 1335 | 1115 | 895 | 675 | 455 | 235 | 15 | 2226 | 2006 | 1786 | 1566 | 1346 | 1126 | 906 | 686 | 466 | 246 | 26 | 2237 | 2017 | 1797 | 1577 | 1357 | 1137 |
| 5 | | 1105 | 885 | 665 | 445 | 225 | 5 | 2216 | 1996 | 1776 | 1556 | 1336 | 1116 | 896 | 676 | 456 | 236 | 16 | 2227 | 2007 | 1787 | 1567 | 1347 | 1127 | 907 | 687 | 467 | 247 | 27 | 2238 | 2018 | 1798 | 1578 | 1358 |
| 6 | | 1326 | 1106 | 886 | 666 | 446 | 226 | 6 | 2217 | 1997 | 1777 | 1557 | 1337 | 1117 | 897 | 677 | 457 | 237 | 17 | 2228 | 2008 | 1788 | 1568 | 1348 | 1128 | 908 | 688 | 468 | 248 | 28 | 2239 | 2019 | 1799 | 1579 |
| 7 | | 1547 | 1327 | 1107 | 887 | 667 | 447 | 227 | 7 | 2218 | 1998 | 1778 | 1558 | 1338 | 1118 | 898 | 678 | 458 | 238 | 18 | 2229 | 2009 | 1789 | 1569 | 1349 | 1129 | 909 | 689 | 469 | 249 | 29 | 2240 | 2020 | 1800 |
| 8 | | 1768 | 1548 | 1328 | 1108 | 888 | 668 | 448 | 228 | 8 | 2219 | 1999 | 1779 | 1559 | 1339 | 1119 | 899 | 679 | 459 | 239 | 19 | 2230 | 2010 | 1790 | 1570 | 1350 | 1130 | 910 | 690 | 470 | 250 | 30 | 2241 | 2021 |
| 9 | | 1989 | 1769 | 1549 | 1329 | 1109 | 889 | 669 | 449 | 229 | 9 | 2220 | 2000 | 1780 | 1560 | 1340 | 1120 | 900 | 680 | 460 | 240 | 20 | 2231 | 2011 | 1791 | 1571 | 1351 | 1131 | 911 | 691 | 471 | 251 | 31 | 2242 |
| 10 | | 2210 | 1990 | 1770 | 1550 | 1330 | 1110 | 890 | 670 | 450 | 230 | 10 | 2221 | 2001 | 1781 | 1561 | 1341 | 1121 | 901 | 681 | 461 | 241 | 21 | 2232 | 2012 | 1792 | 1572 | 1352 | 1132 | 912 | 692 | 472 | 252 | 32 |

*Рис. 1.3.8.* Начальная часть таблицы решений (1.3.22)

В первой строке показаны значения $b_2^3$, которые меняются от нуля до 220, во второй – значения $b_2$ (от нуля до 10), в третьей – значения $b_3$ (от нуля до 16). В таблице прослеживаются витки, необходимо определить их начальные значения.

Определим зависимость $b_2^3$ (верхняя строка таблицы) от $b_2$ и $b_3$. Ее можно записать в виде следующего выражения:

$$b_2^3 \equiv M_2^{23} N_2^{23}\, b_2 + M_3^{23} N_3^{23}\, b_3 \,(\mathrm{mod}\, m_2 m_3). \qquad (1.3.23)$$

Найти $M_2^{23}$, $N_2^{23}$, $M_3^{23}$, $N_3^{23}$ можно по обычным правилам китайской теоремы об остатках. Для $m_2 = 13$, $m_3 = 17$ получим $M_2^{23} = 17$, $N_2^{23} = 10$, $M_3^{23} = 13$, $N_3^{23} = 4$. В результате

$$b_3^2 \equiv 17 \cdot 10 b_2 + 13 \cdot 4 b_3 (\mathrm{mod}\, 13 \cdot 17) = 170 b_2 + 52 b_3 (\mathrm{mod}\, 221). \qquad (1.3.24)$$

Эти решения показаны в таблице на рис. 1.3.9.

| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | | | | | | | | |
| 0 | | 0 | 52 | 104 | 156 | 208 | 39 | 91 | 143 | 195 | 26 | 78 | 130 | 182 | 13 | 65 | 117 | 169 |
| 1 | | 170 | 1 | 53 | 105 | 157 | 209 | 40 | 92 | 144 | 196 | 27 | 79 | 131 | 183 | 14 | 66 | 118 |
| 2 | | 119 | 171 | 2 | 54 | 106 | 158 | 210 | 41 | 93 | 145 | 197 | 28 | 80 | 132 | 184 | 15 | 67 |
| 3 | | 68 | 120 | 172 | 3 | 55 | 107 | 159 | 211 | 42 | 94 | 146 | 198 | 29 | 81 | 133 | 185 | 16 |
| 4 | | 17 | 69 | 121 | 173 | 4 | 56 | 108 | 160 | 212 | 43 | 95 | 147 | 199 | 30 | 82 | 134 | 186 |
| 5 | | 187 | 18 | 70 | 122 | 174 | 5 | 57 | 109 | 161 | 213 | 44 | 96 | 148 | 200 | 31 | 83 | 135 |
| 6 | | 136 | 188 | 19 | 71 | 123 | 175 | 6 | 58 | 110 | 162 | 214 | 45 | 97 | 149 | 201 | 32 | 84 |
| 7 | | 85 | 137 | 189 | 20 | 72 | 124 | 176 | 7 | 59 | 111 | 163 | 215 | 46 | 98 | 150 | 202 | 33 |
| 8 | | 34 | 86 | 138 | 190 | 21 | 73 | 125 | 177 | 8 | 60 | 112 | 164 | 216 | 47 | 99 | 151 | 203 |
| 9 | | 204 | 35 | 87 | 139 | 191 | 22 | 74 | 126 | 178 | 9 | 61 | 113 | 165 | 217 | 48 | 100 | 152 |
| 10 | | 153 | 205 | 36 | 88 | 140 | 192 | 23 | 75 | 127 | 179 | 10 | 62 | 114 | 166 | 218 | 49 | 101 |
| 11 | | 102 | 154 | 206 | 37 | 89 | 141 | 193 | 24 | 76 | 128 | 180 | 11 | 63 | 115 | 167 | 219 | 50 |
| 12 | | 51 | 103 | 155 | 207 | 38 | 90 | 142 | 194 | 25 | 77 | 129 | 181 | 12 | 64 | 116 | 168 | 220 |

*Рис. 1.3.9.* Определение $b_3^2$ по $b_2$ (от нуля до 12) и $b_3$ (от нуля до 16)

Нахождение решения в таблице, показанной на рис. 1.3.9, можно свести к нахождению количества начальных витков (рис. 1.3.10).

| | b3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b2 | 0 | 4 | 8 | 12 | 16 | 3 | 7 | 11 | 15 | 2 | 6 | 10 | 14 | 1 | 5 | 9 | 13 | | ←Число витков | | | | | | | | | | |
| 0 | | 0 | 52 | 104 | 156 | 208 | 39 | 91 | 143 | 195 | 26 | 78 | 130 | 182 | 13 | 65 | 117 | 169 | | | | | | | | | | | |
| 1 | | 170 | 1 | 53 | 105 | 157 | 209 | 40 | 92 | 144 | 196 | 27 | 79 | 131 | 183 | 14 | 66 | 118 | 170 | | | | | | | | | | |
| 2 | | 119 | 171 | 2 | 54 | 106 | 158 | 210 | 41 | 93 | 145 | 197 | 28 | 80 | 132 | 184 | 15 | 67 | 119 | 171 | | | | | | | | | |
| 3 | | 68 | 120 | 172 | 3 | 55 | 107 | 159 | 211 | 42 | 94 | 146 | 198 | 29 | 81 | 133 | 185 | 16 | 68 | 120 | 172 | | | | | | | | |
| 4 | | 17 | 69 | 121 | 173 | 4 | 56 | 108 | 160 | 212 | 43 | 95 | 147 | 199 | 30 | 82 | 134 | 186 | 17 | 69 | 121 | 173 | | | | | | | |
| 5 | | 187 | 18 | 70 | 122 | 174 | 5 | 57 | 109 | 161 | 213 | 44 | 96 | 148 | 200 | 31 | 83 | 135 | 187 | 18 | 70 | 122 | 174 | | | | | | |
| 6 | | 136 | 188 | 19 | 71 | 123 | 175 | 6 | 58 | 110 | 162 | 214 | 45 | 97 | 149 | 201 | 32 | 84 | 136 | 188 | 19 | 71 | 123 | 175 | | | | | |
| 7 | | 85 | 137 | 189 | 20 | 72 | 124 | 176 | 7 | 59 | 111 | 163 | 215 | 46 | 98 | 150 | 202 | 33 | 85 | 137 | 189 | 20 | 72 | 124 | 176 | | | | |
| 8 | | 34 | 86 | 138 | 190 | 21 | 73 | 125 | 177 | 8 | 60 | 112 | 164 | 216 | 47 | 99 | 151 | 203 | 34 | 86 | 138 | 190 | 21 | 73 | 125 | 177 | | | |
| 9 | | 204 | 35 | 87 | 139 | 191 | 22 | 74 | 126 | 178 | 9 | 61 | 113 | 165 | 217 | 48 | 100 | 152 | 204 | 35 | 87 | 139 | 191 | 22 | 74 | 126 | 178 | | |
| 10 | | 153 | 205 | 36 | 88 | 140 | 192 | 23 | 75 | 127 | 179 | 10 | 62 | 114 | 166 | 218 | 49 | 101 | 153 | 205 | 36 | 88 | 140 | 192 | 23 | 75 | 127 | 179 | |
| 11 | | 102 | 154 | 206 | 37 | 89 | 141 | 193 | 24 | 76 | 128 | 180 | 11 | 63 | 115 | 167 | 219 | 50 | 102 | 154 | 206 | 37 | 89 | 141 | 193 | 24 | 76 | 128 | 180 |
| 12 | | 51 | 103 | 155 | 207 | 38 | 90 | 142 | 194 | 25 | 77 | 129 | 181 | 12 | 64 | 116 | 168 | 220 | 51 | 103 | 155 | 207 | 38 | 90 | 142 | 194 | 25 | 77 | 129 | 181 |

*Рис. 1.3.10.* Решение системы сравнений $b_3^2 \equiv 170b_2 + 52b_3 \pmod{221}$
с количеством витков $\left(\text{вторая строка } n_1(i) = N_3^{23} i \bmod(m_3),\ N_2^3 = 4,\ i = 0, ..., 16\right)$

**Пример 1**

Для модулей $m_1 = 11$, $m_2 = 13$, $m_3 = 17$ таблица для нахождения $b_2^3$ по модулям $m_2$ и $m_3$ будет иметь вид, показанный на рис. 1.3.10. Количество витков определяется, как показано во второй строке на рис. 1.3.10. В нашем случае $N_3^{23} = 4$ и

$$n_1(i) = N_3^{23} i \bmod(m_3), \quad i = 0, \ldots, m_3 - 1. \tag{1.3.25}$$

Рассмотрим массив витков (рис. 1.3.8). Количество начальных витков на поверхности тора будет определяться по формуле (1.3.22). В нашем случае это будет $N_2^2$:

$$n_2(i) = N_2^2 i \bmod(m_2 m_3), \text{ где } i = b_2^3. \tag{1.3.26}$$

Эти числа показаны во второй строке на рис. 1.3.11.

Определив $b_3^2$, можно найти и решение системы по трем остаткам.

| | $b_2^3$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b1 | 0 | 201 | 181 | 161 | 141 | 121 | 101 | 81 | 61 | 41 | 21 | 1 | 202 | 182 | 162 | 142 | 122 | 102 | 82 | 62 | 42 | 22 | 2 | 203 | 183 | 163 | 143 | 123 | 103 | 83 | 63 | 43 | 23 | 3 |
| 0 | | 0 | 2211 | 1991 | 1771 | 1551 | 1331 | 1111 | 891 | 671 | 451 | 231 | 11 | 2222 | 2002 | 1782 | 1562 | 1342 | 1122 | 902 | 682 | 462 | 242 | 22 | 2233 | 2013 | 1793 | 1573 | 1353 | 1133 | 913 | 693 | 473 | 253 |
| 1 | | 221 | 1 | 2212 | 1992 | 1772 | 1552 | 1332 | 1112 | 892 | 672 | 452 | 232 | 12 | 2223 | 2003 | 1783 | 1563 | 1343 | 1123 | 903 | 683 | 463 | 243 | 23 | 2234 | 2014 | 1794 | 1574 | 1354 | 1134 | 914 | 694 | 474 |
| 2 | | 442 | 222 | 2 | 2213 | 1993 | 1773 | 1553 | 1333 | 1113 | 893 | 673 | 453 | 233 | 13 | 2224 | 2004 | 1784 | 1564 | 1344 | 1124 | 904 | 684 | 464 | 244 | 24 | 2235 | 2015 | 1795 | 1575 | 1355 | 1135 | 915 | 695 |
| 3 | | 663 | 443 | 223 | 3 | 2214 | 1994 | 1774 | 1554 | 1334 | 1114 | 894 | 674 | 454 | 234 | 14 | 2225 | 2005 | 1785 | 1565 | 1345 | 1125 | 905 | 685 | 465 | 245 | 25 | 2236 | 2016 | 1796 | 1576 | 1356 | 1136 | 916 |
| 4 | | 884 | 664 | 444 | 224 | 4 | 2215 | 1995 | 1775 | 1555 | 1335 | 1115 | 895 | 675 | 455 | 235 | 15 | 2226 | 2006 | 1786 | 1566 | 1346 | 1126 | 906 | 686 | 466 | 246 | 26 | 2237 | 2017 | 1797 | 1577 | 1357 | 1137 |
| 5 | | 1105 | 885 | 665 | 445 | 225 | 5 | 2216 | 1996 | 1776 | 1556 | 1336 | 1116 | 896 | 676 | 456 | 236 | 16 | 2227 | 2007 | 1787 | 1567 | 1347 | 1127 | 907 | 687 | 467 | 247 | 27 | 2238 | 2018 | 1798 | 1578 | 1358 |
| 6 | | 1326 | 1106 | 886 | 666 | 446 | 226 | 6 | 2217 | 1997 | 1777 | 1557 | 1337 | 1117 | 897 | 677 | 457 | 237 | 17 | 2228 | 2008 | 1788 | 1568 | 1348 | 1128 | 908 | 688 | 468 | 248 | 28 | 2239 | 2019 | 1799 | 1579 |
| 7 | | 1547 | 1327 | 1107 | 887 | 667 | 447 | 227 | 7 | 2218 | 1998 | 1778 | 1558 | 1338 | 1118 | 898 | 678 | 458 | 238 | 18 | 2229 | 2009 | 1789 | 1569 | 1349 | 1129 | 909 | 689 | 469 | 249 | 29 | 2240 | 2020 | 1800 |
| 8 | | 1768 | 1548 | 1328 | 1108 | 888 | 668 | 448 | 228 | 8 | 2219 | 1999 | 1779 | 1559 | 1339 | 1119 | 899 | 679 | 459 | 239 | 19 | 2230 | 2010 | 1790 | 1570 | 1350 | 1130 | 910 | 690 | 470 | 250 | 30 | 2241 | 2021 |
| 9 | | 1989 | 1769 | 1549 | 1329 | 1109 | 889 | 669 | 449 | 229 | 9 | 2220 | 2000 | 1780 | 1560 | 1340 | 1120 | 900 | 680 | 460 | 240 | 20 | 2231 | 2011 | 1791 | 1571 | 1351 | 1131 | 911 | 691 | 471 | 251 | 31 | 2242 |
| 10 | | 2210 | 1990 | 1770 | 1550 | 1330 | 1110 | 890 | 670 | 450 | 230 | 10 | 2221 | 2001 | 1781 | 1561 | 1341 | 1121 | 901 | 681 | 461 | 241 | 21 | 2232 | 2012 | 1792 | 1572 | 1352 | 1132 | 912 | 692 | 472 | 252 | 32 |

*Рис. 1.3.11.* Начальная часть таблицы решений системы сравнений (количество витков – вторая строка, $N_1^{23} = 201$)

**Пример 2**

Пусть число в модульном представлении имеет вид $(1, 5, 14)$. По рис. 1.3.10 находим $b_2^3 = (5, 14) = 31$. Далее по рис. 1.3.11 находим исходное число $L = (1, 31) = 694$.

*Проверка*: $L \equiv 221b_1 + 1496b_2 + 715b_3 \pmod{2431} = 694$.

Результирующий алгоритм для определения числа $L$ по трем остаткам $(b_1, b_2, b_3)$ можно записать в следующем виде.

1. По значениям $(b_2, b_3)$ находим $b_2^3$.

1.1. Находим $M_3^{23} = \dfrac{m_2 m_3}{m_3} = m_2$ и $N_3^{23}$ из сравнения $M_3^{23} N_3^{23} \equiv 1 \pmod{m_3}$.

Обозначив $N_1 = N_3^{23}$, формируем массив

$$a_1(i) = \left[ N_1 i \bmod (m_3) \right] m_2, \quad i = 0, \ldots, m_3 - 1.$$

1.2. Находим индекс в массиве:

$$i = b_3 - b_2, \quad \text{если} \quad b_3 - b_2 \geq 0;$$

$$i = b_3 - b_2 + m_3, \quad \text{если} \quad b_3 - b_2 < 0.$$

1.3. Находим результирующее значение числа $b_2^3$:

$$b_2^3 = a_1[i] + b_2.$$

2. Находим результирующее число $L$.

2.1. Находим $N_2^2$ решением сравнения $M_2^2 N_2^2 \equiv 1 \pmod{m_2 m_3}$, где $M_2^2 = \dfrac{m_1 m_2 m_3}{m_2 m_3} = m_1$. Обозначив $N_2 = N_2^2$, получим

$$a_2(i) = \left[ N_2 i \bmod (m_2 m_3) \right] m_1, \quad i = 0, \ldots, m_2 m_3 - 1.$$

2.2. Находим индекс в массиве:

$$i = b_2^3 - b_1, \quad \text{если} \quad b_2^3 - b_1 \geq 0;$$

$$i = b_2^3 - b_1 + m_2, \quad \text{если} \quad b_2^3 - b_1 < 0.$$

2.3. Находим результирующее значение числа $L$:

$$L = a_2[i] + b_1 .$$

Для случая трех модулей перед началом работы с целью ускорения расчетов можно вычислить два массива начальных витков с длинными числами:

$$n_1[i] = N_2^{23} i \pmod{m_3}, \quad i = 0, 1, \ldots, m_3 - 1;$$

$$n_2[i] = N_2^2 i \bmod (m_2 m_3), \quad i = 0, 1, \ldots, (m_2 m_3 - 1)$$

и хранить их в памяти:

$$a_1[i] = n_1(i)\, m_2, \quad i = 0, 1, \ldots, m_3 - 1,$$

$$a_2[i] = n_2(i)\, m_1, \quad i = 0, 1, \ldots, (m_2 m_3 - 1).$$

Необходимо сначала определить $b_2^3$ в диапазоне от нуля до $m_2 m_3 - 1$, а затем $L$ в диапазоне от нуля до $m_1 m_2 m_3 - 1$.

Этот алгоритм несложно обобщить и на многомерный случай.

### РЕШЕНИЕ СИСТЕМЫ СРАВНЕНИЙ ПРИ *N* МОДУЛЯХ

Для случая *n* модулей необходимо хранить $n-1$ массив:

$$a_1[i] = \left[N_1 i \pmod{m_n}\right] m_{n-1}, \quad i = 0, \ldots, m_n - 1;$$

$$a_2[i] = \left[N_2 i \bmod (m_{n-1} m_n)\right] m_{n-2}, \quad i = 0, \ldots, (m_{n-1} m_n - 1);$$

$$a_{n-1}[i] = \left[N_{n-1} i \bmod (m_2 \ldots m_n)\right] m_1, \quad i = 0, \ldots, (m_2 m_{n-1} m_n - 1).$$

Вычисления полного числа $L$ по модульному представлению $(b_1, b_2, \ldots, b_n)$ можно проводить аналогично случаю для трех модулей последовательным вычислением промежуточных значений и поиском значений в двумерной таблице. Последовательно определяются коэффициенты $c_i$:

$c_{n-1} = a_1[i] + b_{n-1}$ по координатам $(b_{n-1}, b_n)$:

размер таблицы $b_{n-1}$ $(0 \ldots m_{n-1} - 1)$, $b_n$ $(0 \ldots m_n - 1)$;

$i = b_n - b_{n-1}$, если $b_n - b_{n-1} \geq 0$;

$i = b_n - b_{n-1} + m_2$, если $b_n - b_{n-1} < 0$;

$c_{n-2} = a_2[i] + b_{n-2}$ по координатам $(b_{n-2}, c_{n-1})$:

размер таблицы $b_{n-2}$ $(0 \ldots m_{n-2} - 1)$, $c_{n-1}$ $(0 \ldots (m_{n-1} m_n - 1))$;

$i = c_{n-1} - b_{n-2}$, если $c_{n-1} - b_{n-2} \geq 0$;

$i = c_{n-1} - b_{n-2} + m_2$, если $c_{n-1} - b_{n-2} < 0$;

...

$L = c_1 = a_{n-1}[i] + b_1$, по координатам $(b_1, c_2)$:

размер таблицы $b_1$ $(0 \ldots m_1 - 1)$, $c_2$ $(0 \ldots (m_2 \ldots m_{n-1} m_n - 1))$;

$i = c_2 - b_1$, если $c_2 - b_1 \geq 0$;

$i = c_2 - b_1 + m_2$, если $c_2 - b_1 < 0$.

Достоинством представленного алгоритма является то, что не требуется делать длинных операций умножения. Используются только операции сложения и сравнения остатков.

К сожалению, вычисления для определения полного числа $L$ нельзя выполнять параллельно, поскольку прежде, чем вычислить $c_1$, необходимо определить $c_2 \ldots c_{n-1}$.

Однако даже при небольших значениях модулей массивы $a_j[i]$ могут быть достаточно большими. Например, пусть модули имеют вид $m_j = 2^{k_j} - 1$: $m_1 = 2^{13} - 1$, $m_2 = 2^{15} - 1$, $m_3 = 2^{16} - 1$. Максимальная величина слова в этом случае составит $m_1 m_2 m_3 = 2^{13} \cdot 2^{15} \cdot 2^{16} = 2^{44} = 17\,592\,186\,044\,416$. В этом числе всего 44 двоичных разряда, или 14 десятичных знаков. Для хранения $a_j[i]$ необходимы два массива: размером $m_2$ и $m_2 m_3$. Максимальный размер для хранения наибольшего массива длинных чисел: $m_2 m_3 = 2^{15} \cdot 2^{16} = 2^{31} \sim 2$ Гб.

Если памяти для хранения массивов не хватает, то можно хранить в памяти только $n - 1$ коэффициентов $N_k$, а значения элементов массивов начальных витков определять с помощью операций умножения длинных чисел и взятия остатка:

$c_{n-1} = \left[ N_1 i \pmod{m_n} \right] m_{n-1} + b_{n-1}$ по координатам $(b_{n-1}, b_n)$:

$i = b_n - b_{n-1}$, если $b_n - b_{n-1} \geq 0$, иначе $i = b_n - b_{n-1} + m_n$;

$c_{n-2} = \left[ N_2 i \pmod{m_{n-1} m_n} \right] m_{n-2} + b_{n-2}$ по координатам $(b_{n-2}, c_{n-1})$:

$$i = c_{n-1} - b_{n-2}, \text{ если } c_{n-1} - b_{n-2} \geq 0, \text{ иначе } i = c_{n-1} - b_{n-2} + m_{n-1};$$

...

$L = c_1 = \left[ N_{n-1} i \bmod (m_2 \ldots m_n) \right] m_1 + b_1$ по координатам $(b_1, c_2)$:

$$i = c_2 - b_1, \text{ если } c_2 - b_1 \geq 0, \text{ иначе } i = c_2 - b_1 + m_2.$$

Однако операции умножения длинных чисел и определения остатка – достаточно медленные операции, поэтому восстановление длинных чисел по набору остатков желательно проводить только на заключительной стадии вычислений.

Быстрые алгоритмы умножения приведены, например, в [12].

Для примера приведем простой алгоритм с разбиением чисел на байты.

Если число разбить на байты и выполнять байтовые умножения таблично, то можно получить следующий алгоритм:

$a = 31$ – 0001 1111 – 1 15,

$b = 201$ – 1100 1001 – 12 9.

Выполняем умножение по байтам и складываем.

Для первого байта числа $b$ (1001):

1111 · 1001 = 15 · 9 = 135 = 1000 0111

0001 · 1001 = 1 · 9 = 9 = **1001** 0000 Старший байт

1 0001 0111

И для второго байта числа $b$ (1100)

1111 · 1100 = 15 · 12 = 180 = **1011 0100** 0000

0001 · 1100 = 1 · 12 = 12 = **1100** 0000 0000

= 1 0111 0100 0000

Получившиеся результаты складываем.

= 1 0111 0100 0000

1 0001 0111

1 1000 0101 0111

31 · 201 = 0001 1111 · 1100 1001 = 1 0100 0111 = 6231.

Операция определения остатка числа по модулю показана в предыдущем разделе.

## 1.4. СРАВНЕНИЕ ЧИСЕЛ И ОПРЕДЕЛЕНИЕ ПЕРЕПОЛНЕНИЯ ПРИ АРИФМЕТИЧЕСКИХ ОПЕРАЦИЯХ

При параллельном выполнении операций сложения, вычитания или умножения использование модульной арифметики дает значительное преимущество. Операции с разными модулями могут выполняться одновременно, что приводит к сокращению общего времени выполнения.

Существенным недостатком операций с модульными числами является сложность их сравнения, т. е. проверки, какое из чисел в модульном представлении больше или меньше, чем другое. Для сравнения чисел необходимо перевести их из модульного представления в позиционное, но это достаточно трудоемкая операция.

В этом разделе рассмотрены быстрые алгоритмы сравнения при сложении и умножении модульных чисел, основанные на геометрическом способе перехода от модульного представления к позиционному [17].

### 1.4.1. СРАВНЕНИЕ ЧИСЕЛ ПРИ ДВУХ МОДУЛЯХ

Для получения числа в позиционном представлении необходимо определить количество витков и количество элементов на этом витке (см. раздел 1.3.3). Количество операций при сравнении чисел в модульном представлении можно уменьшить, если последовательно определять количество витков по каждому из модулей. В большинстве случаев количество витков указывает, какое число больше или меньше другого, возникает переполнение или нет. Если количество витков для одного из чисел больше, чем для другого, то это число заведомо больше. Если количество витков одинаково, то определяем положение числа на расширенной диагонали. Это в ряде случаев позволяет значительно сократить количество вычислительных операций при сравнении.

Пусть необходимо сравнить два числа: $(b_1, b_2)$ и $(c_1, c_2)$.

Количество витков для $(b_1, b_2)$ при двух модулях $m_1$, $m_2$ равно

$$n(b_1, b_2) = N_2 (b_2 - b_1) \bmod m_2, \qquad \text{если } b_2 - b_1 \geq 0; \qquad (1.4.1)$$

$$n(b_1, b_2) = N_2 (b_2 - b_1 + m_2) \bmod m_2, \qquad \text{если } b_2 - b_1 < 0, \qquad (1.4.2)$$

где $N_2$ определяется из китайской теоремы об остатках (см. раздел 1.3.1).

Рассмотрим примеры для двух модулей: $m_1 = 11$ и $m_2 = 17$. В этом случае $N_2 = 14$.

Для быстрого определения количества витков можно воспользоваться следующей таблицей.

| b1 \ b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 14 | 11 | 8 | 5 | 2 | 16 | 13 | 10 | 7 | 4 | 1 | 15 | 12 | 9 | 6 | 3 | ← Число витков |
| 1 | 3 | 0 | 154 | 121 | 88 | 55 | 22 | 176 | 143 | 110 | 77 | 44 | 11 | 165 | 132 | 99 | 66 | 33 |
| 2 | 6 | 34 | 1 | 155 | 122 | 89 | 56 | 23 | 177 | 144 | 111 | 78 | 45 | 12 | 166 | 133 | 100 | 67 | 34 |
| 3 | 9 | 68 | 35 | 2 | 156 | 123 | 90 | 57 | 24 | 178 | 145 | 112 | 79 | 46 | 13 | 167 | 134 | 101 | 68 | 35 |
| 4 | 12 | 102 | 69 | 36 | 3 | 157 | 124 | 91 | 58 | 25 | 179 | 146 | 113 | 80 | 47 | 14 | 168 | 135 | 102 | 69 | 36 |
| 5 | 15 | 136 | 103 | 70 | 37 | 4 | 158 | 125 | 92 | 59 | 26 | 180 | 147 | 114 | 81 | 48 | 15 | 169 | 136 | 103 | 70 | 37 |
| 6 | 1 | 170 | 137 | 104 | 71 | 38 | 5 | 159 | 126 | 93 | 60 | 27 | 181 | 148 | 115 | 82 | 49 | 16 | 170 | 137 | 104 | 71 | 38 |
| 7 | 4 | 17 | 171 | 138 | 105 | 72 | 39 | 6 | 160 | 127 | 94 | 61 | 28 | 182 | 149 | 116 | 83 | 50 | 17 | 171 | 138 | 105 | 72 | 39 |
| 8 | 7 | 51 | 18 | 172 | 139 | 106 | 73 | 40 | 7 | 161 | 128 | 95 | 62 | 29 | 183 | 150 | 117 | 84 | 51 | 18 | 172 | 139 | 106 | 73 | 40 |
| 9 | 10 | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 | 8 | 162 | 129 | 96 | 63 | 30 | 184 | 151 | 118 | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 |
| 10 | 13 | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 | 9 | 163 | 130 | 97 | 64 | 31 | 185 | 152 | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 |
| | ↑ Число витков | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 | 10 | 164 | 131 | 98 | 65 | 32 | 186 | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 |

Определение количества витков на поверхности тора (см. рис. 1.3.5)

**Пример 1**

Какое из двух чисел больше: $(2,12)$ или $(5,10)$?

Для $(2,12)$ $n(2,12)=4$;

для $(5,10)$ $n(5,10)=2$;

$(2,12)>(5,10)$.

*Проверка*: $L_1=(2,12)=46$ и $L_2=(5,10)=27$.

**Пример 2**

Какое из двух чисел больше: $(3,1)$ или $(3,2)$?

Для $(3,1)$ $n(3,1)=6$;

для $(3,2)$ $n(3,2)=3$;

$(3,1)>(3,2)$.

*Проверка*: $L_1=(3,1)=69$ и $L_2=(3,2)=36$.

Если количество витков одинаково, то определяем положение числа на расширенной диагонали (по величине $b_1$).

**Пример 3**

Какое их двух чисел больше: $(2,13)$ или $(8,2)$?

Для $(2,13)$ $n(2,13)=1$;

для $(8,2)$ $n(8,2)=1$.

Количество витков одинаково, поэтому смотрим значения $b_1$. Поскольку $8 > 2$, то $(8, 2) > (2, 13)$.

*Проверка*: $L_1 = (8, 2) = 19$ и $L_2 = (2, 13) = 13$.

## 1.4.2. СРАВНЕНИЕ ЧИСЕЛ ПРИ ТРЕХ МОДУЛЯХ

Число в модульном представлении имеет вид $(b_1, b_2, b_3)$ при трех значениях модулей $m_1, m_2, m_3$.

1. Последовательно находим количество витков для $(b_2, b_3)$. Если количество витков больше, то и число больше. Работа алгоритма заканчивается.

2. Если количество витков одинаковое, то находим $b_2^3$ и количество витков по $\left(b_1, b_2^3\right)$. Если количество витков больше, то и число больше. Работа алгоритма заканчивается.

3. Если количество витков одинаковое, то определяем положение числа на расширенной диагонали $b_1$.

Рассмотрим примеры для трех модулей: $m_1 = 11$, $m_2 = 13$, $m_3 = 17$.

Число в позиционном виде при этих модулях (см. раздел 1.3.3, (формула (1.3.18в)):

$$L \equiv 221\, b_1 + 1496\, b_2 + 715\, b_3 \,(\mathrm{mod}\, 2431).$$

Для быстрого определения количества витков в случае $(b_1, b_2, b_3)$ можно воспользоваться рис. 1.4.1.

Число витков

| b2 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b3 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| $b_2^3$ | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
| b1 | 0 | 201 | 181 | 161 | 141 | 121 | 101 | 81 | 61 | 41 | 21 | 1 | 202 | 182 | 162 | 142 | 122 | 102 | 82 | 62 | 42 | 22 | 2 | 203 | 183 | 163 | 143 | 123 | 103 | 83 | 63 | 43 | 23 | 3 |
| 0 | | 0 | 2211 | 1991 | 1771 | 1551 | 1331 | 1111 | 891 | 671 | 451 | 231 | 11 | 2222 | 2002 | 1782 | 1562 | 1342 | 1122 | 902 | 682 | 462 | 242 | 22 | 2233 | 2013 | 1793 | 1573 | 1353 | 1133 | 913 | 693 | 473 | 253 |
| 1 | | 221 | 1 | 2212 | 1992 | 1772 | 1552 | 1332 | 1112 | 892 | 672 | 452 | 232 | 12 | 2223 | 2003 | 1783 | 1563 | 1343 | 1123 | 903 | 683 | 463 | 243 | 23 | 2234 | 2014 | 1794 | 1574 | 1354 | 1134 | 914 | 694 | 474 |
| 2 | | 442 | 222 | 2 | 2213 | 1993 | 1773 | 1553 | 1333 | 1113 | 893 | 673 | 453 | 233 | 13 | 2224 | 2004 | 1784 | 1564 | 1344 | 1124 | 904 | 684 | 464 | 244 | 24 | 2235 | 2015 | 1795 | 1575 | 1355 | 1135 | 915 | 695 |
| 3 | | 663 | 443 | 223 | 3 | 2214 | 1994 | 1774 | 1554 | 1334 | 1114 | 894 | 674 | 454 | 234 | 14 | 2225 | 2005 | 1785 | 1565 | 1345 | 1125 | 905 | 685 | 465 | 245 | 25 | 2236 | 2016 | 1796 | 1576 | 1356 | 1136 | 916 |
| 4 | | 884 | 664 | 444 | 224 | 4 | 2215 | 1995 | 1775 | 1555 | 1335 | 1115 | 895 | 675 | 455 | 235 | 15 | 2226 | 2006 | 1786 | 1566 | 1346 | 1126 | 906 | 686 | 466 | 246 | 26 | 2237 | 2017 | 1797 | 1577 | 1357 | 1137 |
| 5 | | 1105 | 885 | 665 | 445 | 225 | 5 | 2216 | 1996 | 1776 | 1556 | 1336 | 1116 | 896 | 676 | 456 | 236 | 16 | 2227 | 2007 | 1787 | 1567 | 1347 | 1127 | 907 | 687 | 467 | 247 | 27 | 2238 | 2018 | 1798 | 1578 | 1358 |
| 6 | | 1326 | 1106 | 886 | 666 | 446 | 226 | 6 | 2217 | 1997 | 1777 | 1557 | 1337 | 1117 | 897 | 677 | 457 | 237 | 17 | 2228 | 2008 | 1788 | 1568 | 1348 | 1128 | 908 | 688 | 468 | 248 | 28 | 2239 | 2019 | 1799 | 1579 |
| 7 | | 1547 | 1327 | 1107 | 887 | 667 | 447 | 227 | 7 | 2218 | 1998 | 1778 | 1558 | 1338 | 1118 | 898 | 678 | 458 | 238 | 18 | 2229 | 2009 | 1789 | 1569 | 1349 | 1129 | 909 | 689 | 469 | 249 | 29 | 2240 | 2020 | 1800 |
| 8 | | 1768 | 1548 | 1328 | 1108 | 888 | 668 | 448 | 228 | 8 | 2219 | 1999 | 1779 | 1559 | 1339 | 1119 | 899 | 679 | 459 | 239 | 19 | 2230 | 2010 | 1790 | 1570 | 1350 | 1130 | 910 | 690 | 470 | 250 | 30 | 2241 | 2021 |
| 9 | | 1989 | 1769 | 1549 | 1329 | 1109 | 889 | 669 | 449 | 229 | 9 | 2220 | 2000 | 1780 | 1560 | 1340 | 1120 | 900 | 680 | 460 | 240 | 20 | 2231 | 2011 | 1791 | 1571 | 1351 | 1131 | 911 | 691 | 471 | 251 | 31 | 2242 |
| 10 | | 2210 | 1990 | 1770 | 1550 | 1330 | 1110 | 890 | 670 | 450 | 230 | 10 | 2221 | 2001 | 1781 | 1561 | 1341 | 1121 | 901 | 681 | 461 | 241 | 21 | 2232 | 2012 | 1792 | 1572 | 1352 | 1132 | 912 | 692 | 472 | 252 | 32 |

*Рис. 1.4.1.* Начальная часть таблицы решений системы сравнений для $(b_1, b_2, b_3)$ при трех модулях: $m_1 = 11$, $m_2 = 13$, $m_3 = 17$

Число в позиционном виде можно представить также в двумерном виде:

$$L \equiv 221 b_1 + 2211 b_2^3 \pmod{2431}, \tag{1.3.22}$$

а $b_3^2$ можно записать в виде следующего выражения:

$$b_3^2 \equiv 170 b_2 + 52 b_3 \pmod{221}. \tag{1.3.24}$$

Для быстрого определения количества витков для $b_2^3$ по $(b_2, b_3)$ можно воспользоваться следующей таблицей.

| | b3 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b2 | 0 | 4 | 8 | 12 | 16 | 3 | 7 | 11 | 15 | 2 | 6 | 10 | 14 | 1 | 5 | 9 | 13 | | ← Число витков | | | | | | | | | | | |
| 0 | | 0 | 52 | 104 | 156 | 208 | 39 | 91 | 143 | 195 | 26 | 78 | 130 | 182 | 13 | 65 | 117 | 169 | | | | | | | | | | | | |
| 1 | | 170 | 1 | 53 | 105 | 157 | 209 | 40 | 92 | 144 | 196 | 27 | 79 | 131 | 183 | 14 | 66 | 118 | 170 | | | | | | | | | | | |
| 2 | | 119 | 171 | 2 | 54 | 106 | 158 | 210 | 41 | 93 | 145 | 197 | 28 | 80 | 132 | 184 | 15 | 67 | 119 | 171 | | | | | | | | | | |
| 3 | | 68 | 120 | 172 | 3 | 55 | 107 | 159 | 211 | 42 | 94 | 146 | 198 | 29 | 81 | 133 | 185 | 16 | 68 | 120 | 172 | | | | | | | | | |
| 4 | | 17 | 69 | 121 | 173 | 4 | 56 | 108 | 160 | 212 | 43 | 95 | 147 | 199 | 30 | 82 | 134 | 186 | 17 | 69 | 121 | 173 | | | | | | | | |
| 5 | | 187 | 18 | 70 | 122 | 174 | 5 | 57 | 109 | 161 | 213 | 44 | 96 | 148 | 200 | 31 | 83 | 135 | 187 | 18 | 70 | 122 | 174 | | | | | | | |
| 6 | | 136 | 188 | 19 | 71 | 123 | 175 | 6 | 58 | 110 | 162 | 214 | 45 | 97 | 149 | 201 | 32 | 84 | 136 | 188 | 19 | 71 | 123 | 175 | | | | | | |
| 7 | | 85 | 137 | 189 | 20 | 72 | 124 | 176 | 7 | 59 | 111 | 163 | 215 | 46 | 98 | 150 | 202 | 33 | 85 | 137 | 189 | 20 | 72 | 124 | 176 | | | | | |
| 8 | | 34 | 86 | 138 | 190 | 21 | 73 | 125 | 177 | 8 | 60 | 112 | 164 | 216 | 47 | 99 | 151 | 203 | 34 | 86 | 138 | 190 | 21 | 73 | 125 | 177 | | | | |
| 9 | | 204 | 35 | 87 | 139 | 191 | 22 | 74 | 126 | 178 | 9 | 61 | 113 | 165 | 217 | 48 | 100 | 152 | 204 | 35 | 87 | 139 | 191 | 22 | 74 | 126 | 178 | | | |
| 10 | | 153 | 205 | 36 | 88 | 140 | 192 | 23 | 75 | 127 | 179 | 10 | 62 | 114 | 166 | 218 | 49 | 101 | 153 | 205 | 36 | 88 | 140 | 192 | 23 | 75 | 127 | 179 | | |
| 11 | | 102 | 154 | 206 | 37 | 89 | 141 | 193 | 24 | 76 | 128 | 180 | 11 | 63 | 115 | 167 | 219 | 50 | 102 | 154 | 206 | 37 | 89 | 141 | 193 | 24 | 76 | 128 | 180 | |
| 12 | | 51 | 103 | 155 | 207 | 38 | 90 | 142 | 194 | 25 | 77 | 129 | 181 | 12 | 64 | 116 | 168 | 220 | 51 | 103 | 155 | 207 | 38 | 90 | 142 | 194 | 25 | 77 | 129 | 181 |

Решение системы сравнений $b_3^2 \equiv 170 b_2 + 52 b_3 \pmod{221}$ с количеством витков, указанным во второй строке $\left(n_1(i) = N_3^{23} i \bmod (m_3),\ N_2^3 = 4,\ i = 0, \ldots, 16\right)$ (см. рис. 1.3.10)

**Пример 1**

Сравнить два числа: $(4, 2, 15)$ и $(1, 3, 12)$.

Находим количество витков для $(b_2, b_3)$ по таблице на рис. 1.4.2:

$n(2, 15) = 1$, $n(3, 12) = 2$.

Количество витков для $(3, 12)$ больше, чем для $(2, 15)$, поэтому $(1, 3, 12) > (4, 2, 15)$.

*Проверка*: $L \equiv 221 b_1 + 1496 b_2 + 715 b_3 \pmod{2431}$. Решения показаны на рис. 1.4.1:

$L_1 = (1, 3, 12) = 1134$ и $L_2 = (4, 2, 15) = 15$.

**Пример 2**

Сравнить два числа: $(4, 2, 15)$ и $(2, 2, 15)$.

Находим количество витков для $(b_2, b_3)$:

$n(2, 15) = 1$.

Количество витков одинаковое для первого и второго модульного числа, поэтому находим значения $b_2^3$. Для $(2, 15)$ $b_2^3 = 15$.

По рис. 1.4.1 для $(b_1, b_2^3)$ находим количество витков:

для $(4, 2, 15) \Rightarrow (4, 15)$ $n(4, 15) = 1$;

для $(2, 2, 15) \Rightarrow (2, 15)$ $n(2, 15) = 182$.

Таким образом, $(2, 2, 15) > (4, 2, 15)$.

*Проверка*: $L_1 = (2, 2, 15) = 2004$ и $L_2 = (4, 2, 15) = 15$.

Для *n*-мерного случая процедура развертывания аналогична.

Таким образом, сравнение чисел в модульном представлении выполняется достаточно просто. Максимальное время выполнения сравнения равно времени на восстановление числа в позиционном виде, однако в большинстве случаев определение количества витков может решить эту задачу на более раннем этапе. Поэтому для большинства сравниваемых чисел время выполнения алгоритма значительно уменьшается.

## 1.5. АНАЛИЗ ПЕРЕПОЛНЕНИЯ ЧИСЕЛ ПРИ СЛОЖЕНИИ ИЛИ УМНОЖЕНИИ В МОДУЛЬНОМ ПРЕДСТАВЛЕНИИ

При сложении или умножении чисел в модульном представлении может наступить **переполнение**. Переполнение возникает, если в результате выполнения арифметического действия число превышает максимальное допустимое для модульного представления значение. Эту ситуацию необходимо контролировать для большинства арифметических операций.

Рассмотрим модульное представление при определении времени по часам с 24-часовым циферблатом. Пусть у нас 23 часа; вычислим, какое время суток наступит через 22 часа:

$$(23 + 22) \bmod 24 = 45 \bmod 24 = 21 \bmod 24 .$$

В результате получилось 9 часов вечера, но мы должны убедиться, что это время следующих суток. Для этого необходимо определить, что в результате сложения

произошло переполнение. Для суток значения будут меняться через один виток (через 24 часа), поэтому если

$$(23+22)=45=24+21,$$

то результирующее значение суммы часов (45) можно представить как сумму одного витка (24 часа) плюс значение остатка (21). Если сумма часов (45) больше, чем значение витка (24), то при сложении чисел в модульном представлении наступит переполнение.

Если для одного модуля это сделать достаточно просто, то в случае использования нескольких модулей анализ переполнения можно выполнить лишь при переходе к позиционному виду.

Рассмотрим операцию сложения модульных чисел:

$$\begin{aligned}(a_1, a_2, \ldots, a_n)+(b_1, b_2, \ldots, b_n) \bmod (m_1, m_2, \ldots, m_n) = \\ = (a_1+b_1) \bmod m_1, \ldots, (a_n+b_n) \bmod m_n.\end{aligned} \quad (1.5.1)$$

В этом случае максимальное значение определяется величиной $m_1, m_2, \ldots, m_{m-1}$.

Однако подсчет количества витков может значительно сократить количество операций для анализа переполнения, так же как и при сравнении модульных чисел [17].

### 1.5.1. АНАЛИЗ ПЕРЕПОЛНЕНИЯ ПРИ СЛОЖЕНИИ ЧИСЕЛ, ПРЕДСТАВЛЕННЫХ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ ПРИ ДВУХ МОДУЛЯХ

Сумму двух чисел при сложении в модульном представлении можно записать как

$$\left[(a_1, a_2)+(b_1, b_2)\right] \bmod (m_1 m_2) = (a_1+b_1) \bmod m_1,\ (a_2+b_2) \bmod m_2. \quad (1.5.2).$$

Используем подсчет количества витков:

$$n(a_1, a_2) m_1 + a_1 + n(b_1, b_2) m_1 + b_1 = \left[n(a_1, a_2) + n(b_1, b_2)\right] + b_1 + a_1. \quad (1.5.3)$$

Сумма не будет выходить за диапазон, если сумма витков и $b_1 + a_1$ не выйдут за диапазон. Если учесть, что $b_1 + a_1$ максимально меняется от нуля до $(2m_1 - 1)$, то сложение этих чисел может привести к максимальному увеличению количества витков на единицу.

Можно сформулировать правило определения переполнения при сложении чисел в модульном представлении с двумя модулями при учете количества витков.

При сложении двух чисел в модульном представлении переполнение наступит в следующих случаях.

1. Если сумма витков равна сумме витков каждого слагаемого и меньше максимального количества витков, то переполнения не произойдет.

2. Если сумма витков на единицу больше суммы витков для каждого слагаемого и меньше максимального количества витков, то переполнения не произойдет.

3. Если сумма витков меньше суммы витков слагаемых, то наступит переполнение.

Рассмотрим примеры для двух модулей: $m_1 = 11$ и $m_2 = 17$. Максимальное количество витков не может превышать 16. Для быстрого определения количества витков можно воспользоваться рис. 1.3.1 (см. раздел 1.3.1).

Рассмотрим примеры, показанные в табл. 1.5.1.

Таблица 1.5.1

**Анализ переполнения при сложении чисел при двух модулях**

| Представление числа | Позиционное | Модульное |
|---|---|---|
| Число витков | 11 + 16 = 27 | (0, 11) + (5, 16) = (5, 27) = (5, 10)<br>1 1 2 |
| Число витков | 21 + 21 = 42 | (10, 4) + (10, 4) = (20, 8) = (9, 8)<br>1 1 3 |
| Число витков | 143 + 43 = 186 | (0, 7) + (10, 9) = (10, 16)<br>13 3 16 |
| Число витков | **144 + 43 = 187** | **(1, 8) + (10, 9) = (11, 17) = (0, 0)**<br>**13 3 0** |

В таблице представлены числа, их представления в модульной форме и значения витков. Жирным шрифтом выделен пример сложения чисел, когда наступает переполнение.

Видно, что определение переполнения при сложении двух чисел в модульном представлении производится достаточно просто.

## 1.5.2. АНАЛИЗ ПЕРЕПОЛНЕНИЯ ПРИ СЛОЖЕНИИ ЧИСЕЛ, ПРЕДСТАВЛЕННЫХ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ ПРИ ТРЕХ МОДУЛЯХ

Сложение модульных чисел при трех модулях можно записать как

$$\begin{aligned}&\left[(a_1, a_2, a_3) + (b_1, b_2, b_3)\right] \bmod (m_1, m_2, m_3) = \\ &= (a_1 + b_1) \bmod m_1, (a_2 + b_2) \bmod m_2, (a_3 + b_3) \bmod m_3\end{aligned} \tag{1.5.4}$$

или с учетом числа витков

$$\begin{aligned}&n\,(a_1, a_2, a_3)\; m_1 + a_1 + n\,(b_1, b_2, b_3)\; m_1 + b_1 = \\ &= \left[n\,(a_1, a_2, a_3) + n\,(b_1, b_2, b_3)\right] + b_1 + a_1.\end{aligned} \tag{1.5.5}$$

Рассмотрим примеры сложения чисел для трех модулей: $m_1 = 11$, $m_2 = 13$ и $m_3 = 17$. В табл. 1.4.2 жирным шрифтом выделен пример, когда наступает переполнение.

Таблица 1.5.2

**Анализ переполнения при сложении чисел при трех модулях**

| Представление числа | Позиционное | Модульное |
|---|---|---|
| Число витков | 11 + 16 = 27 | (0, 11, 11) + (5, 3, 16) = (5, 14, 27) = (5, 1, 10)<br>1 1 2 |
| Число витков | 2211 + 30 = 2241 | (0, 1, 1) + (8, 4, 13) = (8, 5, 14) = (8, 5, 14)<br>201 2 203 |
| Число витков | **2211 + 221 = 2432** | **(0, 1, 1) + (1, 0, 0) = (1, 1, 1) = (1, 1, 1)**<br>**201 23 0** |

Таким образом, при сложении двух чисел в модульном представлении переполнение наступит в следующих случаях.

1. По $b_2$ и $b_3$ для двух чисел (так же как и для двух модулей) находим значение $b_3^2$. Если сумма числа витков больше, чем максимальное количество витков, то количество витков результирующего модульного числа будет меньше суммы витков слагаемых.

2. Если $b_1 + c_1 \geq m_1$, то к сумме витков добавляется единица, и результирующее количество витков суммы модульных чисел меньше суммы витков слагаемых.

Анализ переполнения при $n$ модулях выполняется аналогично.

Рассмотрим, как определить переполнение при умножении модульных чисел.

### 1.5.3. ПЕРЕПОЛНЕНИЕ ПРИ УМНОЖЕНИИ МОДУЛЬНЫХ ЧИСЕЛ

При умножении в модульном представлении имеем

$$\begin{aligned}\left[n(b_1, b_2)m_1 + b_1\right]\left[n(c_1, c_2)m_1 + c_1\right] &= \left[n(b_1, b_2)\, n(c_1, c_2)m_1 m_1 + \right.\\ \left. + n(c_1,\, c_2)b_1 m_1 + n(b_1, b_2)c_1 m_1 + b_1 c_1\right] \bmod m_1 &= n(b_1, b_2)n(c_1, c_2)m_1 + \\ + n(c_1, c_2)b_1\, n(b_1, b_2)\, c_1 + b_1 c_1 &= n_p m_1 + b_1 c_1,\end{aligned} \quad (1.5.6)$$

где число витков будет равно

$$n_p = n(b_1, b_2)\, n(c_1, c_2)\, m_1 + n(c_1, c_2)b_1 + n(b_1, b_2)c_1. \quad (1.5.7)$$

Таким образом, при умножении двух чисел в модульном представлении переполнение наступит в следующих случаях.

1. Если результирующее количество витков больше, чем максимальное.

2. Если $b_1 c_1 \geq m_1$, то к сумме витков добавляется целая часть $\text{int}\left[b_1 c_1 / m_1\right]$, и эта сумма превышает максимальное количество витков.

**Пример 1**

$12 \cdot 13 = 156, \quad (1, 12) \cdot (2, 13) = (2, 3),$

$b_1 c_1 < m_1, \quad 2 < 11$

$n(1, 12) = 1, \quad n(2, 13) = 1, \quad n(2, 3) = 14 \rightarrow n_p = 11 + 1 + 2 = 14$

В этом случае переполнения не возникает.

**Пример 2**

$51 \cdot 14 = 204, \quad (7, 0) \cdot (4, 4) = (6, 0),$

$b_1 c_1 > m_1, \quad 28 > 11, \quad \text{int}\left[28 / 11\right] = 2,$

$n(7, 0) = 1, \quad n(4, 4) = 0, \quad n(6, 0) = 1 \rightarrow n_p = 16 + 2 > 16.$

Возникает переполнение.

При сложении или перемножении модулей для вычисления количества витков получаются достаточно большие числа. К сожалению, в этом случае для хранения значений количества витков тоже необходимы длинные числа.

## 1.6. ДЕЛЕНИЕ МОДУЛЬНЫХ ЧИСЕЛ

Деление – одна из самых проблемных операций в арифметике [19].

Определение деления обычно задается как нахождение обратного числа. Если $a / b = x$, то $a = bx$, и нам необходимо найти число $x$, которое будет соответствовать этим равенствам.

Деление чисел в системе остаточных классов часто определяется аналогичным образом:

$$(a_1, a_2, \ldots, a_k) / (b_1, b_2, \ldots, b_k) \equiv (x_1, x_2, \ldots, x_k); \tag{1.6.1}$$

$$(a_1, a_2, \ldots, a_k) \equiv (b_1, b_2, \ldots, b_k)(x_1, x_2, \ldots, x_k) \equiv \equiv b_1 x_1 \bmod m_1, \ldots, b_k x_k \bmod m_k. \tag{1.6.2}$$

В этом случае необходимо найти $x_i$ из уравнений вида

$$a_i \equiv b_i x_i \bmod m_i. \tag{1.6.3}$$

Таким образом, деление сводится к нахождению решений сравнений первой степени.

Рассмотрим сначала способ деления, основанный на нахождении решений сравнений первой степени.

### 1.6.1. НАХОЖДЕНИЕ РЕШЕНИЙ СРАВНЕНИЙ ПЕРВОЙ СТЕПЕНИ НА ОСНОВЕ ТЕОРИИ НЕПРЕРЫВНЫХ ДРОБЕЙ

Следующая теорема [18] о симметричности отношения сравнимости позволяет искать решение (1.6.3) в виде

$$bx \equiv a \bmod m. \tag{1.6.4}$$

**Теорема**
Симметричность отношения сравнимости:

если $a \equiv b \bmod (m)$, то $b \equiv a \bmod (m)$.

Существование решений и их количество определяется из следующей теоремы [2].

**Теорема**

Пусть НОД ($b$, $m$) = $d$.
Сравнение $bx \equiv a \bmod(m)$ невозможно, если $a$ не делится на $d$.
При $a$, кратном $d$, сравнение имеет $d$ решений.

Алгоритм нахождения наибольшего общего делителя НОД приведен в разделе 1.7.

Приведем решение системы сравнений, основанное на теории непрерывных дробей, причем достаточно ограничиться случаем, когда НОД ($b$, $m$) = 1 [11].

Пусть

$$111x \equiv 75 \bmod(321),$$

поскольку НОД (111, 321) = 3, причем 75 кратно трем. Поэтому сравнение имеет три решения. Разделив обе части сравнения и модуль на три, получим

$$37x \equiv 25 \bmod(107).$$

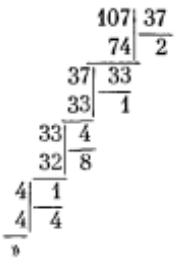
Разложим в непрерывную дробь отношение $m / b$:

$$\frac{m}{b} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \ldots}}.$$

Сравнение будет иметь решение $x \equiv (-1)^{n-1} P_{n-1}\, a \bmod(m)$.

Для нахождения решения достаточно определить $P_{n-1}$.

Разложим в непрерывную дробь отношение $m / b = 107 : 37$.

| | |
|---|---|
| $107 / 37 = 2 \cdot 37 + 33$<br>$37 / 33 = 1 \cdot 33 + 4$<br>$33 / 4 = 8 \cdot 4 + 1$<br>$4 / 1 = 4 \cdot 1 + 0$ |  |

Находим $P_{n-1}$, последовательно вычисляя $P_s = q_s P_{s-1} + P_{s-2}$.

| $n$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $q_s$ | – | **2** | **1** | **8** | **4** |
| $P_s$ | 1 | **2** | **3** | **26** | **107** |

Сравнение имеет решение $x \equiv (-1)^{n-1} P_{n-1}\, a \bmod(m)$. Поскольку $n = 4,\ a = 25$ и $P_{n-1} = 26$, то

$$x \equiv (-1)^3 \cdot 26 \cdot 25 \bmod(107) \equiv -26 \cdot 25 \bmod(107) \equiv -650 \bmod(107) \equiv 99 \bmod(107).$$

Сравнение $37x \equiv 25 \bmod(107)$ будет иметь одно решение:

$$x \equiv 99 \bmod(107).$$

Исходное сравнение $111x \equiv 75 \bmod(321)$ будет иметь три решения:

$$x_1 \equiv 99 \bmod(321);$$

$$x_2 \equiv 99 + 107 \bmod(321) \equiv 206 \bmod(321);$$

$$x_3 \equiv 99 + 2 \cdot 107 \bmod(321) \equiv 313 \bmod(321).$$

Таким образом, результат деления двух чисел можно записать как

$$x \equiv 25 / 37 \bmod(107);$$

$$37x \equiv 25 \bmod(107);$$

$$x \equiv 99 \bmod(107).$$

*Проверка*:

$$37 \cdot 99 \bmod(107) \equiv 3663 \bmod(107) \equiv 25 \bmod(107).$$

**Пример 1**

Найти решения сравнения $17x \equiv 1 \pmod{11}$.

Находим $q_s$.

| $n$ | Представление |
|---|---|
| 1 | $11/17 = 0 \cdot 17 + 11$ |
| 2 | $17/11 = 1 \cdot 11 + 6$ |
| 3 | $11/6 = 1 \cdot 6 + 5$ |
| 4 | $6/5 = 1 \cdot 5 + 1$ |
| 5 | $5/1 = 5 \cdot 1 + 0$ |

$q_s = \{0, 1, 1, 1, 5\}$, $n = 5$ и $n - 1 = 4$.

Далее находим $P_{n-1}$, последовательно вычисляя $P_s = q_s P_{s-1} + P_{s-2}$.

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| $q_s$ | – | 0 | 1 | 1 | 1 | 5 |
| $P_s$ | 1 | 0 | 1 | 1 | **2** | 11 |

Поскольку НОД $(17, 11) = 1$, сравнение имеет единственное решение:

$$x \equiv (-1)^{n-1} P_{n-1}\, b \bmod(m) = 2 \bmod(11) \Rightarrow x = 2.$$

**Пример 2**

Найти решения сравнения $11x \equiv 1 \pmod{17}$.

| $n$ | Представление |
|---|---|
| 1 | $17/11 = 1 \cdot 11 + 6$ |
| 2 | $11/6 = 1 \cdot 6 + 5$ |
| 3 | $6/5 = 1 \cdot 5 + 1$ |
| 4 | $5/1 = 1 \cdot 5 + 0$ |

Следовательно, $n = 4$, и $n - 1 = 3$.

Находим $P_{n-1}$, последовательно вычисляя $P_s = q_s P_{s-1} + P_{s-2}$.

| $n$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $q_s$ | – | 1 | 1 | 1 | 1 |
| $P_s$ | 1 | 1 | 2 | **3** | 5 |

Поскольку $\text{НОД}(11, 17) = 1,$ сравнение имеет единственное решение:

$$x \equiv (-1)^{n-1} P_{n-1}\ a \bmod (m) = -3 \bmod (17) = 14 \bmod (17) \Rightarrow x = 14.$$

Однако определение операции деления таким образом выглядит не совсем естественным.

## 1.6.2. ОПРЕДЕЛЕНИЕ РЕЗУЛЬТАТА ДЕЛЕНИЯ ЧИСЕЛ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ

Посчитаем результат деления в системе остаточных классов с двумя модулями $m_1 = 11$ и $m_2 = 17$ следующих чисел:

$$x = 80 / 20.$$

Представление модульных чисел можно сделать в виде таблицы. Число 80 в системе остаточных классов $m_1 = 11$ и $m_2 = 17$ будет иметь вид $(3, 12)$, а число 20 – вид $(9, 3)$.

| | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| b1 | | | | | | | | | | | | | | | | | | |
| 0 | | 0 | 154 | 121 | 88 | 55 | 22 | 176 | 143 | 110 | 77 | 44 | 11 | 165 | 132 | 99 | 66 | 33 |
| 1 | | 34 | 1 | 155 | 122 | 89 | 56 | 23 | 177 | 144 | 111 | 78 | 45 | 12 | 166 | 133 | 100 | 67 |
| 2 | | 68 | 35 | 2 | 156 | 123 | 90 | 57 | 24 | 178 | 145 | 112 | 79 | 46 | 13 | 167 | 134 | 101 |
| 3 | | 102 | 69 | 36 | 3 | 157 | 124 | 91 | 58 | 25 | 179 | 146 | 113 | 80 | 47 | 14 | 168 | 135 |
| 4 | | 136 | 103 | 70 | 37 | 4 | 158 | 125 | 92 | 59 | 26 | 180 | 147 | 114 | 81 | 48 | 15 | 169 |
| 5 | | 170 | 137 | 104 | 71 | 38 | 5 | 159 | 126 | 93 | 60 | 27 | 181 | 148 | 115 | 82 | 49 | 16 |
| 6 | | 17 | 171 | 138 | 105 | 72 | 39 | 6 | 160 | 127 | 94 | 61 | 28 | 182 | 149 | 116 | 83 | 50 |
| 7 | | 51 | 18 | 172 | 139 | 106 | 73 | 40 | 7 | 161 | 128 | 95 | 62 | 29 | 183 | 150 | 117 | 84 |
| 8 | | 85 | 52 | 19 | 173 | 140 | 107 | 74 | 41 | 8 | 162 | 129 | 96 | 63 | 30 | 184 | 151 | 118 |
| 9 | | 119 | 86 | 53 | 20 | 174 | 141 | 108 | 75 | 42 | 9 | 163 | 130 | 97 | 64 | 31 | 185 | 152 |
| 10 | | 153 | 120 | 87 | 54 | 21 | 175 | 142 | 109 | 76 | 43 | 10 | 164 | 131 | 98 | 65 | 32 | 186 |

Таблица чисел в позиционном представлении в системе остаточных классов для модулей $m_1 = 11,\ m_2 = 17$ (см. рис. 1.4.1)

Таким образом, нужно найти результат деления модульных чисел

$$x \equiv (3,12)/(9,3).$$

В этом случае

$$(9,3)\cdot(x_1,x_2) \equiv (3,12).$$

Необходимо найти решение двух сравнений

$$9x_1 \equiv 3 \bmod (11);$$

$$3x_2 \equiv 12 \bmod (17).$$

$9x_1 \equiv 3 \bmod(11)$

$11/9 = 1\cdot 9 + 2$

$9/2 = 4\cdot 2 + 1$

$2/1 = 2\cdot 1 + 0$

| $n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $q_s$ | – | **1** | **4** | **2** |
| $P_s$ | 1 | **1** | **5** | **11** |

$$x_1 \equiv (-1)^{n-1} P_{n-1}\, a \bmod (m) = (-1)^2 5\cdot 3 \bmod (11) = 4 \bmod (11)$$

$3x_2 \equiv 12 \bmod (17)$

$17/3 = 5\cdot 3 + 2$

$3/2 = 1\cdot 2 + 1$

$2/1 = 1\cdot 2 + 0$

| $n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $q_s$ | – | **5** | **1** | **1** |
| $P_s$ | 1 | **5** | **6** | **11** |

$$x_2 \equiv (-1)^{n-1} P_{n-1}\, a \bmod (m) = (-1)^2 6\cdot 12 \bmod (17) = 72 \bmod (17) = 4 \bmod (17)$$

Получим

$$x \equiv (3,12)/(9,3) \equiv (4,4) = 4.$$

*Проверка*:

$$(9,3)\cdot(x_1,x_2)\equiv(3,12);$$

$$(9,3)\cdot(4,4)\equiv\big(36\bmod(11),12\bmod(17)\big)\equiv(3,12).$$

Приведенный результат деления соответствует случаю, когда числа делятся без остатка.

Возьмем число, которое не делится нацело, т. е. в результате должен быть какой-то остаток:

$$x=81/20;$$

$$x\equiv(4,13)/(9,3);$$

$$(9,3)\cdot(x_1,x_2)\equiv(4,13);$$

$$9x_1\equiv4\bmod(11);$$

$$3x_2\equiv13\bmod(17).$$

$9x_1\equiv4\bmod(11)$

$11/9=1\cdot9+2$

$9/2=4\cdot2+1$

$2/1=2\cdot1+0$

| $n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $q_s$ | – | **1** | **4** | **2** |
| $P_s$ | 1 | **1** | **5** | **11** |

$$x_1\equiv(-1)^{n-1}P_{n-1}\,a\bmod(m)=(-1)^2\,5\cdot4\bmod(11)=9\bmod(11)$$

$3x_2\equiv13\bmod(17)$

$17/3=5\cdot3+2$

$3/2=1\cdot2+1$

$2/1=1\cdot2+0$

| $n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $q_s$ | – | **5** | **1** | **1** |
| $P_s$ | 1 | **5** | **6** | **11** |

$$x_2\equiv(-1)^{n-1}P_{n-1}\,a\bmod(m)=(-1)^2\,6\cdot13\bmod(17)=78\bmod(17)=10\bmod(17)$$

Получим

$$x \equiv (4,13)/(9,3) \equiv (9,10) = 163 .$$

*Проверка*:

$$(9,3)\cdot(x_1, x_2) \equiv (4,13);$$

$$(9,3)\cdot(9,10) \equiv \left(81 \bmod (11), 30 \bmod (17)\right) \equiv (4,13).$$

Ответ хотя и удовлетворяет формальным признакам, но выглядит достаточно странным. Числа не могут быть больше, чем делимое, если делитель больше или равен единице.

В целочисленной арифметике при делении чисел должно образовываться целое число, полученное в результате деления, плюс некоторый остаток.

Рассмотрим алгоритм, который дает более естественный результат.

### 1.6.3. ДЕЛЕНИЕ В МНОГОМОДУЛЬНЫХ СИСТЕМАХ

Если разделить одно целое число на другое, то получим следующее выражение:

$$\frac{A}{B} = N + \text{остаток } r . \qquad (1.6.5)$$

Существует теорема о делимости с остатком.

**Теорема**

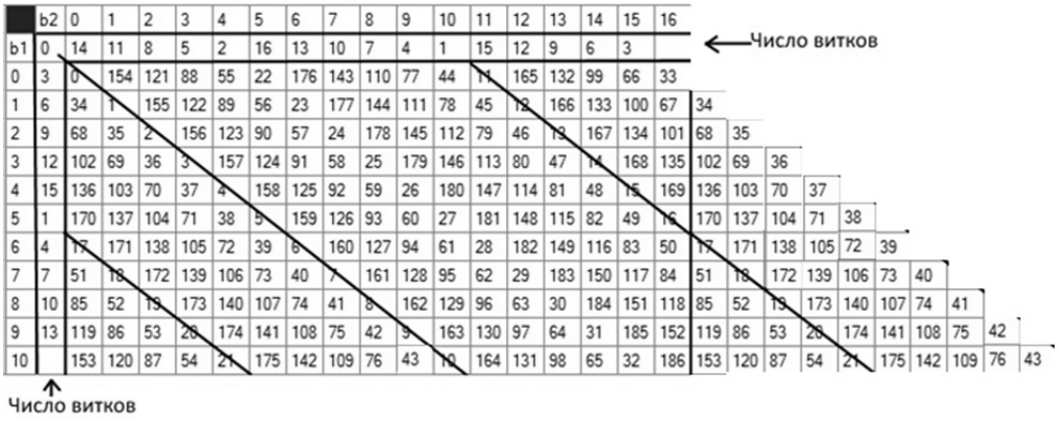
Для любого целого числа $A$ и любого натурального числа $B$ существует единственная пара чисел $N$ и $R$ таких, что $A = BN + r$, где $0 \le r < B$.

В теореме подразумевается, что $N \le A$.

Рассмотрим пример, при котором числа делятся без остатка.

#### ДЕЛЕНИЕ ЧИСЕЛ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ, КОТОРЫЕ ДЕЛЯТСЯ БЕЗ ОСТАТКА

Из приведенной ниже таблицы можно определить взаимное соответствие чисел в системе остаточных классов и в позиционной форме.




Определение числа витков на поверхности тора (см. рис. 1.4.2)

Рассмотрим несколько примеров.

**Пример 1**

Дано

$$\frac{80_{10}}{20_{10}} \equiv \frac{(3,12)}{(9,3)} \equiv (4,4) = 4_{10}.$$

Решение рассмотрено в предыдущем разделе.

В этом случае $(4,4) < (3,12)$, что соответствует $4 < 80$ в позиционном виде.

Более интересным является случай, когда деление происходит с некоторым остатком.

## ДЕЛЕНИЕ ЧИСЕЛ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ, КОТОРЫЕ ДЕЛЯТСЯ С ОСТАТКОМ

Даны следующие числа:

$$x = \frac{20}{80} = \frac{1}{4}, \quad 20 = (9,3), \quad 80 = (3,12).$$

В модульном виде деление этих чисел запишется как

$$(x_1, x_2) \equiv \frac{(9,3)}{(3,12)}.$$

Это выражение можно разбить на два:

$$x_1 \equiv \frac{9}{3} (\mathrm{mod}\, 11) = 3; \qquad x_2 \equiv \frac{3}{12} (\mathrm{mod}\, 17).$$

Для нахождения $x_2$ необходимо решить только второе сравнение.

$$12x_2 \equiv 3 \bmod (17)$$

$$17 / 12 = 1 \cdot 12 + 5$$
$$12 / 5 = 2 \cdot 5 + 2$$
$$5 / 2 = 2 \cdot 1 + 1$$
$$1 / 1 = 1 \cdot 1 + 0$$

| $n$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $q_s$ | – | **1** | **2** | **2** | **1** |
| $P_s$ | 1 | **1** | **3** | **7** | |

$$x_2 \equiv (-1)^{n-1} P_{n-1}\, a \bmod (m) = (-1)^3 7 \cdot 3 \bmod (17) = -21 \bmod (17) \equiv 13 \bmod (17)$$

Получим

$$\frac{20}{80} \equiv \frac{(9,3)}{(3,12)} \equiv (3,13) = 47;$$

$$(3,13) > (9,3).$$

Или, если рассмотреть решения только для НОД $= 1$:

$$4x_1 \equiv 1 \bmod (11)$$

$$11 / 4 = 2 \cdot 4 + 3$$
$$4 / 3 = 1 \cdot 3 + 1$$
$$3 / 1 = 3 \cdot 1 + 0$$

| $n$ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $q_s$ | – | **2** | **1** | **3** |
| $P_s$ | 1 | **2** | **3** | |

$$x_1 \equiv (-1)^{n-1} P_{n-1}\, a \bmod (m) = (-1)^2 3 \cdot 1 \bmod (11) = 3 \bmod (11)$$

<table>
<tr><td colspan="2">$4x_2 \equiv 1 \bmod(17)$</td></tr>
<tr><td>$17 / 4 = 4 \cdot 4 + 1$<br>$4 / 1 = 4 \cdot 1 + 0$</td><td><table>
<tr><td>$n$</td><td>0</td><td>1</td><td>2</td></tr>
<tr><td>$q_s$</td><td>–</td><td>4</td><td>4</td></tr>
<tr><td>$P_s$</td><td>1</td><td>4</td><td></td></tr>
</table></td></tr>
<tr><td colspan="2">$x_1 \equiv (-1)^{n-1} P_{n-1}\, a \bmod(m) = (-1)^1 4 \cdot 1 \bmod(17) = 13 \bmod(17)$</td></tr>
</table>

Результат от деления при сокращении на НОД такой же:

$$\frac{1}{4} \equiv \frac{(1,1)}{(4,4)} \equiv (3,13) = 47;$$

$$(3,13) > (1,1)\,.$$

Необходимо проверить, больше ли числитель знаменателя.
Для данного примера это условие не выполняется:

$$(1,1) < (4,4)\,.$$

В таком случае это число можно представить как дробное или после сокращения НОД оставить в виде дробного числа:

$$\frac{20}{80} = \frac{(9,3)}{(3,12)} = \frac{(1,1)}{(4,4)}\,.$$

**Пример 2**
Возьмем число, которое делится с остатком:

$$\frac{81}{20} = \frac{(4,13)}{(9,3)};$$

$$(x_1, x_2) = \frac{(4,13)}{(9,3)}.$$

Поскольку $(4,13) > (9,3)$, вычитаем до тех пор, пока это условие выполняется:

$$(4,13) - (9,3) = (-5,10) = (-5+11,10) = (6,10) = 61_{10};$$

$$(6,10) - (9,3) = (-3,7) = (-3+11,7) = (8,7) = 41_{10};$$

$$(8,7) - (9,3) = (-1,4) = (-1+11,4) = (10,4) = 21_{10};$$

$$(10,4) - (9,3) = (1,1) = 1_{10}.$$

Теперь определяем числитель, который делится на знаменатель без остатка:

$$(4,13) - (1,1) = (3,12) = 80_{10}.$$

Находим результат деления:

$$\frac{80}{20} \equiv \frac{(3,12)}{(9,3)} \equiv (4,4) = 4_{10}.$$

В результате получим

$$\frac{(4,13)}{(9,3)} \equiv (4,4) + \frac{(1,1)}{(9,3)}.$$

*Проверка*:

$$(9,3)\cdot(x_1, x_2) \equiv (4,13);$$

$$(9,3)\cdot(4,4) \equiv \big(36 \bmod (11), 12 \bmod (17)\big) \equiv (3,12);$$

$$(9,3)\frac{(1,1)}{(9,3)} \equiv (1,1);$$

$$(3,12) + (1,1) \equiv (4,13) = 81.$$

**Пример 3**

Возьмем число

$$\frac{81}{2} = 40 + \frac{1}{2};$$

$$\frac{(4,13)}{(2,2)} \equiv (7,6) + \frac{(1,1)}{(2,2)}.$$

Это более естественный способ деления.

Для многомерного случая операция деления определяется аналогично.

## 1.7. НАХОЖДЕНИЕ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ МОДУЛЬНЫХ ЧИСЕЛ

В модульной арифметике необходимо определять взаимную простоту чисел. Целые числа взаимно простые, если их наибольший общий делитель равен единице. Рассмотрим основные алгоритмы нахождения наибольшего общего делителя (НОД).

Наибольшим общим делителем двух целых чисел $a$ и $b$, одновременно не равных нулю, называется такое наибольшее целое число $d$, на которое $a$ и $b$ делятся без остатка. Из определения понятно, что $d = \text{НОД}(a, b) = \text{НОД}(b, a)$.

### 1.7.1. АЛГОРИТМ НАХОЖДЕНИЯ НОД С ВЗАИМНЫМ ВЫЧИТАНИЕМ

Для нахождения НОД часто используется алгоритм Евклида [21].

В самом простом случае алгоритм Евклида применяется к паре положительных целых чисел и формирует новую пару, которая состоит из меньшего числа и разницы между бóльшим и меньшим числом. Процесс повторяется, пока числа не станут равными. Найденное число и есть наибольший общий делитель исходной пары.

Алгоритм состоит из следующих шагов.

1. Из большего числа вычитаем меньшее.
2. Если получается ноль, то это значит, что числа равны и являются НОД (следует выйти из цикла).
3. Если результат вычитания не равен нулю, то большее число заменяем на результат вычитания.
4. Переходим к пункту 1.

**Пример:** НОД (80, 20).

$\text{НОД}(80 - 20, 20) =$

$= \text{НОД}(60, 20) =$

$= \text{НОД}(40, 20) =$

$= \text{НОД}(20, 20) =$

$= \text{НОД}(20 - 20, 20) = \text{НОД}(0, 20).$

Следовательно, $\text{НОД}(80, 20) = 20$.

Древнегреческие математики называли этот алгоритм «взаимное вычитание».

Реализуем на языке программирования C функцию `gcd` (greatest common divisor):

```
int gcd(int a, int b)
{
  while (a!=b)
   { if (a > b)  a = a – b; else b = b – a; }
  return a;
}
```

Приведенный метод вычисления не является оптимальным. Например, для нахождения НОД $(1\,000\,000, 2)$ следует выполнить 500 000 операций вычитания. Для ускорения вычисления НОД операцию вычитания заменим операцией взятия остатка от деления.

### 1.7.2. АЛГОРИТМ НАХОЖДЕНИЯ НОД ДЕЛЕНИЕМ

Алгоритм состоит из следующих шагов.

1. Большее число делим на меньшее.

2. Если делится без остатка, то меньшее число и есть НОД (следует выйти из цикла).

3. Если есть остаток, то большее число меняем на делитель, а делитель – на остаток от деления.

4. Переходим к пункту 1.

Если $a < b$, то $\text{НОД}(a, b) = \text{НОД}(b, a) = \text{НОД}(b, a \,\%\, b)$, т. е. аргументы функции переставляются. При последующих вызовах функции НОД первый аргумент всегда больше второго. Процесс повторяется, пока второй аргумент не станет равным нулю.

**Пример 1**

Найти НОД для 80 и 20.

$80 / 20 = 4$ (остаток 0);

$\text{НОД}(80, 20) = 20$.

**Пример 2**

Найти НОД для 30 и 18.

$30 / 18 = 1$ (остаток 12);

$18 / 12 = 1$ (остаток 6);

$12 / 6 = 2$ (остаток 0);

НОД $(30, 18) = 6$.

**Пример 3**

Найти НОД для 81 и 20.

$81 / 20 = 4$ (остаток 1);

$20 / 1 = 20$ (остаток 0);

НОД $(81, 20) = 1$.

Нерекурсивный алгоритм определения НОД:

```
int gcd(int a, int b)
 {
   while (b)
      {
        a %= b;
        swap(a, b);
      }
return a;
}
```

Рекурсивный алгоритм определения НОД (знак «%» в языке С обозначает операцию взятия остатка от деления [20]):

```
int gcd(int a, int b)
{
  if (b == 0) return a;
  return gcd(b, a % b);
}
```

Для двух целых чисел $a$ и $b$ существует НОД $(a, b) = d$, и его можно представить в виде линейной комбинации $d = xa + yb$.

Приведем рекурсивный расширенный алгоритм Евклида, который возвращает не только наибольший общий делитель двух чисел, но и коэффициенты его линейного представления.

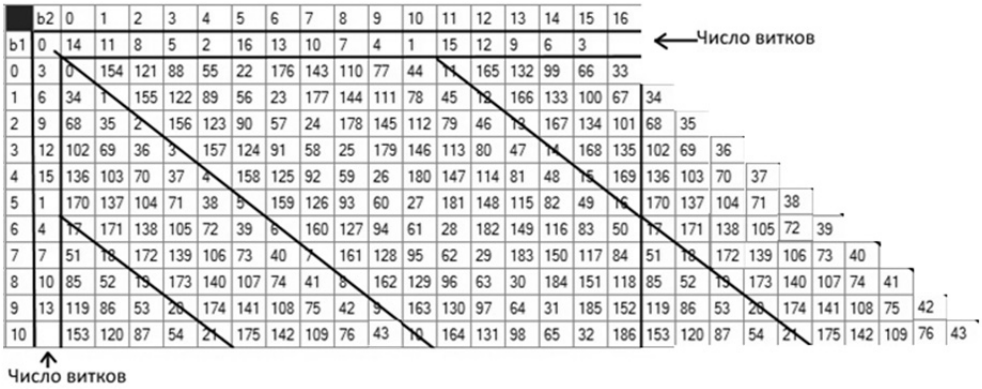
Расширенный алгоритм Евклида:

```
int gcd(int a, int b, int& x, int& y)
 {
    if (b == 0) {
                   x = 1; // База рекурсии.
                          // Если b=0, то НОД=a, 1*a+0=d
                   y = 0;
                   return a;
                }
     int x1, y1;
     int d = gcd(b, a % b, x1, y1); //рекурсивный переход
      x = y1;
      y = x1 - (a / b) * y1;
      return d;
 }
```

### 1.7.3. НАХОЖДЕНИЕ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ В СИСТЕМЕ ОСТАТОЧНЫХ КЛАССОВ С ДВУМЯ МОДУЛЯМИ ПОСЛЕДОВАТЕЛЬНЫМ ВЫЧИТАНИЕМ

Используем алгоритм нахождения НОД вычитанием, поскольку операция вычитания легко реализуется для чисел, представляемых в системе остаточных классов.

Для модулей $m_1 = 11,\ m_2 = 17$ имеем следующую таблицу решений (см. рис. 1.3.5).




**Пример 1**

Найти НОД для 80 и 20.

$$\text{НОД}\left[(3,12),(9,3)\right]=\text{НОД}\left[80_{10},20_{10}\right];$$

$$\text{НОД}\left[(3-9,12-3),(9,3)\right]=\text{НОД}\left[(-6,9),(9,3)\right]=$$

$$=\text{НОД}\left[(5,9),(9,3)\right]=\text{НОД}\left[60_{10},20_{10}\right];$$

$$\text{НОД}\left[(5-9,9-3),(9,3)\right]=\text{НОД}\left[(-4,6),(9,3)\right]=$$

$$=\text{НОД}\left[(7,6),(9,3)\right]=\text{НОД}\left[40_{10},20_{10}\right];$$

$$\text{НОД}\left[(7-9,6-3),(9,3)\right]=\text{НОД}\left[(-2,3),(9,3)\right]=$$

$$=\text{НОД}\left[(9,3),(9,3)\right]=\text{НОД}\left[20_{10},20_{10}\right];$$

$$\text{НОД}\left[(9-9,9-3),(9,3)\right]=\text{НОД}\left[(0,0),(9,3)\right]=\text{НОД}\left[0,20_{10}\right].$$

В результате $\text{НОД}\left[(3,12),(9,3)\right]=(9,3)=20_{10}$.

**Пример 2**

Найти НОД для 81 и 20.

$$\text{НОД}\left[(4,13),(9,3)\right]=\text{НОД}\left[81_{10},20_{10}\right];$$

$$\text{НОД}\left[(4-9,13-3),(9,3)\right]=\text{НОД}\left[(-5,10),(9,3)\right]=$$

$$=\text{НОД}\left[(6,10),(9,3)\right]=\text{НОД}\left[61_{10},20_{10}\right];$$

$$\text{НОД}\left[(6-9,10-3),(9,3)\right]=\text{НОД}\left[(-3,7),(9,3)\right]=$$

$$=\text{НОД}\left[(8,7),(9,3)\right]=\text{НОД}\left[41_{10},20_{10}\right];$$

$$\text{НОД}\left[(8-9,7-3),(9,3)\right]=\text{НОД}\left[(-1,4),(9,3)\right]=$$

$$=\text{НОД}\left[(10,4),(9,3)\right]=\text{НОД}\left[21_{10},20_{10}\right];$$

$$\text{НОД}\left[(10-9,4-3),(9,3)\right]=\text{НОД}\left[(1,1),(9,3)\right]=\text{НОД}\left[1_{10},20_{10}\right];$$

$$\text{НОД}\left[(1-9,1-3),(9,3)\right]=\text{НОД}\left[(-8,-2),(9,3)\right]=$$

$$=\text{НОД}\left[(3,9),(9,3)\right]=\text{НОД}\left[179_{10},20_{10}\right].$$

Можно продолжать эту последовательность операций до получения правильного результата, как в предыдущем примере, а можно воспользоваться результатами, полученными в разделе 1.4 для определения большего из чисел в модульном виде. Если в результате вычислений число превысило максимальный диапазон, то общих делителей нет.

В нашем случае $(3,9)>(1,1)$. Это значит, что общих делителей нет, кроме единицы, т. е. $\text{НОД}\left[(4,13),(9,3)\right]=(1,1)=1_{10}$.

### 1.7.4. НАХОЖДЕНИЕ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ В СИСТЕМЕ МНОГОМОДУЛЬНЫХ ОСТАТОЧНЫХ КЛАССОВ ПОСЛЕДОВАТЕЛЬНЫМ ВЫЧИТАНИЕМ

Найдем НОД для трех модулей $m_1=11, m_2=13, m_3=17$:

$$L\equiv 221\cdot 1\ b_1+11\cdot 201\,b_2^3\ (\text{mod}\,11\cdot 221)\equiv 221b_1+2211b_2^3\ (\text{mod}\,2431). \quad (1.7.1)$$

**Пример 1**

Найти НОД для 80 и 20.

$$\text{НОД}\left[(3,2,12),(9,7,3)\right]=\text{НОД}\left[80_{10},20_{10}\right];$$

$$\text{НОД}\left[(3-9,2-7,12-3),(9,7,3)\right]=\text{НОД}\left[(5,8,9),(9,7,3)\right]=\text{НОД}\left[60_{10},20_{10}\right];$$

$$\text{НОД}\left[(5-9,8-7,9-3),(9,7,3)\right]=\text{НОД}\left[(7,1,6),(9,7,3)\right]=\text{НОД}\left[40_{10},20_{10}\right];$$

$$\text{НОД}\big[(7-9,1-7,6-3),(9,7,3)\big]=\textbf{НОД}\big[\textbf{(9, 7, 3), (9, 7, 3)}\big]=\text{НОД}\big[20_{10},20_{10}\big];$$

$$\text{НОД}\big[(9-9,7-7,3-3),(9,7,3)\big]=\text{НОД}\big[(0,0,0),(9,7,3)\big]=\text{НОД}\big[0_{10},20_{10}\big].$$

В результате $\text{НОД}\big[(3,2,12),(9,7,3)\big]=(9,7,3)=20_{10}$;

НОД = (9, 7, 3) = 20.

Существует алгоритм для модульных чисел, который позволяет записать операции вычисления НОД в обобщенном виде.

## 1.7.5. ОБОБЩЕННЫЙ АЛГОРИТМ ДЛЯ НАХОЖДЕНИЯ НАИБОЛЬШЕГО ОБЩЕГО ДЕЛИТЕЛЯ

При заданных неотрицательных целых числах $u$ и $v$ этот алгоритм определяет вектор $(u_1,u_2,u_3)$ такой, что $u_3=uu_1+vu_2=\text{НОД}(u,v)$.

В процессе вычислений используются вспомогательные векторы $(v_1,v_2,v_3)$ и $(t_1,t_2,t_3)$. Действия с векторами производятся таким образом, что в течение всего процесса вычислений выполняются соотношения

$$ut_1+vt_2=t_3,\quad uu_1+vu_2=u_3,\quad uv_1+vv_2=v_3.$$

Обобщенный алгоритм Евклида можно представить в виде следующей последовательности шагов [6].

1. Начальная установка: $(u_1,u_2,u_3)\leftarrow(1,0,u)$, $(v_1,v_2,v_3)\leftarrow(0,1,v)$.

2. Если $v_3=0$, то алгоритм заканчивается.

3. $q\leftarrow\lfloor u_3/v_3\rfloor$, (операция $\lfloor x\rfloor$ обозначает наибольшее целое число, не превосходящее $x$):

$$(t_1,t_2,t_3)\leftarrow(u_1,u_2,u_3)-(v_1,v_2,v_3)q;$$
$$(u_1,u_2,u_3)\leftarrow(v_1,v_2,v_3);$$
$$(v_1,v_2,v_3)\leftarrow(t_1,t_2,t_3).$$

4. Перейти к шагу 2.

**Пример 1**

Найдем НОД (111, 75) .

| $q$ | $u_1$ | $u_2$ | $u_3$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|---|---|---|
| | 1 | 0 | 111 | 0 | 1 | 75 |
| 1 | 0 | 1 | 75 | 1 | –1 | 36 |
| 2 | 1 | –1 | 36 | –2 | 3 | 3 |
| 12 | –2 | 3 | **3** | – | – | 0 |

НОД (111, 75) = 3 .

**Пример 2**

Найдем НОД (80, 20) .

| $q$ | $u_1$ | $u_2$ | $u_3$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|---|---|---|
| – | 1 | 0 | 80 | 0 | 1 | 20 |
| 4 | 0 | 1 | **20** | 1 | –4 | 0 |

НОД (80, 20) = 4 .

**Пример 3**

Найдем НОД (81, 20) .

| $q$ | $u_1$ | $u_2$ | $u_3$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|---|---|---|
| – | 1 | 0 | 81 | 0 | 1 | 20 |
| 4 | 0 | 1 | 20 | 1 | –4 | 1 |
| 20 | 1 | –4 | **1** | –20 | –79 | 0 |

НОД (81, 20) = 1 .

### 1.7.6. ОБОБЩЕННЫЙ АЛГОРИТМ ДЛЯ МОДУЛЬНЫХ ЧИСЕЛ

Необходимо определить три операции: деление (определение максимального целого числа, не превосходящего делимое), умножение и вычитание. Если они определены, то можно использовать алгоритм, описанный в предыдущем разделе.

**Пример**

Найдем $\text{НОД}\left[(3,12),(9,3)\right] = \text{НОД}\left[80_{10}, 20_{10}\right]$.

| $q$ | $u_1$ | $u_2$ | $u_3$ | $v_1$ | $v_2$ | $v_3$ |
|---|---|---|---|---|---|---|
| – | (1, 1) | (0, 0) | (3, 12) | (0, 0) | (1, 1) | (9, 3) |
| (4, 4) | (0, 0) | (1, 1) | **(9, 3)** | (1, 1) | (–4, –4) | (–33, 0) |

Число $(-33,0) \equiv (0,0)$, поэтому $\text{НОД}\left[(3,12),(9,3)\right] = (9,3) = 20_{10}$.

## 1.8. ВОЗВЕДЕНИЕ В СТЕПЕНЬ ПО МОДУЛЮ

Возведение в степень по модулю – это вычисление остатка от деления натурального числа $b$ (основание), возведенного в степень $e$ (показатель степени), на натуральное число $m$ (модуль). Обозначается как $c \equiv b^e \pmod m$.

Самый простой способ возведения в степень по модулю – это непосредственное вычисление числа $b^e$, а затем нахождение остатка от деления этого числа на $m$. Например, пусть $b = 5$, $e = 3$ и $m = 13$, тогда $5^3 = 125$ и остаток $c = 8$ от деления 125 на 13, т. е.

$$c \equiv 5^3 \pmod{13} \equiv 125 \pmod{13} = 8.$$

Возведение в степень по модулю может быть выполнено и с отрицательным показателем степени $e$. Д я этого необходимо найти число $d$, обратное числу $b$ по модулю $m$. Это легко сделать с помощью алгоритма Евклида. Таким образом, при отрицательной степени $e$

$$c \equiv b^e \pmod m \equiv d^{|e|} \pmod m \text{ и } bd \equiv 1 \pmod m.$$

Возвести в степень по модулю довольно легко даже при больших входных значениях. Однако вычисление дискретного логарифма, т. е. нахождение показателя степени $e$ при заданных $b$, $c$ и $m$, намного сложнее. Такое одностороннее поведение функции делает ее кандидатом для использования в криптографических алгоритмах.

В криптографических системах значения $b$ и $e$ увеличивают, чтобы добиться большего уровня безопасности, из-за чего значение $b^e$ становится большим. Для этого потребуется $e$ умножений и взятие остатка от деления огромного числа по модулю $m$. В криптографии нередко встречаются показатели степени $e > 1000$ бит.

Поэтому нужны методы, которые работают быстрее, чем простой алгоритм возведения в степень и взятия остатка. Приведем некоторые методы оптимизации этой задачи.

### 1.8.1. МЕТОД, ИСПОЛЬЗУЮЩИЙ ЧИСЛА МЕНЬШЕЙ ДЛИНЫ

В этом алгоритме используются числа меньшей длины. Поэтому, хотя количество действий и не сокращается, достигается увеличение быстродействия.

Алгоритм основывается на том, что для заданных $a$ и $b$ следующие сравнения эквивалентны:

$$c \equiv (ab) \pmod m \equiv \left[ a\left(b \pmod m\right)\right] \pmod m.$$

Это значит, что промежуточные результаты вычислений можно брать по модулю $m$. Поэтому можно использовать числа меньшей длины.

Отсюда следующий алгоритм для вычисления $c \equiv b^e \pmod m$.

1. Пусть $c = 1$, $n = 0$.
2. Увеличим $n$ на единицу.
3. Установим $c \equiv (bc) \pmod m$.
4. Если $n < e$, возвращаемся к шагу 2. В противном случае $c$ содержит правильный ответ: $c \equiv b^e \pmod m$.

При каждом проходе шага 3 выражение $c \equiv b^n \pmod m$ верно. После того как шаг 3 был выполнен $n$ раз, в $c$ будет искомое значение.

Например, пусть $b = 5$, $e = 3$ и $m = 13$ ($c \equiv 5^3 \pmod{13}$). Тогда

- $n = 1$, $c = (1 \cdot 5) \pmod{13} = 5 \pmod{13} = 5$;
- $n = 2$, $c = (5 \cdot 5) \pmod{13} = 25 \pmod{13} = 12$;
- $n = 3$, $c = (12 \cdot 5) \pmod{13} = 60 \pmod{13} = \mathbf{8}$.

Получим ответ «8», как и в предыдущем случае, но числа, используемые в этих расчетах, намного меньше ($5^3 = 125$ при простом возведении в степень; в данном примере максимальное число 60).

На псевдокоде это выглядит так:

```
function modular_pow(base, exponent, modulus)
    c := 1
    for n = 1 to exponent
        c := (c * base) mod modulus
    return c
```

### 1.8.2. МЕТОД ВЫЧИСЛЕНИЙ, ОСНОВАННЫЙ НА ПРЕДСТАВЛЕНИИ СТЕПЕНИ В ДВОИЧНОМ ВИДЕ

Разложим степень $e$ по степеням двоек (т. е. представим его в двоичном виде):

$$e = m_0 + 2m_1 + \ldots + 2^k m_k, \quad k = [0,1],$$

тогда

$$b^e \pmod m = b^{m_0} (b^2)^{m_1} (b^4)^{m_2} \ldots \left( b^{2^k} \right)^{m_k} \pmod m.$$

Используя результаты, полученные в предыдущем разделе, все промежуточные результаты тоже будем хранить по модулю $m$. Тогда можно предложить следующий алгоритм вычисления $c \equiv b^e \pmod m$.

1. Положить $c = 1, \quad n = e, \quad x = b$.
2. Повторять, пока $n > 0$:
   2.1. Если $n \equiv 1 \pmod 2$, $c = cx \pmod m$;
   2.2. $x = x^2 \pmod m$;
   2.3. $n = n / 2$ (для этого можно сдвинуть биты числа $n$ на один разряд вправо).

В результате $c \equiv b^e \pmod m$.

**Пример**

Дано $c \equiv 5^3 \pmod{13} = 8$, $3_{10} = 11_2$.

1. $c = 1, \quad n = 3, \quad x = 5$;

2.1. $c \equiv 1 \cdot 5 \,(\mathrm{mod}13) = 5$;

2.2. $x = 5^2 \,(\mathrm{mod}13) = 12$;

2.3. $n = n / 2 = 3 / 2 = 1$;

2.1. $c \equiv 5 \cdot 12 \,(\mathrm{mod}13) = 8$;

2.3. $n = n / 2 = 1 / 2 = 0$. Поскольку $n = 0$, алгоритм завершен.

В результате $c \equiv b^e \,(\mathrm{mod}\, m) = 5^3 \,(\mathrm{mod}\, 13) = 8$.

Возведение в степень по модулю – важная операция, которая часто используется в системах шифрования.

# 2. ИСТОРИЯ РАЗРАБОТКИ КОМПЬЮТЕРОВ, ОСНОВАННЫХ НА МОДУЛЬНОЙ АРИФМЕТИКЕ

Компьютеры с традиционной архитектурой фон Неймана работают с ограниченным размером машинного слова, который определяет максимальный размер числа. Процессоры компьютеров начала 1970-х годов оперировали числами размером в 8 двоичных разрядов (бит), затем размер машинного слова увеличился до 32 бит, в последнее время наиболее распространены процессоры с разрядностью 64 бита.

Кажется, что в 64 двоичных разряда (около 20 десятичных разрядов) укладываются достаточно большие числа. Действительно, максимальное положительное число, которое можно записать с помощью 64 двоичных разрядов, равно $(2^{64}-1)$, или 18 446 744 073 709 551 615. В этом числе 20 десятичных знаков.

Однако многие алгоритмы требуют намного большей разрядности. Наглядным примером являются популярные схемы шифрования. Алгоритм RSA считается достаточно безопасным при длине ключа 2048 бит, что вызывает определенные затруднения при реализации его на 32- и 64-битных машинах.

В 1955 году Миро Валах и Антонин Свобода опубликовали работу [22], в которой предложили для кодирования целых чисел в машинах с целью распараллеливания математических расчетов использовать модульную арифметику, или систему остаточных классов (СОК, англ. residue number system, – система счисления, основанная на модульной арифметике). Перспективная идея привела к большому объему научных публикаций. А. Свобода и М. Валах исследовали и опубликовали базовые принципы модулярной арифметики; Г. Айкен и У. Симон (США, 1957–1959) доказали ее преимущества для параллельного выполнения основных машинных арифметических операций; Х. Л. Гарнер (Канада, 1959) показал возможность арифметической самокоррекции для оперативного контроля вычислений с использованием модульных кодов; П. Чейни (Англия, 1959–1961) сконструировал нашедший широкое применение цифровой коррелятор на модульных принципах; Р. Танака (Япония) и Э. Шапиро (США) в 1962–1964 годах исследовали принципы организации параллельных вычислений и способы введения знака числовой величины в модульном представлении. К сожалению, часть работ была закрытой.

Разработанная в 1958–1962 годах в Чехословакии ламповая ЭВМ ЭПОС (Elektronický POčítací Stroj, EPOS) была первой в мире модульной ЭВМ. Основным автором концепции ЭВМ ЭПОС был Антонин Свобода.

Эта мультипрограммная (до пяти программ) ЭВМ с режимом разделения времени и ферритовым ОЗУ емкостью 1024 65-разрядных слов выполняла 5–20 тыс. операций в секунду (оп/с) над десятичными 12-разрядными операндами. В 1963 и 1964 годах было изготовлено два экземпляра ЭПОС. В 1960–1965 годах разработали более совершенную и экономичную транзисторную версию ЭВМ – ЭПОС-2 с производительностью до 40 тыс. оп/с (рис. 2.1).

Два варианта ЭПОС-2 под названиями ZPA-600 и ZPA-601 серийно выпускались до 1973 года, всего было выпущено 30 комплектов ЭВМ, что по тем временам, особенно для маленькой Чехословакии, было много.

*Рис. 2.1.* Первые в мире модульные ЭВМ ЭПОС и ЭПОС-2

В 1964 году А. Свобода эмигрировал в США. В 1968 году он получил премию IEEE за «вклад в логическое проектирование, механическое проектирование и фундаментальный труд по системам счисления в остаточных классах».

В СССР первым в конце 1950-х годов на создание компьютеров на основе модульной арифметики обратил внимание Федор Викторович Лукин. По закрытым каналам поступила информация об этих работах в США, которая заинтересовала И. Я. Акушского и Д. И. Юдицкого, ставших впоследствии основоположниками модульной арифметики в СССР. Информация о работах в США и вызвала запоздалый интерес к статье, вышедшей в Праге еще в 1955 году [23].

В 1960–1963 годах коллектив под руководством Д. И. Юдицкого создал первую в стране модульную ЭВМ – Т340А (К340А). Эта ЭВМ (рис. 2.2) была освоена в серийном производстве и стала базовой для разрабатываемых в те годы РЛС. Это первая в мире ЭВМ с быстродействием более 1 млн оп/с. Такую производительность удалось получить благодаря применению модульной арифметики. Только суперкомпьютер БЭСМ-6, созданный пять лет спустя, превзошел порог в 1 млн оп/с. Всего было выпущено около 50 комплектов Т340А/К340А.

*Рис. 2.2.* ЭВМ К340А: 20 шкафов, установленных в три ряда, инженерный пульт и регистратор

В 1968 году ОКБ «Вымпел» и Центр микроэлектроники заключили договор на разработку высокопроизводительной ЭВМ 5Э53. Главным конструктором был назначен Д. И. Юдицкий.

Применение модульной арифметики обеспечивало два основных бесспорных преимущества: повышенную производительность и простоту аппаратной реализации, а также повышенную надежность системы. Разработка 5Э53 была проведена в рекордно короткий срок, причем с изготовлением экспериментального образца. В ходе разработки продолжались теоретические исследования с целью совершенствования методов обработки информации в СОК. Операции умножения, деления, определение знака и тому подобные к тому времени не имели удовлетворительных алгоритмов решения в СОК. В результате напряженной работы алгоритмы были разработаны и реализованы в проекте. В начале 1971 года разработка документации была завершена (она включала в себя 97 272 листа), проведены все необходимые испытания, изготовлен и испытан макетный образец 5Э53 (рис. 2.3), который представлял собой 8-процессорный комплекс (4 модулярных и 4 двоичных процессора). Производительность ЭВМ была около 40 млн оп/с. Архитектура 5Э53 отличалась от классической в те годы «фон-неймановской» и имела много принципиально новых элементов. Так, команды разделялись на арифметические и управленческие. Первые выполнялись на модульных процессорах, вторые – на традиционных двоичных.

В 1960–1970-х годах в связи с разработками ЭВМ К340А и 5Э53 проводились серьезные научные исследования в области модульной арифметики и было много публикаций на эту тему в открытой печати. На основе методов модульной арифметики

И. Я. Акушским, В. М. Амербаевым и их учениками были разработаны методы проведения вычислений с числами в сотни тысяч разрядов. Это определило подходы к решению ряда вычислительных задач теории чисел, остававшихся нерешенными со времен Эйлера, Гаусса и Ферма [24].

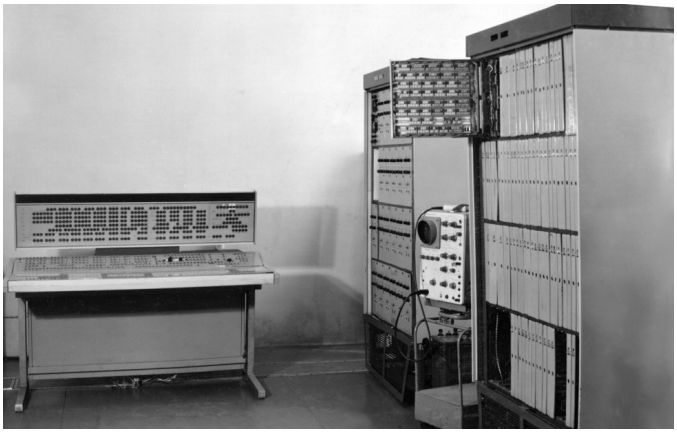


*Рис. 2.3.* Фрагмент экспериментального образца суперЭВМ 5Э53

В 1970–1971 годах большой интерес к модульной арифметике проявили банковские структуры США, которым требовались высокопроизводительные средства для высоконадежных вычислений с самокоррекцией – именно этим и характерна модульная арифметика. По данным открытой печати, они оценили результаты работы И. Я. Акушского и Д. И. Юдицкого как передовые в мире и обратились в МЭП с предложением о закупке модульных алгоритмов (предложили около 20 млн долларов США). Начавшиеся переговоры были пресечены Комитетом государственной безопасности СССР (КГБ).

В 1972 году финансирование работ по 5Э53 было прекращено. В результате аппаратных интриг разработка перспективного проекта суперкомпьютера 5Э53 был остановлена. Тем самым было фактически закрыто новое перспективное направление развития вычислительной техники, превосходящее всё имевшееся и в стране, и за рубежом, – модульная арифметика. Прекращение работ по 5Э53 вызвало определенный психологический шок у сторонников модульной арифметики, их научная активность существенно снизилась, число открытых публикаций резко сократилось. Этот факт был замечен зарубежными учеными, сделавшими вывод о засекречивании этих работ в СССР (истинных причин они не знали). В США последовали этому «примеру» и засекретили у себя в стране работы по модульной арифметике.

В дальнейшем модульной арифметикой в нашей стране занимались в теоретическом плане только в учебных и академических институтах.

# 3. АЛГОРИТМЫ ШИФРОВАНИЯ, ОСНОВАННЫЕ НА МОДУЛЬНОЙ АРИФМЕТИКЕ

Проблема сокрытия содержания текста при его передаче волновала людей с древних времен. Чаще всего шифруются тексты документов, но в последнее время шифрованию подвергаются также изображения, голосовые данные и даже тексты программ.

По тому, используют ли алгоритмы шифрования и дешифрования одинаковые ключи, различают следующие виды шифрования:

- симметричные;
- асимметричные.

При симметричном шифровании обе стороны используют один и тот же ключ для шифрования и расшифровки сообщения. При асимметричном шифровании используется два ключа: один для шифрования, другой для расшифровки. При шифровании используется открытый ключ получателя. Он предназначен только для шифрования, и его открытость не причиняет вреда. У получателя сообщения есть закрытый ключ, который используется для расшифровки полученных сообщений.

Известно, что еще Цезарь (100–44 годы до н. э.) при переписке использовал шифр, основанный на модульной арифметике. В шифре Цезаря ключи представляют собой число символов, на которое сдвигаются буквы алфавита. При шифровании вместо каждой буквы открытого текста ставится буква, отстоящая от нее правее на число букв, задаваемое значением ключа. Расшифровка заключается в смещении каждой буквы левее на число букв, задаваемое тем же значением ключа, которое использовалось при шифровании.

В общем случае если $x$ обозначает позицию буквы в алфавите, которую мы хотим зашифровать, то формула выглядит следующим образом:

$$C(x) \equiv (x + k) \pmod{n}, \tag{3.1}$$

где $n$ – длина алфавита; $k$ – ключ, используемый в данном шифре.

Например, для латинского алфавита с ключом $k = 3$ выражение (3.1) примет вид

$$C(x) \equiv (x + 3) \pmod{26}. \tag{3.2}$$

Пример шифра Цезаря приведен в табл. 3.1.

Таблица 3.1

**Шифр Цезаря с $k = 3$**

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

Таким образом, слово CAESAR зашифруется в слово FDHVAU.

Для расшифровки используется следующее выражение:

$$x \equiv (C(x) - k) \pmod{n}. \tag{3.3}$$

К сожалению, шифр Цезаря обладает низкой стойкостью: в нем в случае русского алфавита возможны всего 32 различных ключа.

В аффинном шифре при помощи модульной арифметики для каждого числа, соответствующего букве исходного алфавита, вычисляется новое число:

$$C(x) \equiv (ax + k) \pmod{n}, \tag{3.4}$$

где $n$ – длина алфавита; пара $a$ и $k$ – ключ шифра.

Значение $a$ должно быть выбрано таким образом, чтобы $a$ и $n$ были взаимно простыми. Для расшифровки необходимо выполнить следующую операцию:

$$x \equiv a^{-1}(C(x) - k) \pmod{n}, \tag{3.5}$$

где $a^{-1}$ – обратное к $a$ число по модулю $n$, т. е.

$$1 \equiv aa^{-1} \pmod{n}. \tag{3.6}$$

Обратное к $a$ число существует только в том случае, когда $a$ и $n$ – взаимно простые.

Из выражений (3.3) и (3.1) видно, что шифр Цезаря – аффинный с $a = 1$.

Для шифрования сообщений на русском языке (т. е. $n = 33$) существует всего 20 чисел, взаимно простых с числом 33 и меньших 33 (а это и есть возможные значения $a$). Каждому значению $a$ могут соответствовать 33 разных дополнительных сдвига (значение $k$), т. е. всего существует $20 \cdot 33$, или 660 возможных ключей.

Аналогично для сообщений на английском языке (т. е. $n = 26$) всего существует $12 \cdot 26$, или 312 возможных ключей.

Сложность алгоритмов шифрования зависит от выбранных в качестве ключей взаимно простых чисел. В следующем разделе рассмотрим, как можно определить максимальное простое число.

## 3.1. НАХОЖДЕНИЕ ПРОСТЫХ ЧИСЕЛ

Наиболее известным методом получения простых чисел является решето Эратосфена (276–194 годы до н. э.).

### 3.1.1. РЕШЕТО ЭРАТОСФЕНА

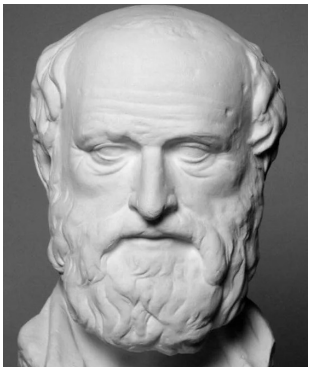
Этот метод описан во «Введении в арифметику» Никомаха Герасского. Сам Никомах называет автором этого метода Эратосфена Киренского. Название «решето» метод получил потому, что во времена Эратосфена писали числа на дощечке, покрытой воском, и прокалывали дырочки в тех местах, где были написаны составные числа. Поэтому дощечка являлась неким подобием решета, через которое «просеивались» все составные числа, а оставались только числа простые.

Эратосфен Киренский (276–194 годы до н. э.) – древнегреческий ученый, математик, астроном, первый в истории картограф, директор Александрийской библиотеки в Египте.

Эратосфен-математик известен как автор *решета Эратосфена*, алгоритма нахождения простых чисел.

Как географ, Эратосфен впервые оценил длину дуги александрийского меридиана, т. е. фактически определил длину окружности земного шара. Радиус Земли, по Эратосфену, составляет 6287 км. Фактический радиус – 6371 км.

В старости у Эратосфена воспалились глаза, что в дальнейшем привело к слепоте. Невозможность читать и наблюдать за природой сильно угнетала его, и в 194 году до н. э. он принял решение умереть от голода.



Для нахождения всех простых чисел не больше заданного числа $n$, следуя методу Эратосфена, нужно выполнить следующие шаги.

1. Выписать подряд все целые числа от двух до $n$ (2, 3, 4, …, $n$).
2. Пусть переменная $p$ изначально равна двум – первому простому числу.
3. Зачеркнуть в списке числа от $2p$ до $n$ с шагом $p$ (это будут числа, кратные $p$: $2p$, $3p$, $4p$, …).
4. Найти первое незачеркнутое число в списке, которое больше, чем $p$, и присвоить его значение переменной $p$.
5. Повторять шаги 3 и 4, пока возможно.

Теперь все незачеркнутые числа в списке – это все простые числа от двух до $n$. На практике алгоритм можно улучшить следующим образом. На шаге 3 числа можно зачеркивать, начиная с числа $p^2$, потому что все меньшие числа, кратные $p$,

имеют простой делитель меньше $p$, а они уже зачеркнуты к этому времени. Соответственно останавливать алгоритм можно, когда $p^2$ станет больше, чем $n$. Все простые числа (кроме числа 2) – нечетные числа, и поэтому для них можно выполнять зачеркивание с шагом $2p$, начиная с $p^2$.

Например, для $n = 33$:

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

Первое число в списке – 2, простое. Пройдем по ряду чисел, зачеркивая все числа, кратные двум (т. е. каждое второе, начиная с $2^2 = 4$):

2 3 x 5 x 7 x 9 x 11 x 13 x 15 x 17 x 19 x 21 x 23 x 25 x 27 x 29 x 31 x 33

Следующее незачеркнутое число – 3, простое. Пройдем по ряду чисел, зачеркивая все числа, кратные трем (т. е. каждое третье, начиная с $3^2 = 9$):

2 3 x 5 x 7 x x x 11 x 13 x x x 17 x 19 x x x 23 x 25 x x x 29 x 31 x x

Следующее простое число – 5. Пройдем по ряду чисел, зачеркивая все числа, кратные пяти (т. е. каждое пятое, начиная с $5^2 = 25$). Следующее незачеркнутое число – 7. Его квадрат равен 49, он больше 30, и на этом работа завершена. Все составные числа уже зачеркнуты:

2 3 5 7 11 13 17 19 23 29 31

Главной проблемой реализации решета Эратосфена на вычислительных машинах является не количество выполняемых операций, а требования по объему занимаемой памяти.

Согласно теореме Евклида, число простых чисел бесконечно. В настоящее время самыми большими простыми найденными числами являются числа Мерсенна.

### 3.1.2. ЧИСЛА МЕРСЕННА

Числа $M_p = 2^p - 1$, где $p$ – простое число, называются числами Мерсенна в честь французского математика Марена Мерсенна.

Марен Мерсенн – французский математик, физик, богослов (1588–1648).

Мерсенн один из первых оценил скорость звука. Он описал схему зеркального телескопа, позднее реализованную Ньютоном.

На протяжении первой половины XVII века был по существу координатором научной жизни Европы, ведя активную переписку практически со всеми видными учеными того времени. В числе его 78 корреспондентов были Декарт, Галилей, Кавальери, Паскаль, Роберваль, Торричелли, Ферма, Гюйгенс, Гассенди и многие другие.

Особенно важным общение с Мерсенном было для Декарта и Ферма. Об открытиях Ферма мы знаем практически только из его переписки с Мерсенном.

Мерсенн известен более всего как исследователь *чисел Мерсенна*.

Не все числа $M_p$ являются простыми. Однако среди них простые числа встречаются чаще. В 1772 году Эйлер доказал, что число $M_{31} = 2^{31} - 1 = 2\,147\,483\,647$ является простым.

Определение того, является ли данное число простым, – достаточно сложная задача. Только в 2002 году было доказано, что она полиномиально разрешима. Теоретически обоснованный детерминированный алгоритм практически непригоден ввиду его большой сложности. Поэтому в криптографии с открытым ключом, где используются простые числа порядка $10^{300}$, простоту по-прежнему определяют с помощью эффективных вероятностных тестов, таких как тест Миллера – Рабина.

Однако если утверждается, что число простое, то это должно быть строго доказано. В случае чисел Мерсенна существуют алгоритмы, позволяющие достаточно быстро определить, простое число или нет. Быстрейшим из известных тестов простоты является тест Люка – Лемера для чисел Мерсенна (предложен Эдуардом Люка в 1878 году и доказан Лемером в 1930 году), реализованный с использованием быстрого преобразования Фурье. Поэтому самыми большими из найденных простых чисел являются числа Мерсенна.

Тест основывается на следующем критерии простоты чисел Мерсенна.

Пусть $p$ – простое нечетное число. Число Мерсенна $M_p = 2^p - 1$ простое тогда и только тогда, когда оно делит нацело $(p-1)$-й член последовательности 4, 14, 194, 37 634, ..., задаваемой рекуррентно:

$$S_k = \begin{cases} 4 & k = 1, \\ S_{k-1}^2 - 2 & k > 1 \end{cases}.$$

Для проверки простоты $M_p$ последовательность чисел $S_1, S_2, \ldots, S_{p-1}$ вычисляется по модулю числа $M_p$ (т. е. вычисляются не сами числа $S_k$, длина которых растет экспоненциально, а остатки от деления $S_k$ на $M_p$, длина которых ограничена $p$ битами). Последнее число в этой последовательности $S_{p-1}(\mathrm{mod}\, M_p)$ называется вычетом Люка – Лемера.

Таким образом, число Мерсенна $M_p$ является простым тогда и только тогда, когда число $p$ – нечетное простое и вычет Люка – Лемера равен нулю. Сам алгоритм можно записать в виде следующего псевдокода.

```
►Вход: простое нечетное число p
    S = 4;   k = 1;  M = 2^p - 1
    while ( k != p - 1)
      {
          S = ((S * S) - 2) mod M;
          k += 1;
      }
    if ( S == 0 )  Возвратить ПРОСТОЕ;
     else          Возвратить СОСТАВНОЕ;
```

Например, число $M_{13} = 2^{13} - 1 = 8191$ является простым, поскольку $S_{12} = 128^2 - 2 \,(\mathrm{mod}\, 8191) = 0$, а число $M_{11} = 2^{11} - 1 = 2048$ составным, поскольку $S_{10} = 282^2 - 2 \,(\mathrm{mod}\, 2047) \neq 0$.

В алгоритме две основные операции: возведение в квадрат (умножение $S$ на $S$) и вычисление остатка при делении на модуль.

Эффективный алгоритм деления по модулю числа Мерсенна на компьютере дает следующая теорема.

**Теорема**

Для целого числа $x$ и числа Мерсенна $M_p = 2^p - 1$ имеет место сравнение по модулю

$$x \equiv \left( x \,(\mathrm{mod}\, 2^p) \right) + \left\lfloor \frac{x}{2^p} \right\rfloor (\mathrm{mod}\, M_p),$$

причем умножение на $2^k$ по модулю $M_p$ равносильно левому циклическому сдвигу на $k$ бит (если $k < 0$, то сдвиг осуществляется в обратную сторону).

Операция $\lfloor x \rfloor$ – округление вещественного числа до меньшего целого. Например, $\lfloor 3,9 \rfloor = 3$.

Таким образом, чтобы вычислить остаток от деления числа $x = 916$ на $M_5 = 2^5 - 1 = 31$, нужно исходное число преобразовать в двоичный вид: $916 = 1110010100_2$ и согласно теореме разбивать *x* на две части каждый раз, когда оно превосходит $M_5$:

$$916 (\mathrm{mod}\, 2^5 - 1) \equiv 916 (\mathrm{mod}\, 2^5) + \left\lfloor \frac{916}{2^5} \right\rfloor (\mathrm{mod}\, 2^5 - 1) \equiv$$

$$\equiv 10100_2 + 11100_2 \ (\mathrm{mod}\, 2^5 - 1) \equiv 110000_2 \ (\mathrm{mod}\, 2^5 - 1) \equiv$$

$$\equiv 10000_2 + 1_2 \ (\mathrm{mod}\, 2^5 - 1) \equiv 10001_2 \ (\mathrm{mod}\, 2^5 - 1) =$$

$$= 10001_2 = 17.$$

Для умножения чисел большой разрядности разработано достаточно много алгоритмов (более быстрых, чем умножение в столбик) [12]. Наиболее быстрые алгоритмы основаны на быстром преобразовании Фурье, например, метод умножения Шенхаге – Штрассена [27] или еще более быстрый алгоритм Фюрера [28]. В 2008 году Аниндая Де, Шэнден Саха, Пьюш Курур и Рампрасад Саптхариши построили похожий алгоритм, основанный на модульной, а не на комплексной арифметике, достигнув при этом такого же времени работы [28].

### 3.1.3. ПРОЕКТ РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ ПО ПОИСКУ ПРОСТЫХ ЧИСЕЛ МЕРСЕННА – GIMPS

К концу XX века стало понятно, что индивидуальный поиск зашел в тупик: слишком велика стала размерность чисел и слишком большим – их разброс. В 1995 году программист Джордж Уолтман (George Woltman) создал проект распределенных вычислений GIMPS (Great Internet Mersenne Prime Search). Теперь каждый участник проекта мог быть уверен, что проверяет никем еще не изученный диапазон значений. Это добровольный проект разработчиков, которые используют доступное программное обеспечение для поиска простых чисел Мерсенна.

В настоящее время известно 51 простое число Мерсенна, 17 наиболее больших чисел были обнаружены в рамках проекта GIMPS.

Существуют денежные призы за нахождение простых чисел. Найденное GIMPS в августе 2008 года число $2^{43\,112\,609}-1$ содержит 12 978 189 десятичных цифр, что позволило участникам проекта получить премию в 100 000 долларов США. Объявлены премии в 150 000 и 250 000 долларов США за нахождение простых чисел соответственно из более чем 100 миллионов и одного миллиарда десятичных цифр.

В 2018 году было обнаружено число $2^{82\,589\,933}-1$. Оно было найдено Патриком Ларошем в рамках проекта GIMPS и содержит 24 862 048 десятичных цифр. Это наибольшее известное простое число.

## 3.2. ФАКТОРИЗАЦИЯ ЦЕЛЫХ ЧИСЕЛ

> Проблема различения простых и составных чисел и разложения последних в их главные факторы является одной из самых важных и полезных в арифметике.
>
> *Карл Фридрих Гаусс*
> «Арифметические исследования» (1801)

Факторизацией натурального числа называется его разложение в произведение простых множителей (от англ. factor – множитель). Целью факторизации является приведение составного числа к произведению простых чисел.

Существование и единственность (с точностью до порядка следования множителей) такого разложения следует из основной теоремы арифметики. Основная теорема арифметики утверждает, что каждое натуральное число $n$ больше единицы можно представить в виде $n = p_1 \cdot \ldots \cdot p_k$, где $p_i$ – простые числа, причем такое разложение единственное, если не учитывать порядок следования множителей.

Факторизация больших чисел предположительно является вычислительно сложной задачей. В настоящее время неизвестно, существует ли эффективный алгоритм факторизации целых чисел. Однако доказательства того, что не существует решения этой задачи, также нет. Это предположение лежит в основе широко используемых алгоритмов шифрования (например, алгоритма RSA). Для взлома ключа RSA требуется разложить на множители число $m = pq$, где $p$, $q$ – большие простые числа.

В конкурсном списке RSA [30] приводятся большие числа, за факторизацию которых объявлены премии.

Любое простое число, не равное двум, является нечетным. Существуют признаки делимости на различные простые числа. Чтобы число делилось на 3, достаточно, чтобы сумма его цифр делилась на 3. Чтобы число делилось на 5, нужно, чтобы

последняя цифра была 0 или 5. Кроме этих чисел известны признаки делимости на 7, 11, 13, 17, 19, 23, 27, 29, 31, 37, 41, 59, 79, 99, 101, 1091 (https://ru.wikipedia.org/wiki/Признаки делимости). Эти признаки можно использовать, чтобы выявить заведомо составные числа.

В настоящее время самыми эффективными алгоритмами факторизации являются вариации решета числового поля [31, 33].

### 3.2.1. МЕТОД ФАКТОРИЗАЦИИ ФЕРМА

Метод факторизации Ферма – это алгоритм факторизации (разложения на множители) нечетного целого числа *n*, предложенный Пьером Ферма (Pierre de Fermat) в 1643 году. Метод основан на поиске таких целых чисел *x* и *y*, которые удовлетворяют соотношению $x^2 - y^2 = n$, что ведет к разложению $n = (x - y)(x + y)$.

Метод Ферма основан на теореме о представлении числа в виде разности двух квадратов.

**Теорема**
Если $n > 1$ нечетно, то существует взаимно однозначное соответствие между разложением на множители $n = ab$ и представлением в виде разности квадратов $n = x^2 - y^2$ с $x > y > 0$, задаваемое формулами $x = \frac{a+b}{2}$, $y = \frac{a-b}{2}$, $a = x + y$, $b = x - y$.

*Доказательство*
Если задана факторизация $n = ab$, то имеет место соотношение

$$n = ab = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2.$$

Таким образом, получается представление в виде разности двух квадратов. Наоборот, если дано $n = x^2 - y^2$, то правую часть можно разложить на множители: $n = (x - y)(x + y)$.

Для разложения на множители нечетного числа *n* ищется пара чисел $(x, y)$ таких, что $x^2 - y^2 = n$ или $(x - y)(x + y) = n$. При этом числа $(x - y)$ и $(x + y)$ являются делителями *n*, возможно, тривиальными (т. е. одно из них равно единице, а другое – *n*).

В нетривиальном случае равенство $x^2 - y^2 = n$ равносильно $x^2 - n = y^2$.

Для всякого заданного числа *n* достаточно испытать в качестве возможных множителей простые числа, меньшие чем $\sqrt{n}$. Действительно, если у числа есть множитель $m > \sqrt{n}$, то ему соответствует (как результат деления *n* на *m*) и некоторый множитель, меньший чем $\sqrt{n}$.

Алгоритм:

• поиск квадрата *x* начинается с $x = \lceil \sqrt{n} \rceil$ – наименьшего числа, при котором разность $x^2 - n$ неотрицательна (операция $\lceil x \rceil$ – округление вещественного числа до большего целого. Например, $\lceil 3{,}1 \rceil = 4$);

• для каждого целого *k*, начиная с $k = 1$, вычисляют $\left( \lceil \sqrt{n} \rceil + k \right)^2 - n$ и проверяют, не является ли это число точным квадратом. Если не является, то *k* увеличивают на единицу и переходят на следующую итерацию;

• если $\left( \lceil \sqrt{n} \rceil + k \right)^2 - n$ является точным квадратом, то *x* найдено. Если оно является тривиальным и единственным, то *n* – простое число.

Для ускорения вычислений, чтобы не возводить в квадрат, значение выражения на шаге $(k + 1)$ вычисляется с учетом значения на шаге *k*:

$$(s+1)^2 - n = s^2 + 2s + 1 - n, \quad \text{где } s = \lceil \sqrt{n} \rceil + k.$$

**Пример 1**

Пусть $n = 10\,873$. Тогда $\sqrt{n} \approx 104{,}27$ и $s = \lceil \sqrt{n} \rceil = 105$. Для $k = 0, 1, 2, \ldots$ будем вычислять значения функции $x = (s + k)^2 - n$ и определим, при каком *k* будет целое значение $\sqrt{x}$.

| $k$ | $q(k) = (s+k)^2 - n$ | $\sqrt{x}$ |
|---|---|---|
| 1 | 363 | 19,052 |
| 2 | 576 | **24** |

На втором шаге $(k = 2)$ было получено целое значение $\sqrt{x} = 24$.

Таким образом, $(105 + 2)^2 - n = 24^2$ и $n = 10\,873 = (107 + 24)(107 - 24) = 131 \cdot 84$.

Если один из множителей не является простым, то его тоже необходимо разложить на множители. Процесс остановится, когда все множители будут простыми.

**Пример 2**

Пусть $n = 43$. Тогда $\sqrt{n} \approx 6{,}557$ и $s = \lceil \sqrt{n} \rceil = 7$. Для $k = 0, 1, 2, \ldots$ будем вычислять значения функции $x = (s + k)^2 - n$ и определим, при каком $k$ будет целое значение $\sqrt{x}$.

| $k$ | $q(k) = (s + k)^2 - n$ | $\sqrt{x}$ |
|---|---|---|
| 1 | $(7 + 1)^2 - 43 = 21$ | 4,58 |
| 2 | 38 | 6,16 |
| 3 | 57 | 7,549 |
| 4 | 78 | 8,831 |
| 5 | 101 | 10,049 |
| 6 | 126 | 11,224 |
| 7 | 153 | 12,369 |
| 8 | 182 | 13,49 |
| 9 | 213 | 14,594 |
| 10 | 246 | 15,684 |
| 11 | 281 | 16,763 |
| 12 | 318 | 17,832 |
| 13 | 357 | 18,894 |
| 14 | 398 | 19,949 |
| 15 | 442 | 21 |

На шаге $k = 15$ было получено целое значение $\sqrt{x} = 21$.

Таким образом, $(7+15)^2 - n = 21^2$ и $n = 43 = (22+21)(22-21) = 43 \cdot 1$. Это значит, что $n = 43$ является целым числом.

При определении, является ли число $\sqrt{x}$ целым, необходимо вычислить квадратный корень из большого целого числа. Можно использовать для этого алгоритм Ньютона, но это очень медленная операция. Поэтому для повышения производительности рассмотрим следующий алгоритм вычисления целого квадратного корня от натурального числа.

Алгоритм для нахождения целого $q = \sqrt{x}$.

1. Возьмем $x = n$.

2. $y = \left\lfloor \dfrac{x + \left\lfloor \dfrac{n}{x} \right\rfloor}{2} \right\rfloor$.

3. Если $y < x$, то $x = y$ и перейти к шагу 2.

4. $q = x$.

**Пример 3**

1. Возьмем $x = n = 129$.

2. $y = \left\lfloor \dfrac{129 + \left\lfloor \dfrac{129}{129} \right\rfloor}{2} \right\rfloor = 65.$

3. Если $y < x$, то $x = 65$ и перейти к шагу 2.

Последовательно вычисляем $y$, пока $y < x$:

$$y = \left\lfloor \frac{65 + \left\lfloor \frac{129}{65} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{65+1}{2} \right\rfloor = 33, \quad x = 33;$$

$$y = \left\lfloor \frac{33 + \left\lfloor \frac{129}{33} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{33+3}{2} \right\rfloor = 18, \quad x = 18;$$

$$y = \left\lfloor \frac{18 + \left\lfloor \frac{129}{18} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{18+7}{2} \right\rfloor = 12, \quad x = 12;$$

$$y = \left\lfloor \frac{12 + \left\lfloor \frac{129}{12} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{12+10}{2} \right\rfloor = 11, \quad x = 11;$$

$$y = \left\lfloor \frac{11 + \left\lfloor \frac{129}{11} \right\rfloor}{2} \right\rfloor = \left\lfloor \frac{11+11}{2} \right\rfloor = 11.$$

Так как $y = x$, останавливаем алгоритм.

Ответ: $\left\lfloor \sqrt{129} \right\rfloor = 11$.

Наибольшая эффективность расчета методом факторизации Ферма достигается в случае, когда множители числа $n$ примерно равны между собой. Если случайные числа $p$ и $q$, произведение которых равно $n$, близки друг другу, то они могут быть найдены методом факторизации Ферма. Поэтому в алгоритме криптоанализа RSA не выбираются близкие числа.

### 3.2.2. МЕТОД КРАЙЧИКА – ФЕРМА

В 1926 году Морис Крайчик (Maurice Kraitchik) предложил улучшение метода Ферма. Вместо чисел $x$ и $y$ таких, что $x^2 - y^2 = n$, он предложил искать числа, которые удовлетворяют сравнению $x^2 - y^2 \equiv 0 \pmod{n}$. Если $x$ несравнимо с $\pm y$,

то $n$ делится на $(x-y)(x+y)$, но не делится ни на $(x-y)$, ни на $(x+y)$, следовательно, НОД $(x-y, n)$ – нетривиальный делитель $n$ [32].

Алгоритм при этом такой же, как и в методе Ферма: брать $q(x)=x^2-n$ для разных $x$, но не ждать, когда получится целочисленный квадрат, а попытаться его построить.

Заметим, что если

$$q(x_1)\,q(x_2)\cdot\ldots\cdot q(x_k)=y^2,$$

то

$$(x_1^2-n)\,(x_2^2-n)\ldots(x_k^2-n)=y^2,$$

т. е.

$$x^2=x_1^2x_2^2\ldots x_k^2\equiv y^2 \pmod n.$$

**Пример**

Пусть $n=2041$. Тогда $\sqrt{n}\approx 45{,}177$ и $s=\lceil\sqrt{n}\rceil=46$.

Построим следующую таблицу ($k$ = 0, ... , 5).

| $x=s+k$ | $q(x)=x^2-n$ | $\sqrt{q(x)}$ |
|---|---|---|
| 46 | 75 | 8,66 |
| 47 | 168 | 12,96 |
| 48 | 263 | 16,21 |
| 49 | 360 | 18,97 |
| 50 | 459 | 21,42 |
| 51 | 560 | 23,66 |

По методу Ферма следовало бы продолжать вычисления, пока не был бы найден квадрат какого-либо целого числа, но мы попытаемся его построить.

М. Крайчик заметил, что многие из чисел $q(x) = x^2 - n$ раскладываются в произведение небольших простых чисел, т. е. являются гладкими. В теории чисел гладким числом называется целое число, все простые делители которого малы.

Можно заметить, что некоторые из $q(x)$ разложимы в произведение небольших простых чисел:

$q(1) = 75 = 3 \cdot 5^2$;

$q(2) = 168 = 2^3 \cdot 3 \cdot 7$;

$q(4) = 360 = 2^3 \cdot 3^2 \cdot 5$;

$q(6) = 560 = 2^4 \cdot 5 \cdot 7$.

Выберем множество небольших простых чисел и назовем его факторной базой. Например, $FB = \{2, 3, 5, 7\}$. Таким образом, выбранные нами числа являются гладкими относительно этой факторной базы. Когда факторная база выбрана, любое гладкое число можно представить вектором показателей степеней $v = (r_1, r_2, \ldots, r_k)$, $k$ – размер факторной базы.

Например, для факторной базы с размером $k$ = 4 число $q(6)$ представится как $560 = 2^4 \cdot 3^0 \cdot 5^1 \cdot 7^1$, или в виде вектора $v = (4, 0, 1, 1) \equiv (0, 0, 0, 0) \pmod 2$.

Теперь перемножаем некоторые гладкие функции так, чтобы найти полные квадраты. При умножении чисел их векторы показателей степени складываются. Чтобы произведение гладких чисел было полным квадратом, необходимо подобрать такую комбинацию сомножителей, чтобы сумма векторов показателей имела бы только четные составляющие. Например, для $75 \cdot 168 \cdot 360 \cdot 560 = 50\,400^2$ сумма векторов составит

$$(0, 1, 2, 0) + (3, 1, 0, 1) + (3, 2, 1, 0) + (4, 0, 1, 1) = (10, 4, 4, 2).$$

Поскольку сумма должна быть четной, вместо целочисленных составляющих векторов можно просто рассматривать их остатки по модулю 2. В этом случае произведение элементов является полным квадратом тогда, когда вектор суммы по модулю 2 всех векторов будет нулевым вектором.

Теперь можно вычислить пару чисел $(x, y)$:

$x = 46 \cdot 47 \cdot 49 \cdot 51 = 5\,402\,838 \equiv 311 \pmod{2041}$;

$y = 2^5 \cdot 3^2 \cdot 5^2 \cdot 7 = 50\,400 \equiv 1416 \pmod{2041}$.

Далее с помощью алгоритма Евклида находим делитель $n$ НОД (1416 – 311,2041) = 13. Таким образом, $2041 = 13 \cdot 157$.

В этом алгоритме непонятно, как определить гладкие числа. Выделение гладких чисел разложением на простые сомножители – достаточно трудоемкая задача.

Другой проблемой является нахождение гладких чисел, произведение которых будет полным квадратом.

Найти решение этих задач позволяют алгоритмы квадратичного решета.

### 3.2.3. КВАДРАТИЧНОЕ РЕШЕТО

Рекордсменом среди алгоритмов факторизации чисел до недавнего времени являлся алгоритм «квадратичное решето» (Quadratic Sieve algorithm, QS), предложенный в 1981 году Карлом Померанцем. Это универсальный алгоритм факторизации, так как время его выполнения зависит исключительно от размера факторизуемого числа, а не от его особой структуры и свойств.

Алгоритм почти 10 лет был лучшим (до предложенного в 1990 году метода решета числового поля SNFS (Special Number Field Sieve). Метод решета числового поля даже для базового описания требует достаточно сложных обоснований, в то же время основные идеи обоих методов совпадают и основаны на методе факторизации Ферма.

Алгоритм квадратичного решета работает в два этапа:

1) этап сбора данных, где он собирает информацию, которая может привести к равенству квадратов;

2) этап обработки данных, где он помещает всю собранную информацию в матрицу и обрабатывает ее для получения равенства квадратов.

Первый этап может быть легко распараллелен на много процессов, но второй этап требует больших объемов памяти, и его трудно распараллелить.

**Пример**

Пусть $n = 15\,347$. Тогда $\sqrt{n} \approx 123{,}88$ и $s = \lceil \sqrt{n} \rceil = 124$.

1. Первый этап – сбор данных.

Строим следующую таблицу ($k = 0, ..., 99$).

| $k$ | $x = s + k$ | $q(x) = x^2 - n$ |
|---|---|---|
| 0 | 124 | 29 |
| 1 | 125 | 278 |
| 2 | 126 | 529 |
| 3 | 127 | 782 |
| 4 | 128 | 1037 |
| 5 | 129 | 1294 |
| … | … | … |
| 99 | 223 | 34 282 |

В этой таблице найдем гладкие числа.

2. Второй этап – просеивание.

Выделение гладких чисел разложением на простые сомножители – достаточно трудоемкая задача, поэтому ее решают по-другому.

Выберем факторную базу. Поскольку $n = 15\,347$ мало, то возьмем только четыре простых числа. Первые четыре простых числа $p$, для которых у числа 15 347 есть квадратный корень по модулю $p$, равны 2, 17, 23 и 29. Другими словами, число 15 347 является квадратичным вычетом для этих простых чисел.

Целое число $a$ называется квадратичным вычетом по модулю $p$, если разрешимо сравнение $x^2 \equiv a \pmod{p}$.

Запишем систему сравнений:

$$x^2 \equiv 15\,347 \pmod 2, \quad x^2 \equiv 1 \pmod 2, \quad 1^2 \equiv 1 \pmod 2,$$

$$x^2 \equiv 15\,347 \pmod{17}, \quad x^2 \equiv 13 \pmod{17}, \quad 8^2 \equiv 13 \pmod{17}, \quad 9^2 \equiv 13 \pmod{17},$$

$$x^2 \equiv 15\,347 \pmod{23}, \quad x^2 \equiv 6 \pmod{23}, \quad 11^2 \equiv 6 \pmod{23}, \quad 12^2 \equiv 6 \pmod{23},$$

$$x^2 \equiv 15\,347 \pmod{29}, \quad x^2 \equiv 6 \pmod{29}, \quad 8^2 \equiv 11 \pmod{29}, \quad 21^2 \equiv 11 \pmod{29}.$$

Для каждого $p$ в выбранной нами факторной базе $\{2, 17, 23, 29\}$ решаем уравнение

$$(k+s)^2 - n \equiv 0 \pmod{p},$$

чтобы найти записи в массиве $q(x)$, которые делятся на $p$.

Для $p = 2$:

$$(k+124)^2 - 15\,347 \equiv 0 \pmod 2;$$

$$k \equiv \sqrt{15\,347} - 124 \equiv 1 \pmod 2.$$

Таким образом, начиная с $k = 1$ с шагом 2, каждая запись в таблице будет делиться на 2.

| $k$ | Значение |
|---|---|
| 0 | 29 |
| **1** | **139** |
| 2 | 529 |
| **3** | **391** |
| 4 | 1037 |
| **5** | **647** |
| … | … |
| **99** | **17 191** |

Аналогично для $\{17, 23, 29\}$.

Для $p = 17$: $k \equiv \sqrt{15\,347} - 124 \equiv 8 - 124 \equiv 3 \pmod{17}$;

$k \equiv \sqrt{15\,347} - 124 \equiv 9 - 124 \equiv 4 \pmod{17}$.

Для $p = 23$: $k \equiv \sqrt{15\,347} - 124 \equiv 11 - 124 \equiv 2 \pmod{23}$;

$k \equiv \sqrt{15\,347} - 124 \equiv 12 - 124 \equiv 3 \pmod{23}$.

Для $p = 29$: $k \equiv \sqrt{15\,347} - 124 \equiv 8 - 124 \equiv 0 \pmod{29}$;

$k \equiv \sqrt{15\,347} - 124 \equiv 21 - 124 \equiv 13 \pmod{29}$.

Обратим внимание, что для каждого $p$ из $\{17, 23, 29\}$ будет два уравнения из-за наличия двух квадратных корней по модулю.

В результате получим

| $k$ | Значение |
|---|---|
| **0** | **1** |
| **1** | **139** |
| **2** | **23** |
| **3** | **1** |
| **4** | **61** |
| **5** | **647** |
| … | … |
| **99** | **17 191** |

Элемент в таблице, который равен единице, соответствует гладкому числу по выбранной факторной базе. В следующей таблице это числа, соответствующие $k = 0, 3, 71$.

| $k$ | $x = s + k$ | $q(x) = x^2 - n$ | Разложение по факторной базе |
|---|---|---|---|
| 0 | 124 | 29 | $2^0 \cdot 17^0 \cdot 23^0 \cdot 29^1$ |
| 3 | 127 | 782 | $2^1 \cdot 17^1 \cdot 23^1 \cdot 29^0$ |
| 71 | 195 | 22678 | $2^1 \cdot 17^1 \cdot 23^1 \cdot 29^1$ |

Рассмотрим равенства:

$29 = 2^0 \cdot 17^0 \cdot 23^0 \cdot 29^1$;

$782 = 2^1 \cdot 17^1 \cdot 23^1 \cdot 29^0$;

$22678 = 2^1 \cdot 17^1 \cdot 23^1 \cdot 29^1$.

Выпишем показатели простых чисел факторной базы в виде матрицы и решим систему

$$S\cdot\begin{bmatrix}0&0&0&1\\1&1&1&0\\1&1&1&1\end{bmatrix}\equiv\begin{bmatrix}0&0&0&0\end{bmatrix}\ (\mathrm{mod}\ 2).$$

Ее решение

$$S=\begin{bmatrix}1&1&1\end{bmatrix}.$$

Таким образом, произведение всех трех уравнений есть квадрат $(\mathrm{mod}\ n)$ :

$$29\cdot 782\cdot 22\,678=22\,678^2 \text{ и } 124^2\cdot 127^2\cdot 195^2=3\,070\,860^2,$$

откуда $22\,768^2=3\,070\,860^2$.

Теперь вычисляем

НОД $(3\,070\,860-22\,678,15347)=103$ . Таким образом, $15\,347=103\cdot 149$ .

В этом примере на первом шаге при составлении таблицы мы забыли проверить корни, которые будут являться целыми числами.

Строим следующую таблицу.

| $k$ | $x=s+k$ | $q(x)=x^2-n$ | $\sqrt{q(x)}$ |
|---|---|---|---|
| 0 | 124 | 29 | 5,39 |
| 1 | 125 | 278 | 16,67 |
| 2 | 126 | 529 | 23 |

Видно, что при $k=2$ квадрат целого числа $\sqrt{q(x)}=23$ .

По алгоритму Ферма

$$(124+2)^2-15\,347=23^2 \text{ и } n=15\,347=(126+23)(126-23)=149\cdot 103.$$

Общий метод решета числового поля (англ. General Number Field Sieve, GNFS) является наиболее эффективным алгоритмом факторизации чисел длиной более 110 десятичных знаков. Метод является обобщением классического метода квадратичного решета числового поля. Если последний позволяет факторизовать числа

только некоторого специального вида, то общий метод работает на множестве целых чисел, за исключением степеней простых чисел (которые факторизуются тривиально извлечением корней).

В 1996 году с помощью алгоритма было получено разложение числа RSA-130. Позднее с помощью метода были факторизованы, например, числа RSA-140 [13] и RSA-155 [14].

9 мая 2005 года Ф. Бар, М. Бом, Й. Франке и Т. Клейнжунг объявили, что они разложили число RSA-200, используя общий метод решета числового поля.

В феврале 2010 года группе ученых из Швейцарии, Японии, Франции, Нидерландов, Германии и США удалось успешно вычислить данные, зашифрованные при помощи криптографического ключа стандарта RSA длиной 768 битов.

В феврале 2020 года Фабрис Будо, Пьеррик Годри, Аврора Гильевич, Надя Хенингер, Эммануэль Томе и Пол Циммерман выделили методом RSA-250 829-битное число с 250 десятичными знаками. Эта факторизация была завершена реализацией сита общего числового поля, запущенной на сотнях машин.

RSA-250 =

= 21403246502407449612644230728393335630086147151447550177977549208814180234471401366433455190958046796109928518724709145876873962619215573630474547705208051190564931066876915900197594056934574522305893259766974716817380693648946998715784949759374979 37

равно

641352894770715802787901901705773890848250147429434472081168596320245323446302386235987526683477087376619255856946397988533 67

умножить на

333720275949781565562260106053551142279407603447675546667845209870238417292100370802574486732968818775657189862580369320627 11.

Для своего времени метод квадратичного решета сыграл положительную роль. Однако, судя по результатам факторизации больших чисел, похоже, что этот метод в современное время себя исчерпал.

Поэтому интересны новые подходы при разработке данной задачи.

## 3.3. АЛГОРИТМ ШИФРОВАНИЯ RSA

Криптосистема RSA стала первой системой, пригодной для надежного шифрования, ее алгоритм используется в большом числе криптографических приложений. RSA – это криптографический алгоритм с открытым ключом, основанный на вычислительной сложности задачи факторизации больших целых чисел.

Идея асимметричной криптосистемы с открытым и закрытым ключом приписывается Уитфилду Диффи и Мартину Хеллману, которые опубликовали эту концепцию в 1976 году [40]. Они также ввели цифровые подписи и попытались применить теорию чисел. В их формулировке использовался секретный ключ с общим доступом, созданный путем экспоненциализации некоторого числа по модулю простого числа.

Рон Ривест, Ади Шамир и Леонард Адлеман из Массачусетского технологического института в течение года предприняли несколько попыток создать одностороннюю функцию, которую было бы трудно инвертировать. Алгоритм известен как RSA – это аббревиатура, составленная на основе инициалов их фамилий в том же порядке, что и в статье [34]. Статья была опубликована в журнале Communications of the ACM в феврале 1978 года.

Клиффорд Кокс, английский математик, работавший в британской разведывательной службе Government Communications Headquarters (GCHQ), описал эквивалентную систему во внутреннем документе в 1973 году. Однако, учитывая относительно дорогие компьютеры, необходимые для ее реализации в то время, она не нашла практического применения. Это открытие было обнародовано только в 1997 году из-за его секретности.

В августе 1977 года в колонке «Математические игры» Мартина Гарднера в журнале Scientific American с разрешения Рональда Ривеста [35] появилось первое описание криптосистемы RSA. Читателям также было предложено дешифровать английскую фразу, зашифрованную описанным алгоритмом:

| | | | |
|---|---|---|---|
| 9686 | 9613 | 7546 | 2206 |
| 1477 | 1409 | 2225 | 4355 |
| 8829 | 0575 | 9991 | 1245 |
| 7431 | 9874 | 6951 | 2093 |
| 0816 | 2982 | 2514 | 5708 |
| 3569 | 3147 | 6622 | 8839 |
| 8962 | 8013 | 3919 | 9055 |
| 1829 | 9451 | 5781 | 5154 |

В качестве открытых параметров системы были использованы числа $n$ = 1143816...6879541 (129 десятичных знаков, 425 бит, также известное как RSA-129) и $e$ = 9007. По заявлению Ривеста, для факторизации числа потребовалось бы более 40 квадриллионов лет. Однако чуть более чем через 15 лет, 3 сентября 1993 года, было объявлено о запуске проекта распределенных вычислений с координацией через электронную почту по нахождению сомножителей числа RSA-129 и решению головоломки. На протяжении полугода более 600 добровольцев из 20 стран жертвовали процессорное время 1600 машин. В результате были найдены простые множители и расшифровано исходное сообщение.

Криптографические системы с открытым ключом используют так называемые односторонние функции, которые обладают следующим свойством:

- если известен $x$, то $f(x)$ вычислить относительно просто;
- если известен $y = f(x)$, то для вычисления $x$ нет простого (эффективного) пути.

Под односторонностью понимается практическая невозможность вычислить обратное значение, используя современные вычислительные средства, за обозримый интервал времени.

В основу криптографической системы с открытым ключом RSA положена сложность задачи факторизации произведения двух больших простых чисел. Для шифрования используется операция возведения в степень по модулю большого числа. Для дешифрования (обратной операции) за разумное время необходимо уметь вычислять функцию Эйлера от данного большого числа, для этого необходимо знать разложение числа на простые множители.

В криптографической системе с открытым ключом каждый участник располагает как открытым ключом (англ. public key), так и закрытым (англ. private key). В криптографической системе RSA каждый ключ состоит из пары целых чисел. Каждый участник создает свой открытый и закрытый ключ самостоятельно. Закрытый ключ каждый из них держит в секрете, а открытые ключи можно сообщать кому угодно или даже публиковать их.

RSA-ключи генерируются следующим образом [36].

1. Выбираются два различных случайных простых числа $p$ и $q$ заданного размера (например, 1024 бита каждое).
2. Вычисляется их произведение $n = pq$, которое называется модулем.
3. Вычисляется значение функции Эйлера от числа $n$:

$$\varphi(n) = (p-1)(q-1).$$

4. Выбирается целое число $e$ $\left(1 < e < \varphi(n)\right)$, взаимно простое со значением функции $\varphi(n)$. Число $e$ называется открытой экспонентой; обычно в качестве $e$ берут простые числа, содержащие небольшое количество единичных битов в двоичной записи (например, простые из чисел Ферма: 17, 257 или 65 537), так как в этом случае время, необходимое для шифрования с использованием быстрого возведения в степень, будет меньше. Слишком малые значения $e$, например 3, потенциально могут ослабить безопасность схемы RSA.
5. Вычисляется число $d$, мультипликативно обратное числу $e$ по модулю $\varphi(n)$, т. е. число, удовлетворяющее сравнению

$$de \equiv 1 \left(\mathrm{mod}\ \varphi(n)\right).$$

Число $d$ называется секретной экспонентой, обычно оно вычисляется при помощи расширенного алгоритма Евклида.

6. Пара $(e, n)$ публикуется в качестве открытого ключа RSA (англ. RSA public key); сообщение шифруется открытым ключом:

$$c = E(m) = m^e \pmod{n}. \tag{3.3.1}$$

7. Пара $(d, n)$ играет роль закрытого ключа RSA (англ. RSA private key) и держится в секрете; расшифровать сообщение может только обладатель секретного ключа $d$:

$$m = D(c) = c^d \pmod{n}. \tag{3.3.2}$$

Корректность последнего утверждения основывается на теореме Эйлера и китайской теореме об остатках.

**Пример 1**

| Этап | Операция | Результат |
|---|---|---|
| Генерация ключей | Выбрать два простых различных числа | $p = 13,\ q = 17$ |
| | Вычислить произведение | $n = pq = 13 \cdot 17 = 221$ |
| | Вычислить функцию Эйлера | $\varphi(n) = (p-1)(q-1) = 192$ |
| | Вычислить открытую экспоненту | $e = 29$ |
| | Вычислить секретную экспоненту (обратную $e$) | $d \equiv e^{-1} \pmod{\varphi(n)}$ $\quad d = 53$ |
| | Опубликовать открытый ключ | $\{e, n\} = \{29, 221\}$ |
| | Сохранить закрытый ключ | $\{d, n\} = \{53, 221\}$ |
| Шифрование | Выбрать текст для шифрования | $m = 19$ |
| | Вычислить шифротекст | $c = E(m) = m^e \pmod{n} \equiv$ $\equiv 19^{29} \pmod{221} \equiv 15$ |
| Расшифровка | Вычислить исходное сообщение | $m = D(c) = c^d \pmod{n} \equiv$ $\equiv 15^{53} \pmod{221} \equiv 19$ |

**Пример 2**

| Этап | Операция | Результат |
|---|---|---|
| Генерация ключей | Выбрать два простых различных числа | $p = 3557, \quad q = 2579$ |
| | Вычислить произведение | $n = pq = 3557 \cdot 2579 = 9\,173\,503$ |
| | Вычислить функцию Эйлера | $\varphi(n) = (p-1)(q-1) = 9167368$ |
| | Вычислить открытую экспоненту | $e = 3$ |
| | Вычислить секретную экспоненту | $d \equiv e^{-1} \pmod{\varphi(n)} \quad d = 6\,111\,579$ |
| | Опубликовать открытый ключ | $\{e, n\} = \{3, 9173503\}$ |
| | Сохранить закрытый ключ | $\{d, n\} = \{6\,111\,579, 9\,173\,503\}$ |
| Шифрование | Выбрать текст для шифрования | $m = 111111$ |
| | Вычислить шифротекст | $c = E(m) = m^e \pmod{n} =$ $= 111111^3 \pmod{9\,173\,503} = 4\,051\,753$ |
| Расшифровка | Вычислить исходное сообщение | $m = D(c) = c^d \pmod{n} =$ $= 4\,051\,753^{6\,111\,579} \pmod{9\,173\,503} = 111111$ |

Данная схема на практике не используется по причине того, что она не является практически надежной. Действительно, односторонняя функция (3.3.1) $E(m)$ является детерминированной: при одних и тех же значениях входных параметров (ключа и сообщения) она выдает одинаковый результат. Это значит, что не выполняется необходимое условие практической надежности шифра.

Наиболее используемым в настоящее время является смешанный алгоритм шифрования, в котором сначала шифруется сеансовый ключ, а потом уже с его помощью участники шифруют свои сообщения симметричными системами. После завершения сеанса сеансовый ключ, как правило, уничтожается.

Алгоритм шифрования

1. Взять открытый ключ $(e, n)$.
2. Создать случайный сеансовый ключ $m$.
3. Зашифровать сеансовый ключ с использованием открытого ключа Алисы: $c = E(m) = m^e \pmod n$.
4. Зашифровать сообщение $M_A$ с помощью сеансового ключа симметричным алгоритмом:
$$C = E_m(M_A).$$

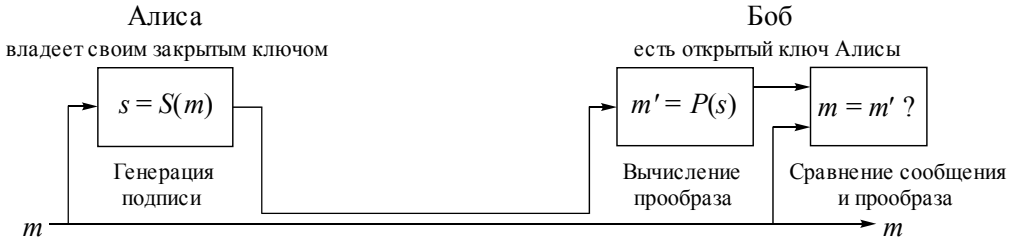
Алгоритм расшифровки

1. Принять зашифрованный сеансовый ключ $c$.
2. Взять свой закрытый ключ $(d, n)$.
3. Применить закрытый ключ для расшифровывания сеансового ключа: $m = D(c) = c^d \pmod n$.
4. Расшифровать сообщение $C$ с помощью сеансового ключа симметричным алгоритмом:
$$M_A = D_m(C).$$

Если сеансовый ключ больше, чем модуль $n$, то его разбивают на блоки нужной длины (при необходимости дополняют нулями) и шифруют каждый блок.

Система RSA может использоваться не только для шифрования, но и для цифровой подписи.

Предположим, что Алисе (стороне $A$) нужно отправить Бобу (стороне $B$) сообщение $m$, подтвержденное электронной цифровой подписью.




1. Взять открытый текст $m$.
2. Создать цифровую подпись $s$ с помощью своего секретного ключа $\{d, n\}$:
$$s = S_A(m) = m^d \pmod n.$$
3. Передать пару $\{m, s\}$, состоящую из сообщения и подписи.

1. Принять пару $\{m, s\}$.
2. Взять открытый ключ $\{e, n\}$ Алисы.
3. Вычислить прообраз сообщения из подписи:
$$m' = P_A(s) = s^e \pmod n.$$
4. Проверить подлинность подписи (и неизменность сообщения), сравнив $m$ и $m'$.

Поскольку цифровая подпись обеспечивает как аутентификацию автора сообщения, так и подтверждение целостности содержимого подписанного сообщения, то она служит аналогом подписи, сделанной от руки в конце рукописного документа.

Важное свойство цифровой подписи заключается в том, что ее может проверить каждый, кто имеет доступ к открытому ключу ее автора. Один из участников обмена сообщениями после проверки подлинности цифровой подписи может передать подписанное сообщение еще кому-то, кто тоже в состоянии проверить эту подпись. Например, сторона *A* может переслать стороне *B* электронный чек. После того как сторона *B* проверит подпись стороны *A* на чеке, она может передать его в свой банк, служащие которого также имеют возможность проверить подпись и осуществить соответствующую денежную операцию.

Заметим, что подписанное сообщение *m* не зашифровано. Оно пересылается в исходном виде, и его содержимое не защищено от нарушения конфиденциальности. Путем совместного применения представленных выше схем шифрования и цифровой подписи в системе RSA можно создавать зашифрованные сообщения, которые также будут содержать цифровую подпись. Для этого автор сначала должен добавить к сообщению свою цифровую подпись, а затем зашифровать получившуюся в результате пару (состоящую из самого сообщения и подписи к нему) с помощью открытого ключа, принадлежащего получателю. Получатель расшифровывает полученное сообщение с помощью своего секретного ключа. Если провести аналогию с пересылкой обычных бумажных документов, то этот процесс похож на то, как если бы автор документа поставил под ним свою печать, а затем положил его в бумажный конверт и запечатал, с тем чтобы конверт был распечатан только тем человеком, кому адресовано сообщение.

Стойкость алгоритма основывается на сложности вычисления функции, обратной к функции шифрования [37]:

$$c = E(m) = m^e \pmod{n}.$$

Для вычисления *m* по известным *c*, *e* и *n* нужно найти такое значение *d*, чтобы

$$de \equiv 1 \left(\mathrm{mod}\ \varphi(n)\right),$$

то есть

$$d \equiv e^{-1} \left(\mathrm{mod}\ \varphi(n)\right).$$

Вычисление обратного элемента по модулю не является сложной задачей, однако злоумышленнику не известно значение $\varphi(n)$. Для вычисления функции Эйлера от известного числа *n* необходимо знать разложение этого числа на простые множители.

Нахождение таких множителей и является сложной задачей. Существует множество алгоритмов для нахождения простых множителей (так называемой факторизации), самый быстрый из которых на сегодняшний день – общий метод решета числового поля.

В 2010 году группе ученых из Швейцарии, Японии, Франции, Нидерландов, Германии и США удалось успешно вычислить данные, зашифрованные при помощи криптографического ключа стандарта RSA длиной 768 бит. Нахождение простых множителей осуществлялось общим методом решета числового поля. В настоящее время криптосистемы используют минимальную длину ключа 3072 бита.

Кроме того, при неправильной или неоптимальной реализации алгоритма возможны специальные криптографические атаки (например, атаки на схемы с малой секретной экспонентой или на схемы с общим выбранным значением модуля).

# 4. РАСШИРЕНИЕ ДИАПАЗОНА ИЗМЕРЕНИЙ ИНТЕНСИВНОСТИ

При определении профиля поверхности объектов методами структурированного освещения часто используются методы управляемого фазового сдвига (англ. Phase Shift Interferometry, PSI). Для этого на объект проецируется серия синусоидальных картин с заданным сдвигом [38, 39].

Источники погрешности – неточное задание фазовых сдвигов [40] и ошибки при квантовании уровня интенсивности [41].

## 4.1. ИСПОЛЬЗОВАНИЕ МЕТОДОВ СТРУКТУРИРОВАННОГО ОСВЕЩЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ ПРОФИЛЯ

Формируется серия синусоидальных картин с заданным сдвигом фаз. Общая формула для компьютерного моделирования синусоидальных картин, ориентированных вдоль оси *X*:

$$I_i(j) = a_r \cos\left(\frac{2\pi N_p}{N_x} j - \delta_i\right), \qquad (4.1.1)$$

где $N_x$ – число точек в массиве; $N_p$ – требуемое число полос; $\delta_i$ – фазовый сдвиг; $a_r$ – диапазон изменения интенсивности.

Затем полосы проецируются на объект. Зафиксированная картина полос содержит информацию о профиле изучаемого объекта, в том числе о наклоне этого объекта к плоскости проецирования. Однако нас интересует в большей степени профиль объекта без наклона. Для этого используется простой алгоритм вычитания плоскости наклона [42]. Принципиальная схема системы проекции показана на рис. 4.1.1.

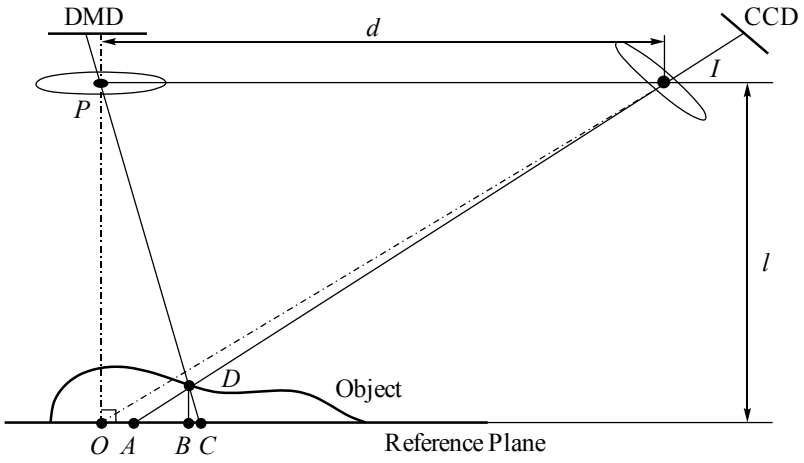



*Рис. 4.1.1.* Схема определения профиля объекта с помощью фазовой информации

Точки $P$ и $I$ – это центры выходных зрачков проектора и камеры соответственно. Оптические оси проектора и камеры пересекаются в точке $O$. Перед началом измерений объекта измеряется фазовая карта опорной плоскости, которая используется в качестве эталона при дальнейших измерениях. Профиль объекта измеряется по отношению к этой плоскости. С точки зрения проектора точка $D$ на поверхности объекта имеет то же значение, что и точка $C$ на опорной плоскости. В то же время на фотокамере точка $D$ на поверхности объекта и точка $A$ на базовой плоскости отображаются в один пиксель.

Вычитая фазовую карту опорной плоскости из фазового распределения на объекте, мы получим фазовую разность на этом же пикселе:

$$\varphi_{AD} = \varphi_{AC} = \varphi_A - \varphi_C \,. \tag{4.1.2}$$

Предположим, точки $P$ и $I$ расположены так, чтобы быть в одной плоскости с вектором $l$ (расстояние до опорной плоскости), и имеют фиксированное расстояние $d$ между ними, а опорная плоскость параллельна линии между зрачками камеры и проектора. Таким образом, треугольники $\Delta PID$ и $\Delta CAD$ подобны, высота точки $D$ на поверхности объекта по отношению к опорной плоскости $DB$ зависит от расстояния между точками $A$ и $C$:

$$\frac{d}{\overline{AC}} = \frac{l - \overline{DB}}{\overline{DB}} = \frac{l}{\overline{DB}} - 1 \,. \tag{4.1.3}$$

Поскольку $d$ намного больше, чем $\overline{AC}$ для реальных измерений, это выражение можно упростить:

$$z(x, y) = \overline{DB} \approx \frac{l}{d}\overline{AC} = \frac{pl}{2\pi d}\varphi_{AC} = K\varphi_{AC}, \qquad (4.1.4)$$

где $p$ – ширина полосы на опорной плоскости.

Таким образом, высота поверхности находится в пропорциональной зависимости от измеренной фазы.

Для объектов с большой глубиной рельефа выражения не дают точных результатов. В данной модели не учитывается также дисторсия объективов, которая приводит к неравномерности распределения полос в опорной плоскости. Эта неравномерность может быть разной для опорной плоскости и на поверхности объекта, поэтому при вычитании поля фаз на опорной поверхности из поля фаз на поверхности объекта искажения устраняться не будут. Для этого перед началом работы необходимо проводить калибровку системы.

## 4.2. ИСПОЛЬЗОВАНИЕ МЕТОДОВ ПОШАГОВОГО ФАЗОВОГО СДВИГА ДЛЯ ОПРЕДЕЛЕНИЯ ФАЗЫ

Искажения фазы проецируемых синусоидальных полос несут информацию о профиле объекта. Зафиксированную модуляцию фазы после проекции синусоидальных полос можно представить в виде

$$I(x, y) = I_0(x, y)\left[1 + V(x, y)\cos\left(\varphi(x, y)\right)\right], \qquad (4.2.1)$$

где $\varphi(x, y)$ – искомая фаза, которая несет информацию о профиле; $I_0(x, y)$ – средняя интенсивность, $V(x, y)$ – контраст.

Необходимо выделить $\varphi(x, y)$ из уравнения (4.2.1). В этом уравнении три неизвестных: $I_0(x, y)$, $V(x, y)$, $\varphi(x, y)$.

Метод пошагового фазового сдвига заключается в добавлении известного фазового сдвига к распределению синусоид при проекции полос. Если добавлять ряд сдвигов в (4.2.1), то получим систему линейных уравнений

$$I_i(x, y) = I_0(x, y)\left[1 + V(x, y)\cos\varphi\left((x, y) + \delta_i\right)\right], \qquad (4.2.2)$$

где $i = 0, 1, \ldots, m-1$ – число известных фазовых сдвигов и число уравнений в системе (4.2.2). Если допустить, что при внесении фазового сдвига значения $I_0(x, y)$

и $V(x, y)$ не будут меняться, то в каждом из этих уравнений тоже три неизвестных: $I_0(x, y)$, $V(x, y)$, $\varphi(x, y)$.

*В* [40, 43] *показано, что для однозначного определения разности фаз с помощью системы уравнений* (4.2.2) *необходимо не менее трех взаимно независимых уравнений.*

В [44–46] представлена обобщенная схема алгоритма для различного числа задаваемых фазовых сдвигов:

$$\varphi(x, y) = \arctan\left(\frac{\vec{I}^{\perp}\vec{C}}{\vec{I}^{\perp}\vec{S}}\right) = \arctan\left(\frac{\vec{I}\,\vec{C}^{\perp}}{\vec{I}\,\vec{S}^{\perp}}\right), \tag{4.2.3}$$

где $\vec{I} = (I_0, \ldots, I_{m-1})^T$ – набор измеренных интенсивностей с различными фазовыми сдвигами $\delta_i$ в каждой точке интерференционной картины $(x, y)$; $\vec{C} = (\cos\delta_0, \ldots, \cos\delta_{m-1})^T$, $\vec{S} = (\sin\delta_0, \ldots, \sin\delta_{m-1})^T$ – набор синусов и косинусов от известных фазовых сдвигов; $\vec{C}^{\perp}$, $\vec{S}^{\perp}$ – ортогональные векторы для векторов синусов и косинусов. Ортогональные векторы можно найти с помощью следующих выражений:

$$\vec{I}^{\perp} = M\,\vec{I}\,; \tag{4.2.4а}$$

$$\vec{C}^{\perp} = M\vec{C}\,; \tag{4.2.4б}$$

$$\vec{S}^{\perp} = M\vec{S}\,, \tag{4.2.4в}$$

где матрица $M$ будет иметь вид

$$M = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & -1 \\ -1 & 0 & 1 & \cdots & 0 & 0 \\ 0 & -1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & -1 & 0 \end{bmatrix}_{m \times m.} \tag{4.2.5}$$

Область определения функции arctan определена от $-\pi/2$ до $\pi/2$. Если нам известен знак $\sin(x)$ и знак $\cos(x)$, то можно расширить область определения

до $[0, 2\pi)$. В современных языках программирования вводится специальная функция двух аргументов $\mathsf{atan2(y,x)}$, которая определяется как

$$\operatorname{atan2}(y, x) = \begin{cases} \arctan\left(\dfrac{y}{x}\right) & \text{if } x > 0; \\ \arctan\left(\dfrac{y}{x}\right) + \pi & \text{if } x > 0 \text{ and } y \geq 0; \\ \arctan\left(\dfrac{y}{x}\right) - \pi & \text{if } x < 0 \text{ and } y < 0; \\ \dfrac{\pi}{2} & \text{if } x = 0 \text{ and } y > 0; \\ -\dfrac{\pi}{2} & \text{if } x = 0 \text{ and } y < 0; \\ \text{не определено} & \text{if } x = 0 \text{ and } y = 0. \end{cases} \tag{4.2.6}$$

Эта функция имеет область допустимых значений $(-\pi, \pi]$, которая добавлением $\pi$ может быть отображена на область $[0, 2\pi)$.

Расширить область определения выражения (4.2.3) можно, анализируя знаки числителя и знаменателя. Видно, что знак числителя совпадает со знаком синуса, а знак знаменателя – со знаком косинуса. Поэтому выражение (4.2.3) можно представить как

$$\varphi = \arctan\left(\frac{\vec{I}\ \vec{C}^{\perp}}{\vec{I}\ \vec{S}^{\perp}}\right) = \operatorname{atan2}\left(\vec{I}\ \vec{C}^{\perp},\ \vec{I}.\vec{S}^{\perp}\right). \tag{4.2.7}$$

Используя выражение (4.2.3), можно получить формулы для определения фазы при различном числе известных фазовых сдвигов. Для трех фазовых сдвигов имеем

$$\vec{I}^{\perp} = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \end{bmatrix} = \begin{bmatrix} I_2 - I_3 \\ -I_1 + I_3 \\ I_1 - I_2 \end{bmatrix}. \tag{4.2.8}$$

Перемножая полученный вектор на векторы $\vec{C}$ и $\vec{S}$, получим

$$\varphi(x, y) = \arctan\frac{(I_2 - I_3)\cos\delta_1 + (I_3 - I_1)\cos\delta_2 + (I_1 - I_2)\cos\delta_3}{(I_2 - I_3)\sin\delta_1 + (I_3 - I_1)\sin\delta_2 + (I_1 - I_2)\sin\delta_3}. \tag{4.2.9}$$

Таким образом, при использовании фиксированных фазовых сдвигов можно выводить различные формулы получения фазы.

Наиболее часто используются выражения для четырех сдвигов:

$$\varphi(x, y) = \arctan\frac{I_1 - I_3}{I_4 - I_2} \tag{4.2.10}$$

и пяти сдвигов:

$$\varphi_5 = \operatorname{arctg}\frac{2(I_2 - I_4)}{I_1 - 2I_3 + I_5}. \tag{4.2.11}$$

Для сдвигов, кратных $2\pi$:

$$\delta_1 = 0°, \quad \delta_2 = \frac{2\pi}{N}, \quad \delta_3 = \frac{3\pi}{N}, \ldots, \delta_n = 2\pi \ . \tag{4.2.12}$$

Обобщенное уравнение расшифровки (4.2.7) можно переписать в виде

$$\varphi(x, y) = \arctan\frac{\sum_{i=0}^{m-1} I_i \sin\delta_i}{\sum_{i=0}^{m-1} I_i \cos\delta_i}. \tag{4.2.13}$$

Впервые такое выражение было получено в [47].

Одним из недостатков вышерассмотренных алгоритмов является необходимость установки точного значения сдвига. Иногда бывает проще задавать одинаковый сдвиг, но величина этого сдвига может быть неизвестной.

Например, в алгоритме Carré [48] фазовые сдвиги должны точно удовлетворять условию

$$\delta_1 = -3\delta, \ \delta_2 = -\delta, \ \delta_3 = \delta, \ \delta_4 = 3\delta, \tag{4.2.14}$$

где $\delta$ – произвольный вносимый фазовый сдвиг. В этом случае можно получить выражение для определения фазы

$$\varphi_{\text{Carre}} = \operatorname{arctg}\frac{\sqrt{[(I_1 - I_4) + (I_2 - I_3)][3(I_2 - I_3) - (I_1 - I_4)]}}{(I_2 + I_3) - (I_1 + I_4)}. \tag{4.2.15}$$

Этот алгоритм может быть получен из обобщенной формулы расшифровки [40, 49]. Поскольку в числителе выражения (4.2.15) используется взятие корня, то для анализа знаков числителя и знаменателя (для расширения области определения

от $[0,\pi)$ до $[0,2\pi))$ необходимо анализировать не знаки числителя и знаменателя, а знаки следующих выражений:

$$(I_2 + I_3) = [2I_0 V \sin\alpha]\sin\varphi\,; \tag{4.2.16}$$

$$(I_2 + I_3) - (I_1 + I_4) = [2I_0 V \cos\alpha \sin^2\alpha]\cos\varphi\,. \tag{4.2.17}$$

Поскольку значения множителей, заключенных в квадратные скобки, положительны, то левые части этих равенств определяют квадрант, в котором находится искомый фазовый угол. Заметим, что выражение (4.2.17) совпадает с выражением для знаменателя формулы Carré (4.2.15). Поэтому знак знаменателя в формуле расшифровки совпадает со знаком $\cos\varphi$, а знак числителя необходимо определять с помощью выражения (4.2.16).

## 4.3. ИСТОЧНИКИ ПОГРЕШНОСТЕЙ ПРИ ИСПОЛЬЗОВАНИИ МЕТОДОВ ПОШАГОВОГО ФАЗОВОГО СДВИГА В ПРОЕКЦИОННЫХ МЕТОДАХ

Рассмотрим выражение (4.2.13). Видно, что погрешность измерения фазы зависит от погрешности задания фазового сдвига и погрешности измерения интенсивности.

В [50] было получено выражение для оценки абсолютной погрешности при условии $\Delta\varphi << \varphi_i$ и $\Delta I << I_i$:

$$\Delta\varphi \approx \sum_{i=1}^{m}\left|\frac{\partial\varphi}{\partial\delta_i}\right|(\pm\Delta\delta_i) + \sum_{i=1}^{m}\left|\frac{\partial\varphi}{\partial I_i}\right|(\pm\Delta I_i)\,; \tag{4.3.1}$$

$$\frac{\partial\varphi}{\partial\delta_i} = \frac{I_i\cos\delta_i C_d + I_i\sin\delta_i S_d}{(C_d)^2 + (S_d)^2}\,; \tag{4.3.2}$$

$$\frac{\partial\varphi}{\partial I_i} = \frac{\sin\delta_i C_d - \cos\delta_i S_d}{(C_d)^2 + (S_d)^2}\,, \tag{4.3.3}$$

где $m$ – число фазовых сдвигов; $C_d = \sum_{i=1}^{m}\Delta I_i\cos\delta_i$; $\quad S_d = \sum_{i=1}^{m}\Delta I_i\sin\delta_i$ .

В проекционных системах возможно точное задание фазовых сдвигов. Поэтому ошибки, связанные с неправильным заданием сдвига (4.3.2), можно не учитывать.

Однако в проекционных системах из-за нелинейности систем проекции и ввода условие $\Delta I << I_i$ не выполняется, поэтому необходимо использовать специальные методики для снижения уровня отклонения зарегистрированной интенсивности от реальной.

## 4.4. СНИЖЕНИЕ НЕЛИНЕЙНЫХ ИСКАЖЕНИЙ ИНТЕНСИВНОСТИ

Когда проецируется и регистрируется цифровое изображение, оно подвергается искажениям. Для измерительных систем такие искажения интенсивности необходимо устранить.

В качестве передаточной функции при нелинейных искажениях чаще всего используется степенная функция (гамма-искажения) в виде

$$I_{res}(x, y) = a\, I(x, y)^{\gamma}, \qquad (4.4.1)$$

где $I(x, y)$ – интенсивность по полю; $\gamma$ – уровень гамма-искажений. При $\gamma = 1$ характеристика передачи полутонов линейна.

Кроме гамма-искажений, которые описываются выражением (4.4.1), могут возникать и другие типы искажений, связанные с неравномерностью освещения, неравномерностью отражения от исследуемой поверхности и другие, которые могут иметь более сложный вид.

Принцип гамма-коррекции показан на рис. 4.4.1. Если проектировать равномерный клин, то в результате искажений при проекции и регистрации можно зарегистрировать график интенсивности $I_{reg}$. Для получения равномерной характеристики (рис. 4.4.1, *б*) исходный сигнал необходимо исказить по закону, показанному на графике рис. 4.4.1, *а*.

Необходимо подобрать обратную зависимость (график *а* на рис. 4.4.1) с целью скорректировать проецируемое изображение так, чтобы интенсивность при вводе изображений была линейной.

Уровень нелинейных искажений может быть оценен способом, основанным на проекции идеального клина с возрастанием и убыванием интенсивности (рис. 4.4.2). Если сложить эти картины, то при правильной регистрации интенсивности получится прямая. Если же при проекции или регистрации возникают искажения, то отклонение суммы прямого и обратного клина от прямой будет сигнализировать о наличии нелинейных искажений.

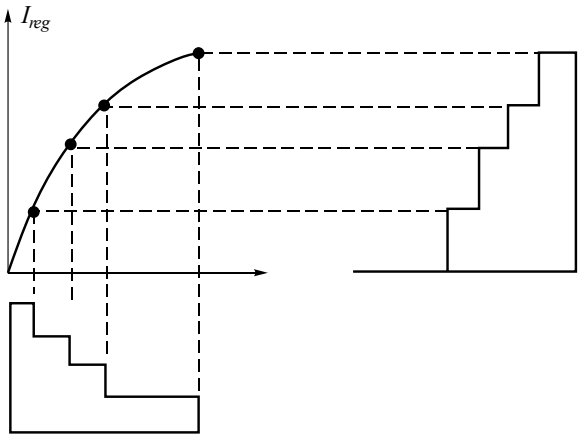



*Рис. 4.4.1.* Принцип гамма-коррекции

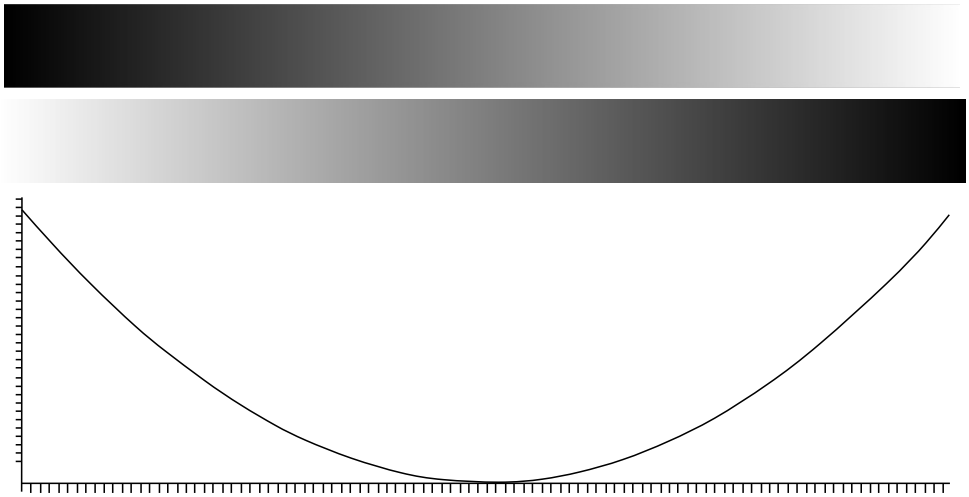


*Рис. 4.4.2.* Сумма прямого и обратного клина с искажениями интенсивности с коэффициентом $\gamma = 2$

Для небольшого числа градаций можно подобрать значения обратного закона вручную, но для большого числа градаций желательно сделать эту процедуру автоматической [51].

Для автоматической калибровки будем использовать клин с 16 градациями яркости. Использование небольшого числа градаций (16 вместо 256) определяется шумами при отражении от диффузной поверхности объекта (рис. 4.4.3).

Если поверхность белого цвета, то в случае проекции нулевой интенсивности при отражении всегда будет фиксироваться некоторое значение, отличное от нуля. В данном эксперименте при отсутствии освещения фиксируется значение интенсивности, равное 16.

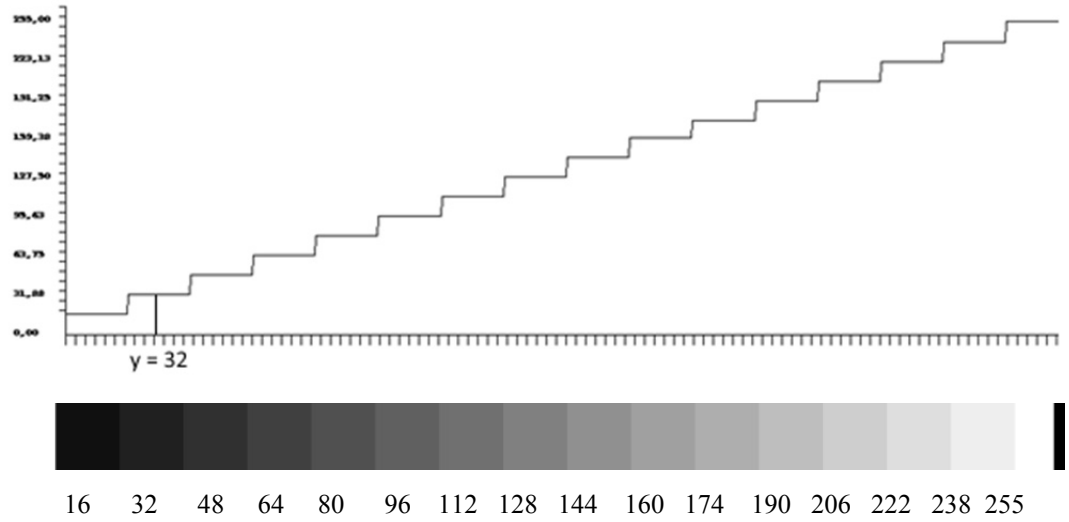



*Рис. 4.4.3.* Идеальный клин с 16 градациями интенсивности

Необходимо для каждой ступени проецированного клина найти интенсивность, при освещении которой получим в результате регистрации значение, соответствующее идеальному значению.

Например, для второй ступени необходимо получить на выходе интенсивность со значением 32. Для этого проецируем на поверхность идеальный клин со значениями интенсивности от нуля до 255. На графике, соответствующем идеальному клину, получим исходное значение интенсивности, равное 32 (рис. 4.4.4). Это значение соответствует координате на введенном изображении клина $x = 177$.

Теперь необходимо найти соответствие этого значения тому, которое мы проецировали. Для этого формируем идеальный клин с размерами, соответствующими введенному клину. Координате, равной 177, соответствует значение интенсивности, равное 50 (рис. 4.4.5).

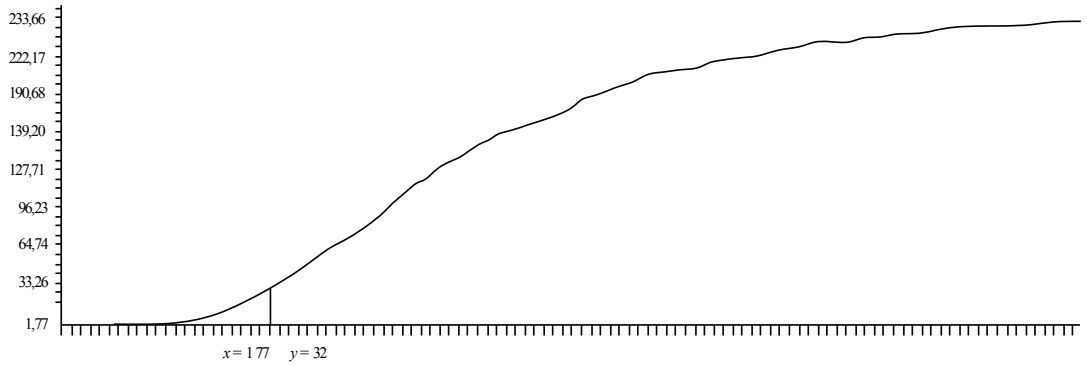



*Рис. 4.4.4.* Значение для второй ступени на графике интенсивности при проекции равномерного клина от нуля до 255 уровней

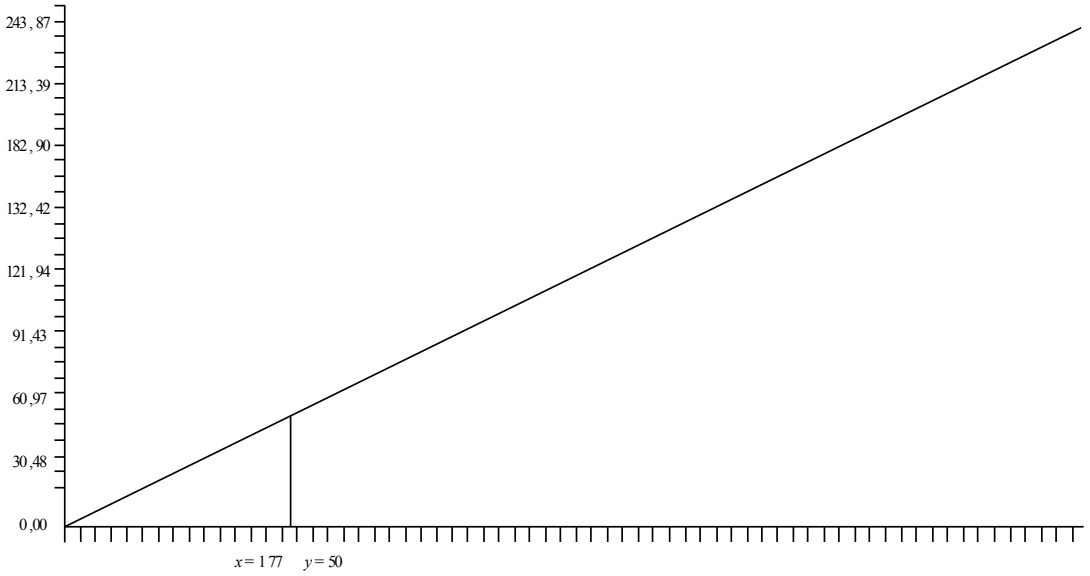



*Рис. 4.4.5.* Идеальный клин для нахождения искомых точек

Таким образом можно определить все ступени обратного клина (рис. 4.4.6).

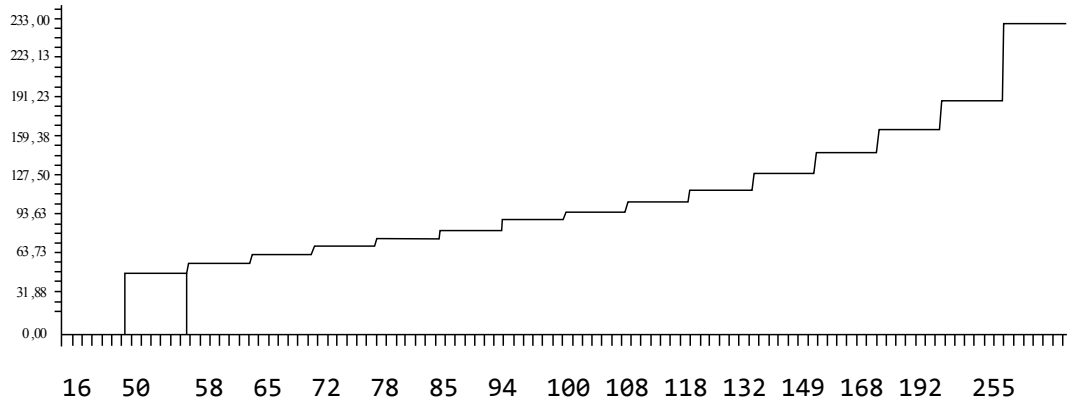



*Рис. 4.4.6.* Обратный клин для коррекции нелинейности освещения

Если спроецировать такой клин и зарегистрировать его с помощью фотокамеры, то получим клин, близкий к идеальному.

Для аппроксимации клина с 16 градациями интенсивности в клин, который имеет 256 градаций, был разработан следующий простой алгоритм.

Для перевода клина с 16 градациями каждая ступенька разбивается на две. Первая часть ступени остается такой же, а вторая получает среднее значение между этой ступенью и следующей. Эта процедура может быть описана на С-подобном языке следующим образом:

```
double[] MasX2(double[] am)
  {
    int nx  = am.Length;
    int nx2 = nx * 2;
    double[] am2 = new double[nx2];

    for (int i = 0; i < nx2-1; i++)
     {
       if (i % 2 == 0) am2[i] =  am[i/2];
       if (i % 2 != 0) am2[i] = (am[i/2] + am[i / 2 + 1])/2;
      }

    return am2;
  }
```

К сожалению, в результате становится на одну точку меньше, чем надо. Например, при увеличении из 16 ступенек становится 31. Это видно из табл. 4.4.1.

Таблица 4.4.1

**Разбиение 16 ступенек на две**

| 1 | | 2 | | 3 | | | 15 | | 16 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1,5 | 2 | 2,5 | 3 | 3,5 | ... | 15 | 15,5 | 16 | |
| 1 | 2 | 3 | 4 | 5 | 6 | | 29 | 30 | 31 | 32 |

Последняя, 32-я, ступенька не определена или может повторять предыдущую (рис. 4.4.7). В результате мы получим последовательность ступенек короче на $nx / 32$, где $nx$ – длина проецируемого массива.

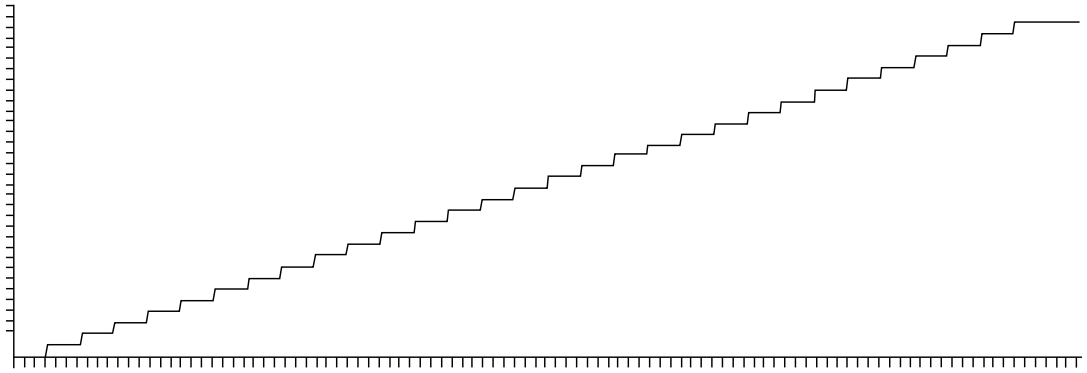


*Рис. 4.4.7.* Разбиение 16 ступенек на две

Для получения правильного клина необходимо расширить массив на $nx/32$. При следующем делении на два необходимо расширить на $nx/32+nx/64$. При делении на 256 необходимо расширить клин на $nx/32+nx/64+nx/128+nx/256$ пикселей. В результате мы получим 241 ступень при диапазоне сигнала от нуля до 255 (рис. 4.4.8).

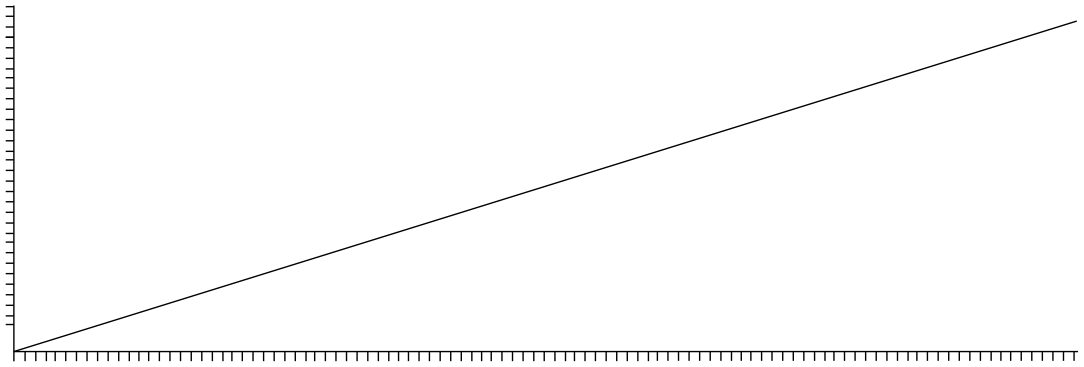


*Рис 4.4.8.* Билинейная аппроксимация 16 ступенек

Учитывая погрешности при отражении от диффузной поверхности объекта, такая погрешность аппроксимации будет несущественной.

Если обратный клин с 16 градациями аппроксимировать на 255 значений, то получим график, представленный на рис. 4.4.9.

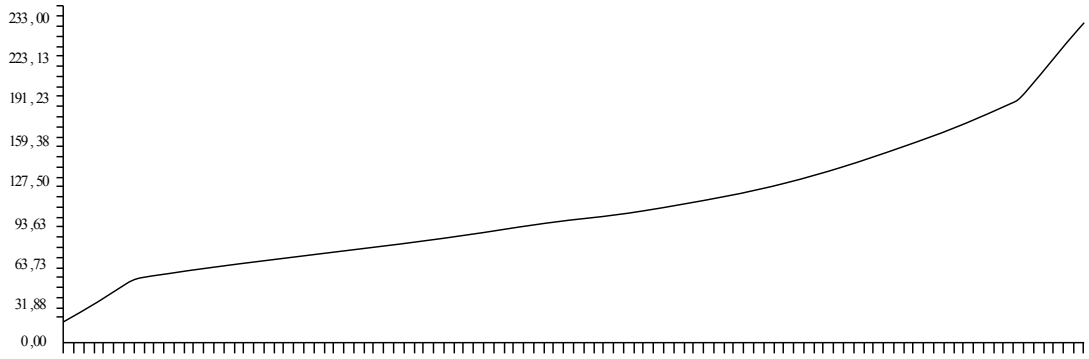



*Рис. 4.4.9.* Обратный клин для коррекции нелинейности
с градациями интенсивности от нуля до 255

Если спроектировать этот график (рис. 4.4.9) на поверхность, то после регистрации получим характеристику, близкую к равномерной (рис. 4.4.10).

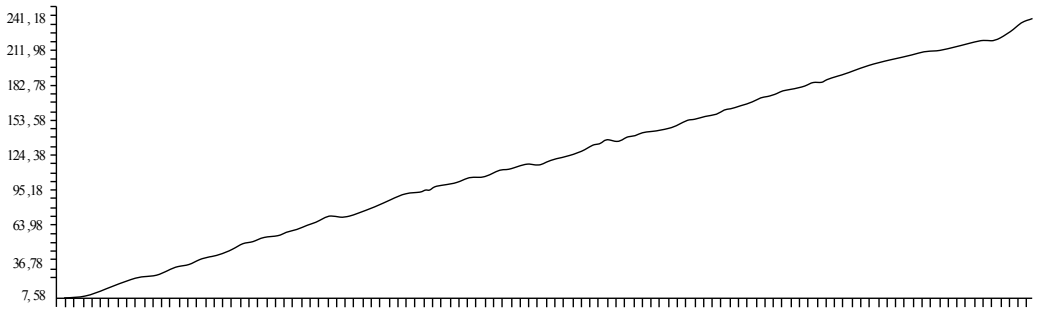
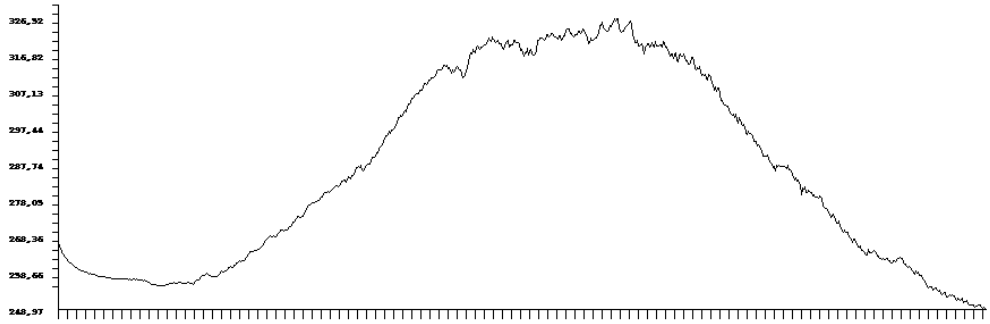
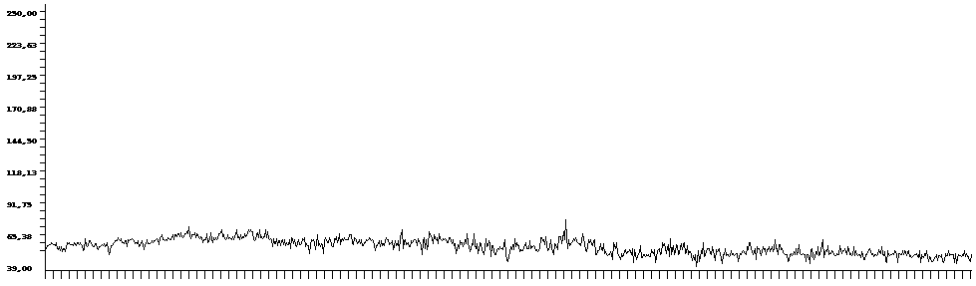



*Рис. 4.4.10.* График интенсивности при проекции равномерного клина от нуля до 255 уровней после устранения нелинейности

Результаты суммирования прямого и обратного клина до устранения нелинейности и после него показаны на рис. 4.4.11. Из графиков видно, что нелинейность при передаче интенсивности практически отсутствует.




*а*




*б*

*Рис. 4.4.11.* Сумма прямого и обратного клина до коррекции (*а*) и после нее (*б*)

В трактах передачи цветного изображения гамма-коррекция осуществляется в каждом из трех каналов основных цветов. Предварительную гамма-коррекцию с помощью обратного клина необходимо проводить программно при формировании картины освещения.

Из нижнего графика (рис. 4.4.11) видно, что хотя нелинейность устраняется, но погрешность определения интенсивности при отражении от поверхности достаточно велика. Поэтому и погрешность определения фазы при использовании методов PSI остается достаточно большой.

## 4.5. РАСШИРЕНИЕ ДИАПАЗОНА ИЗМЕРЕНИЙ ИНТЕНСИВНОСТИ НА ОСНОВЕ МОДУЛЬНОЙ АРИФМЕТИКИ

Большая часть устройств проецирования имеет фиксированный диапазон интенсивности от нуля до 255. Повышение диапазона – сложная задача. В этом разделе опишем способ ее решения, основанный на модульной арифметике.

Для повышения числа квантов разбиваем картину синусоидальных полос с диапазоном от нуля до 800 на две части с диапазоном 213 и 167 (рис. 4.5.1, цветной вариант см. на форзаце).

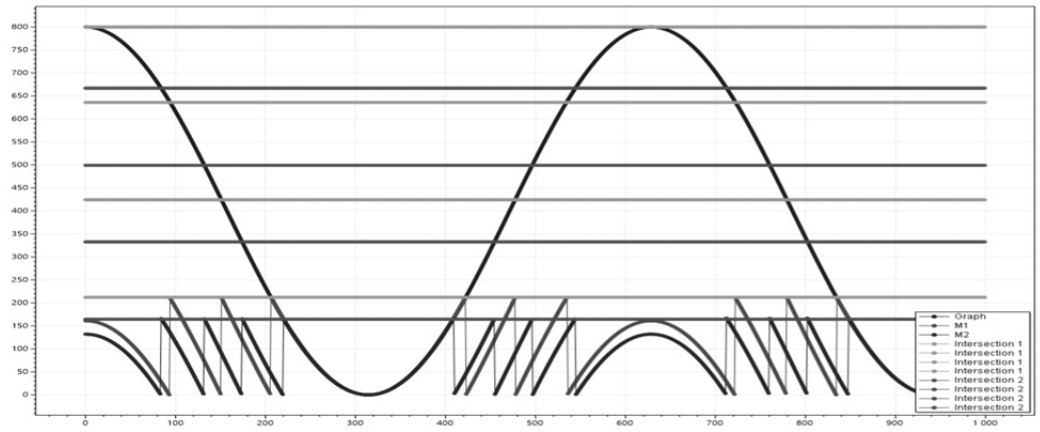



*Рис. 4.5.1.* Разбиение синусоиды с диапазоном изменения интенсивности от нуля до 800 на две части с диапазоном интенсивности от нуля до 213 и от нуля до 167

Для уменьшения ошибок, связанных с погрешностями от нелинейности при проецировании этих синусоидальных картин, приведем их к диапазону от нуля до 255 (рис. 4.5.2, 4.5.3).

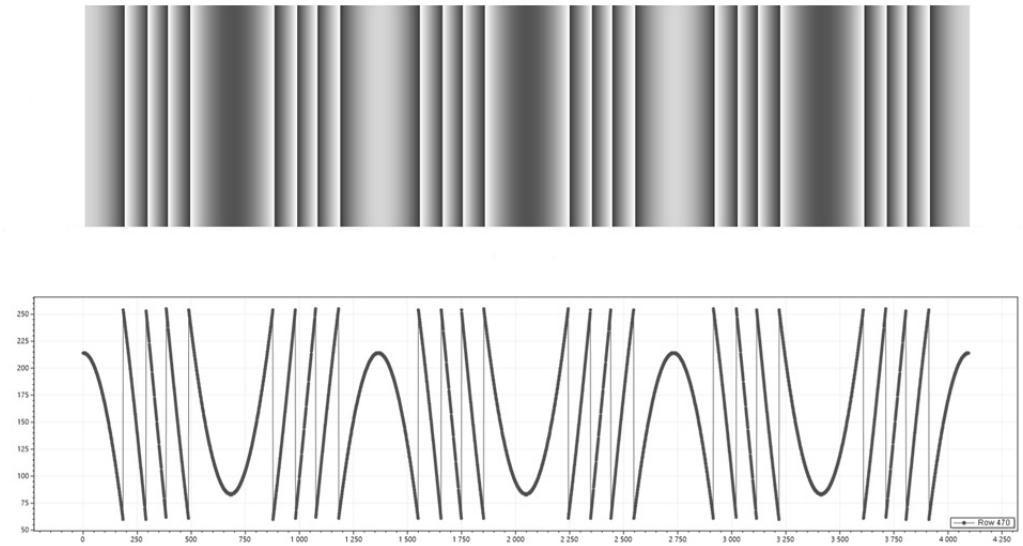



*Рис. 4.5.2.* Приведение синусоиды с интенсивностью от нуля до 167 к диапазону от 60 до 250

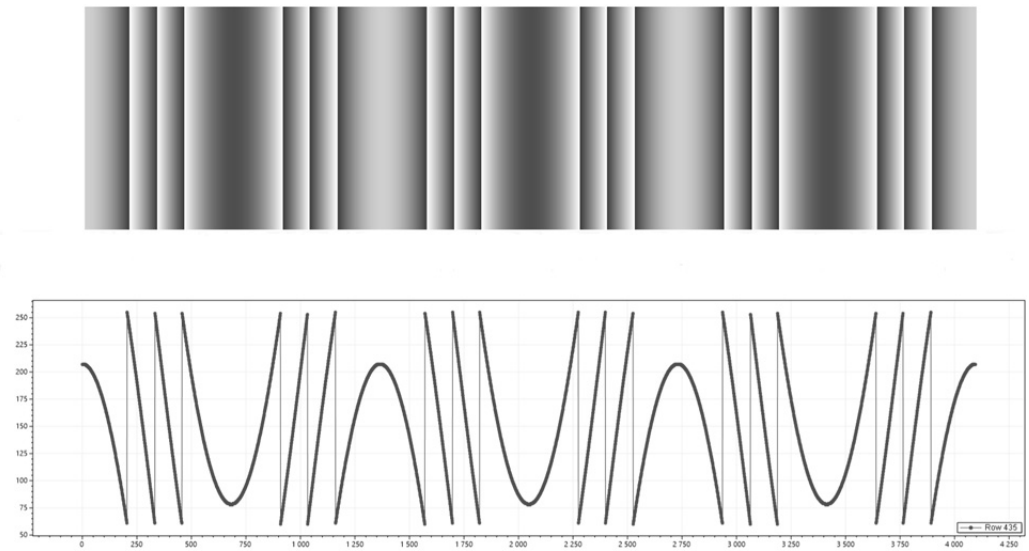



*Рис. 4.5.3.* Приведение синусоиды с интенсивностью от нуля до 213 к диапазону от 60 до 250

Эти картины проецируем на объект. После отражения от объекта и ввода с помощью фотокамеры получим синусоиды, графики сечений которых показаны на рис. 4.5.4, 4.5.5.

После ввода графики на этих рисунках приведены к масштабу от нуля до 213 и от нуля до 167.

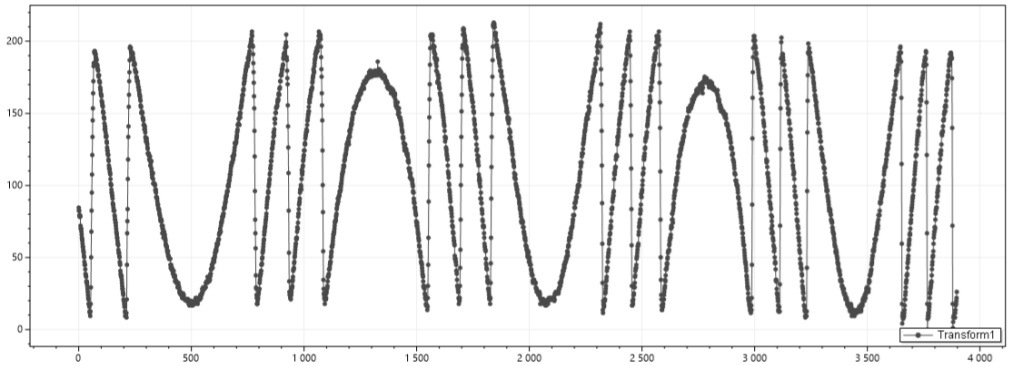



*Рис. 4.5.4.* Синусоидальная картина, отраженная от объекта и масштабированная к диапазону от нуля до 213

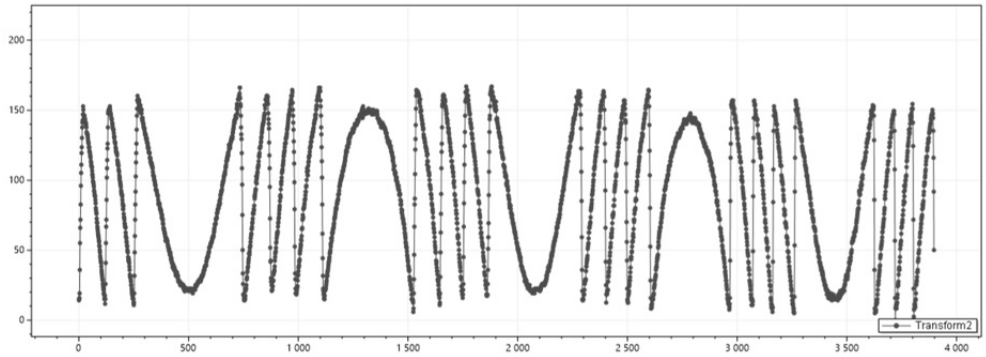



*Рис. 4.5.5.* Синусоидальная картина, отраженная от объекта и масштабированная к диапазону от нуля до 167

Если взять распределения точек и показать их в таблице, в которой по оси $Y$ значения от нуля до 167, а по оси $X$ – от нуля до 213, то получим изображение, показанное на рис. 4.5.6. Отклонения точек от идеальных диагоналей вызваны погрешностями в тракте проецирования и ввода.

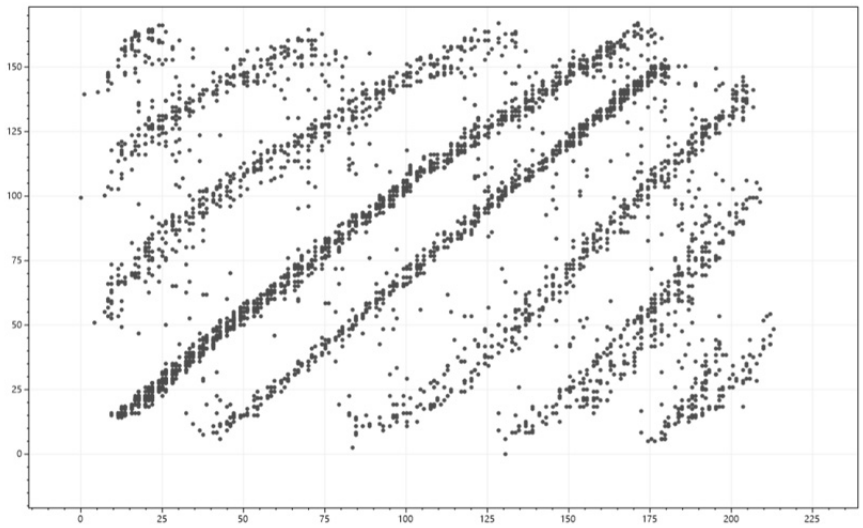



*Рис. 4.5.6.* Распределение экспериментальных точек в таблице остатков

На рис. 4.5.7 и 4.5.8 показан алгоритм устранения сбойных точек. Этот алгоритм описан в следующем разделе. (Цветные варианты рис. 4.56, 4.57 и 4.58 см. на форзаце.)

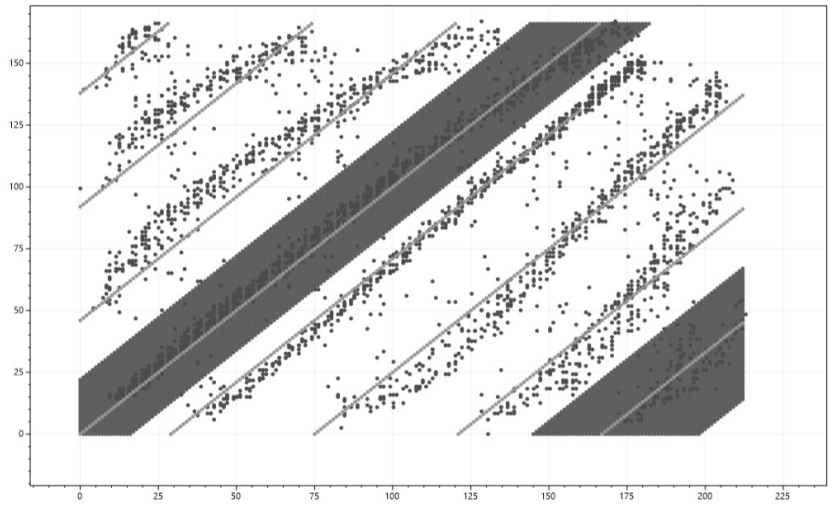



*Рис. 4.5.7.* Выделение одной диагонали в таблице остатков

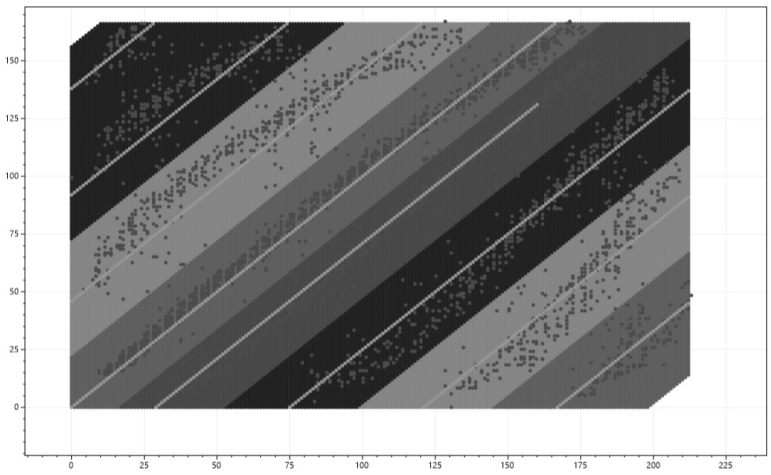



*Рис. 4.5.8.* Прослеживание всех диагоналей

На рис. 4.5.9 показана восстановленная синусоида с диапазоном от нуля до 800.

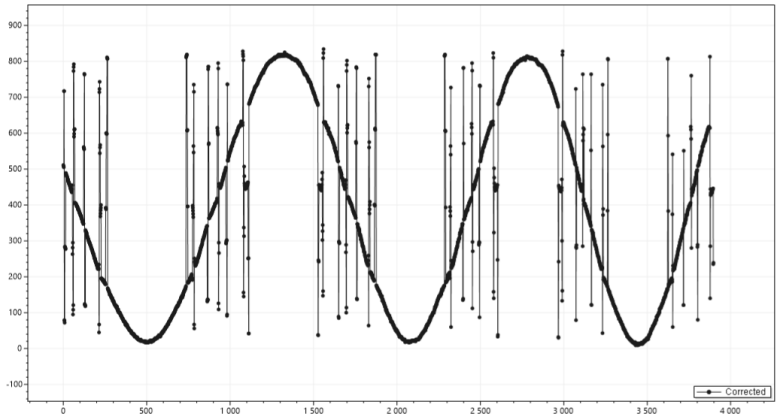



*Рис. 4.5.9.* Восстановление синусоидальной картины
в диапазоне от нуля до 800

Таким образом, мы получили синусоидальную картину с диапазоном распределения интенсивности в три раза больше (от нуля до 800), чем позволяет устройство проецирования (от нуля до 255). Сбойные точки являются следствием нелинейности тракта «проецирование – ввод интенсивностей» и могут быть устранены методами нелинейной фильтрации.

# 5. РАСШИРЕНИЕ ОБЛАСТИ ФАЗОВОЙ НЕОДНОЗНАЧНОСТИ ПРИ ИНТЕРФЕРЕНЦИОННЫХ И ПРОЕКЦИОННЫХ ИЗМЕРЕНИЯХ НА ОСНОВЕ МОДУЛЬНОЙ АРИФМЕТИКИ

Одним из наиболее точных оптических методов измерений являются интерференционные и голографические методы. Они основаны на расшифровке системы синусоидальных полос, которая создается на поверхности объекта с помощью когерентно-оптических лазерных систем. Одной из причин, сдерживающих широкое распространение подобных методов, является ограничение диапазона измерений. По картинам синусоидальных полос фазовые значения могут однозначно восстанавливаться только в пределах периода – от нуля до $2\pi$. Это связано с периодичностью синусоидальной картины.

Методы расшифровки, которые используются в интерферометрии, можно использовать и для проекционных фазовых систем. В таких системах синусоидальные полосы проецируются на поверхность объекта. Если проецировать на объект картины с различными периодами, то в результате обработки получим серию фазовых картин в периоде от нуля до $2\pi$, по которым можно восстановить в достаточно большом диапазоне действительное распределение, соответствующее профилю поверхности.

Величина периода полос определяется фазовыми значениями, при которых регистрируемая интенсивность меняется от минимума до максимума. Для интерференционных методов эта величина соизмерима с длиной волны используемого освещения (примерно 0,5 мкм), для проекционных методов – с размером периода проецируемой синусоиды (3…10 мм).

Для получения полного значения измеряемых фазовых значений обычно используются методы развертывания.

## 5.1. УСТРАНЕНИЕ ФАЗОВОЙ НЕОДНОЗНАЧНОСТИ МЕТОДОМ РАЗВЕРТЫВАНИЯ

Полная фаза может определяться развертыванием фазы, т. е. последовательным добавлением $2\pi$ к соседним точкам (либо вычитанием $2\pi$), если перепад между ними превышает некоторый порог (рис. 5.1.1). Такая процедура основана на допущении, что резкие (более периода) скачки отсутствуют [39].

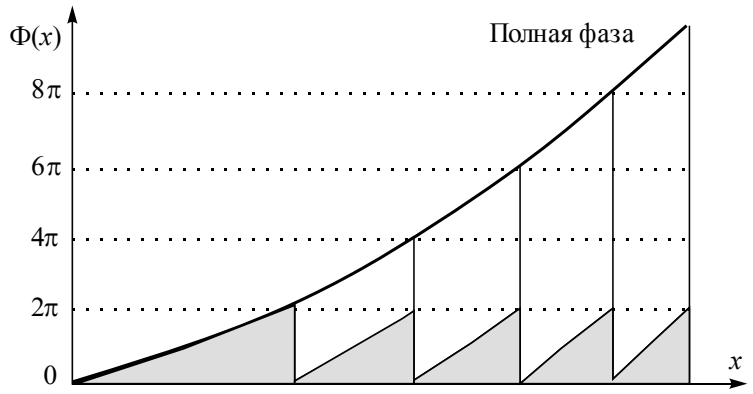


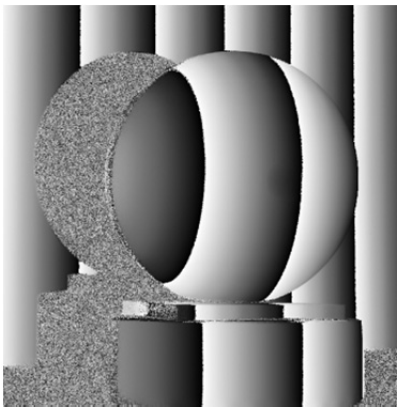

*Рис. 5.1.1.* Восстановление полной фазы (серым цветом обозначены измеренные фазовые значения в пределах $2\pi$)

Для прослеживания границы перехода необходимо, чтобы количество периодов было на порядок меньше, чем количество точек в массиве детектора. Гипотеза о существовании фазового перехода в некоторой точке фазового поля принимается в зависимости от результатов анализа ее окрестности. Такая процедура называется фазовым развертыванием.

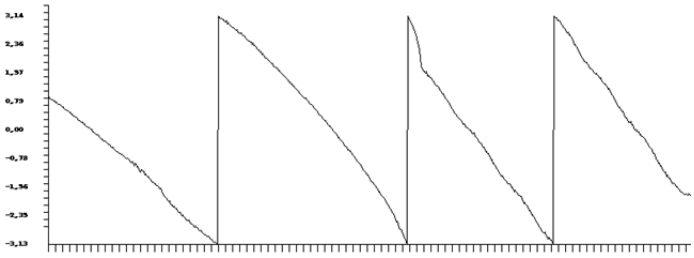
Первые работы по фазовому развертыванию строились на сравнении соседних значений в столбцах и строках массива. При этом использовалась информация о предыдущих восстановленных точках для определения волнового фронта в следующих точках. Поскольку процедура развертывания на каждом шаге зависит от предыдущих вычислений, единичная ошибка приводила к лавинообразному нарастанию погрешности.

Лучшие результаты дает модификация метода с использованием анализа по областям. Каждая точка связывается с определенной областью, в которой не должно быть фазовых скачков. Такие области ограничены линиями фазовых разрывов. Каждой области ставится в соответствие целое число $n$, которое показывает, что в этой области ко всем точкам нужно добавить $2\pi n$ (или вычесть $2\pi n$).

Рассмотрим фазовую картину (рис. 5.1.2).



*а*



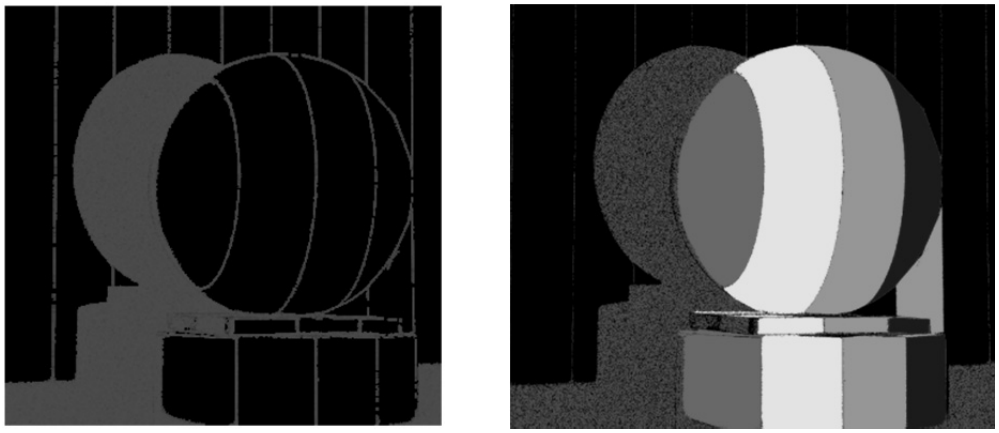

*б*

*Рис. 5.1.2.* Исходное поле фаз (*а*)
и график по центральному сечению (*б*)

**Первый этап –** выделение границ зон фазовой монотонности. На рис. 5.1.2, *а* показаны выделенные границы областей. Границы областей монотонности распределения фаз выделяются по перепадам, близким к $2\pi$. В пределах этих зон фазовые зна-

чения распределяются от нуля до $2\pi$. Выделенные границы очень редко имеют хорошее качество. В большинстве случаев в них присутствуют разрывы, которые необходимо устранять вручную. Иногда границы областей могут быть достаточно сложными для анализа оператором. Поэтому системы для определения однородных фазовых областей похожи на интерактивные графические редакторы.

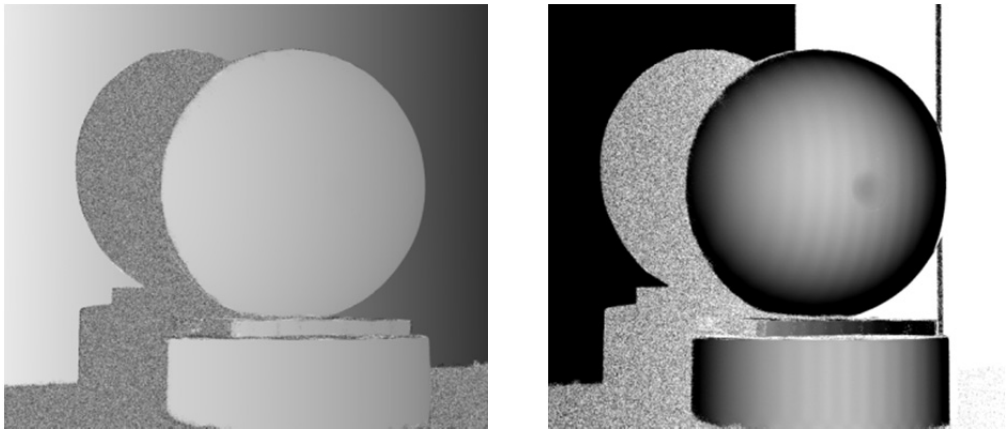
**Второй этап** – заполнение областей. На этом этапе оператор для каждой области устанавливает число $n$, которое указывает, сколько периодов надо добавить или вычесть из всех точек области. Это число задается исходя из априорного знания о форме поверхности. На рис. 5.1.3, *б* показана нумерация зон фазовой монотонности, каждый цвет соответствует $n$. Для простых распределений этот процесс может быть автоматизирован. Однако в общем случае (например, незамкнутые области, области с переходом через несколько периодов) такой процесс зависит от наличия априорной информации у оператора.



*а* *б*

*Рис. 5.1.3.* Выделенные границы областей (*а*) и зоны монотонности фазы (*б*)

**Третий этап** – восстановление полной фазы. Этот этап выполняется автоматически. Для всех точек из выделенных областей исходного фазового распределения добавляются значения $2\pi n$. На рис. 5.1.4, *а* показано поле фаз после устранения фазовой неоднозначности. После удаления наклона получим результирующее поле (рис. 5.1.4, *б*).



*а* *б*

*Рис. 5.1.4.* Восстановление полной фазы анализом областей:

*а* – результирующее поле фаз после устранения фазовой неоднозначности;
*б* – после устранения наклона

Цветные варианты рис. 5.1.3 и 5.1.4 см. на форзаце.

Основные недостатки алгоритмов развертывания:

- необходимость ручной коррекции зон монотонности;
- количество анализируемых областей зависит от параметров устройства регистрации и на практике сравнительно невелико ( $n = 10 \sim 20$ ).

Указанные ограничения можно убрать, если использовать модульную арифметику для решения этой задачи.

## 5.2. УСТРАНЕНИЕ ФАЗОВОЙ НЕОДНОЗНАЧНОСТИ НА ОСНОВЕ РЕШЕНИЯ СИСТЕМЫ СРАВНЕНИЙ

Возьмем две серии измерений с картинами периодичных полос, одна из которых имеет размер полосы 3, другая – 5 (в некоторых условных единицах). В результате измерений получится две серии фазовых распределений, которые будут меняться от нуля до $2\pi$ (рис. 5.2.1).

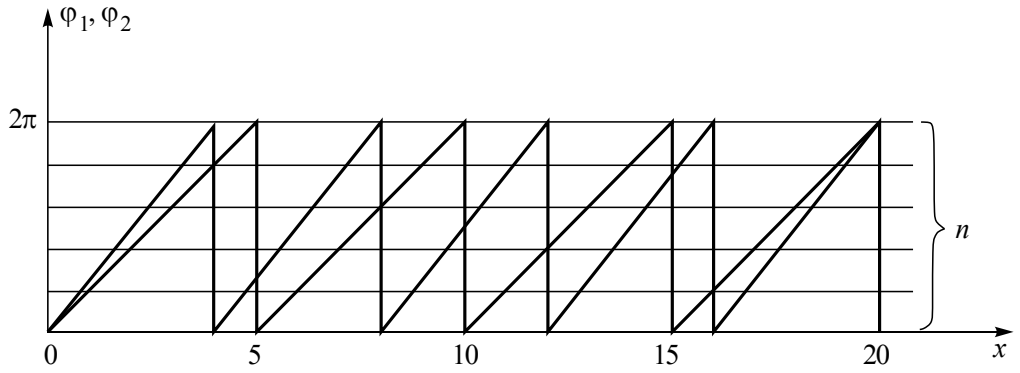



*Рис. 5.2.1.* Фазовые значения при двух ценах интерференционных полос

После пересчета от фазовых значений к размерам соответствующих полос получим два распределения (рис. 5.2.2), которые будут периодично меняться уже в диапазонах 0…3 и 0…5.

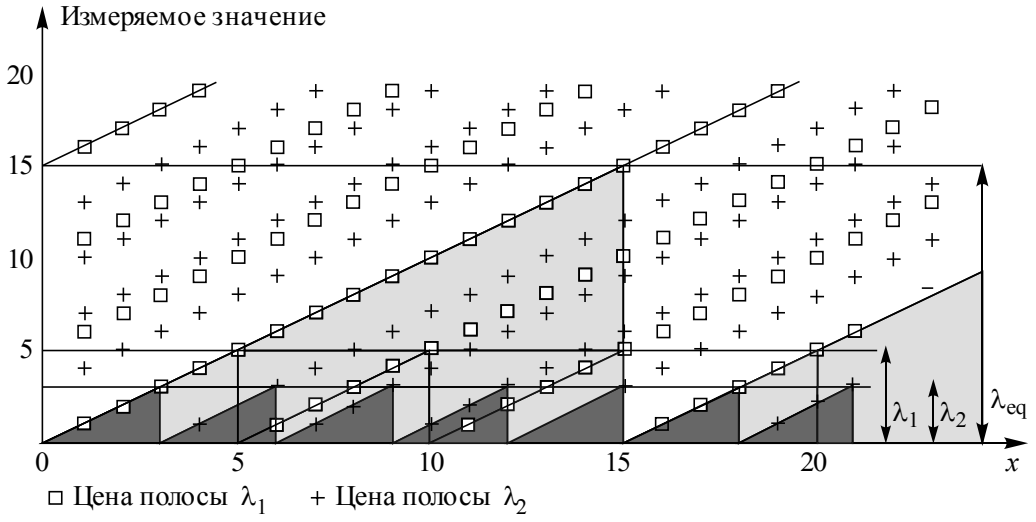



*Рис. 5.2.2.* Расширение диапазона при использовании двух синусоидальных картин с периодами, которые относятся как 3 : 5

Действительным значением для каждой разности хода может быть любое из значений, отмеченное на графике значками «□» и «+». Эти значения получаются добавлением к своему значению величин, кратных периоду. Поскольку результат при различных измерениях должен быть одинаковым (измеряем одинаковые значения в выбранной точке объекта), то отбрасываются все несовпадающие значения. Видно, что значения, удовлетворяющие обоим наборам, всё же повторяются с некоторым периодом, но этот период значительно больше, чем период каждой из пилообразных функций. Он и определяет диапазон однозначных решений при измерениях с различными периодами.

Можно составить таблицу решений, просчитав все возможные сочетания. При больших значениях периодов таблица получается слишком громоздкой. В случае нескольких измерений многомерную таблицу сложно интерпретировать, поэтому лучше найти аналитическое решение.

В [52, 53] приведен метод восстановления абсолютных значений измеряемой величины по значениям нескольких измерений в пределах различных периодов. В литературе этот метод получил название G–S-алгоритм [54–58]. Метод основан на аппарате модульной арифметики.

Если провести серию измерений с разными ценами полос $m_i$, то получим серию значений измеряемой величины $(b_1, b_2, \ldots, b_n)$. Каждое значение $b_i$ меняется в диапазоне от нуля до $m_i - 1$.

По серии этих значений необходимо определить измеряемую величину в большем диапазоне, чем цена полосы.

Возникают два вопроса.

1. Как это сделать?

2. Какое максимальное увеличение диапазона можно получить?

Задача расширения диапазона измерений сводится к задаче решения системы сравнений.

Рассмотрим систему сравнений первой степени с одним неизвестным:

$$\begin{cases} L \equiv b_1 \bmod (m_1); \\ \quad \cdots \\ L \equiv b_n \bmod (m_n), \end{cases} \tag{5.2.1}$$

где $b$ – остаток от деления $L$ на число $m$.

Решить систему сравнений (5.2.1) – это значит найти число $L$ по его остаткам $(b_1, b_2, \ldots, b_n)$. Для нахождения числа по набору остатков можно использовать китайскую теорему об остатках.

Решение этой системы можно записать в виде

$$L \equiv M_1 N_1 b_1 + M_2 N_2 b_2 + \ldots + M_n N_n b_n (\mathrm{mod}\, m_1 m_2 \ldots m_n). \qquad (5.2.2)$$

Таким образом, максимально возможный диапазон определяется произведением чисел $m_1 m_2 \ldots m_n$. Через период, определяемый этим произведением, числа будут повторяться (рис. 5.2.2).

Недостатком такого алгоритма является то, что решение системы сравнений – задача неустойчивая. Даже небольшие погрешности при измерениях начальных значений $b_i$ приводят к значительным ошибкам при определении полного значения измеряемой величины. Неустойчивость решения долгие годы не позволяла использовать китайскую теорему в инженерных расчетах. Задачу можно сделать устойчивой, если ограничить максимально возможный диапазон получения решений системы сравнений.

## 5.3. КОРРЕКЦИЯ ОШИБОЧНЫХ ЗНАЧЕНИЙ

Как показано в разделе 1.3.1, решения системы сравнений расположены на поверхности тора. Если ограничить максимальный диапазон, то между витками тора образуется некоторое свободное место. Постепенно при увеличении максимального диапазона числа заполняют всю поверхность тора (рис. 5.3.1).

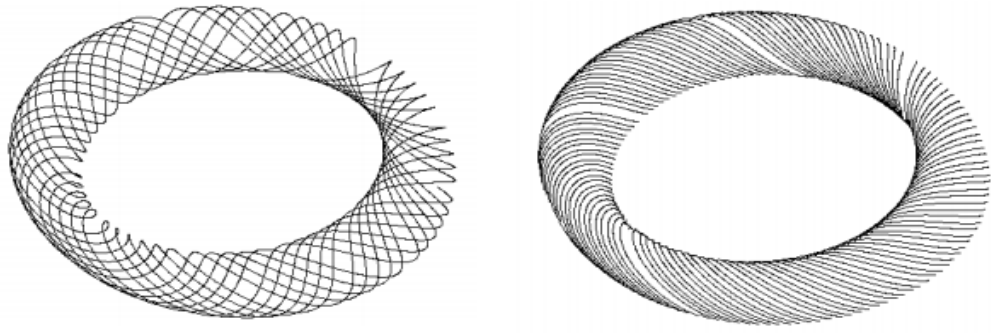


*Рис. 5.3.1.* Постепенное заполнение поверхности тора числами при увеличении числа

Рассмотрим случай для двух взаимно простых модулей. Пусть дана система из двух сравнений с $m_1 = 53$ и $m_2 = 63$. Решение системы имеет вид

$$L \equiv 1008\, b_1 + 2332\, b_2 \bmod (53 \cdot 63) = 1008\, b_1 + 2332\, b_2 \bmod (3339)\,. \qquad (5.3.1)$$

Решение можно представить в виде таблицы (рис. 5.3.2). Первая строка и первый столбец таблицы содержат значения остатков, на пересечении находятся результаты решения системы сравнений. Таким образом, в массиве результатов числа меняются от нуля до $\min(m_1 m_2)$ по главной диагонали, затем последовательно возрастают по другим диагоналям.

В разделе 1.3.1 показано, что эти диагонали будут располагаться на поверхности тора. Если существует некоторая погрешность в определении исходных данных, то находится заведомо неправильная величина полной фазы, поскольку соседние числа в таблице решений отличаются значительно.

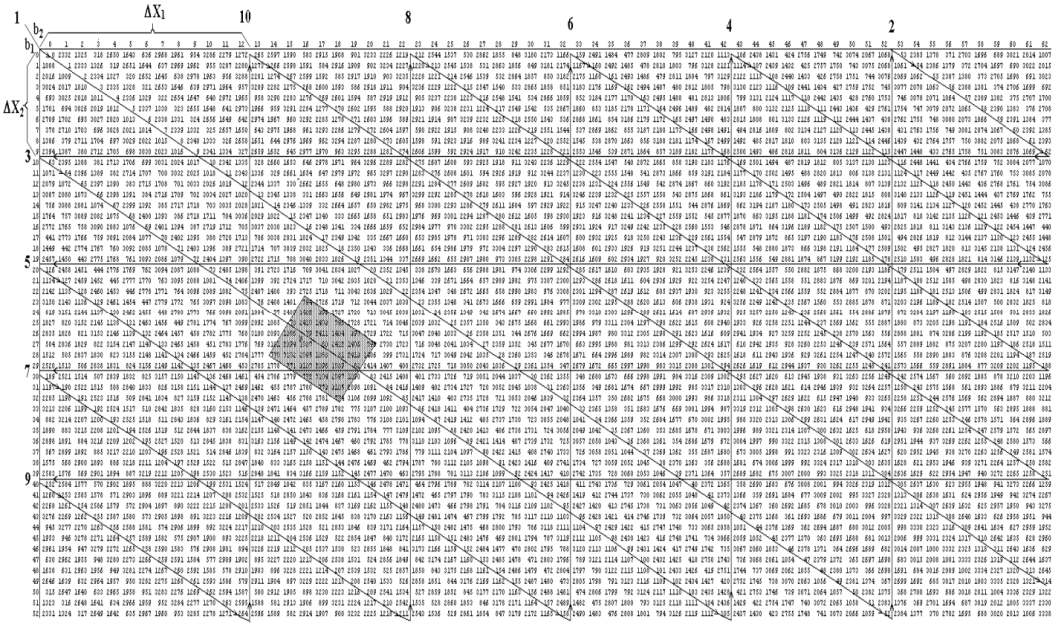



*Рис. 5.3.2.* Таблица решений двух сравнений с модулями 53 и 63

Однако если ограничить максимальный диапазон определения, то витки будут разрежены. Это приведет к тому, что в таблице появятся отстоящие друг от друга диагонали, на которых могут быть расположены правильные решения. На рис. 5.3.2 в порядке возрастания чисел показаны первые десять диагоналей. Переход от одной диагонали к другой обозначен стрелками. Номера витков показаны на рисунке. Первый виток начинается при значении остатков (0, 0) в левом верхнем углу таблицы. В этом случае мы ограничили максимальный диапазон измерения величиной, равной 314.

На этих диагоналях будут лежать числа, попадающие в выбранный диапазон, а значения между диагоналями будут лежать за пределами этого диапазона (рис. 5.3.3).

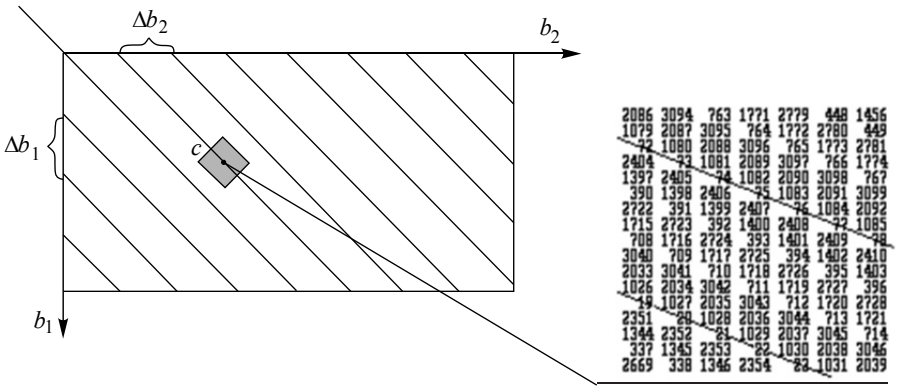



*Рис. 5.3.3.* Схематично представленная таблица решений с модулями $m_1 = 53$ и $m_2 = 63$ (справа увеличенная часть таблицы с численными значениями)

Если у нас есть априорная информация о максимальном размере измеряемой величины, то, ограничив этим значением допустимый диапазон, можно увидеть, что правильные измерения будут располагаться на выделенных диагоналях, а ошибочные данные будут выходить за пределы выбранного нами диапазона.

В разделе 1.3 показано, как двумерную задачу можно свести к одномерной. Для этого необходимо схематично представить таблицу в виде рис. 5.3.4.

На рис. 5.3.2–5.3.4 закрашена область (область грубых промахов), в которую может попасть измеряемая величина при наличии погрешности измерений.

Необходимо каким-либо образом скорректировать эти значения. Это можно сделать двумя способами:

1) полностью игнорировать ошибочные значения;

2) скорректировать их к ближайшей правильной диагонали.

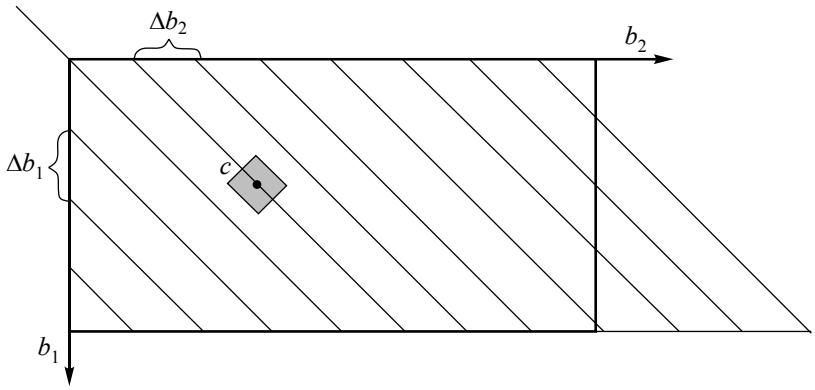



*Рис. 5.3.4.* Продление диагоналей для нахождения решений по одной строке

Лучше скорректировать ошибочные значения. Для этого от двумерной задачи можно перейти к рассмотрению значений только в первой строке. Из рис. 5.3.2 видно, что расстояние между разреженными диагоналями определяется координатами в первой строке. Из выражения (5.2.2) при $b_1 = 0$ получим

$$L \equiv M_2 N_2 b_2 (\text{mod } m_1 \dots m_n) = 2332\, b_2 \bmod (53 \cdot 63)\,. \tag{5.3.2}$$

Из свойства симметричности сравнений это выражение можно переписать в виде

$$M_2 N_2 b_2 \equiv L \bmod (m_1 \dots m_n) \text{ – для } n \text{ модулей,} \tag{5.3.3}$$

$$2332 b_2 \equiv L \bmod (53 \cdot 63) \text{ – для двух выбранных модулей.} \tag{5.3.4}$$

Надо искать $b_2$, которые соответствуют сравнениям (5.4.3 и 5.3.4). Переименовав переменные, получим следующее сравнение первой степени, которое необходимо решить относительно $x$:

$$ax \equiv L \bmod (m). \tag{5.3.5}$$

Вопрос о существовании и количестве решений этого сравнения определяется следующей теоремой [11].

**Теорема**
Пусть НОД $(a, m) = d$ .

- Сравнение $ax \equiv L \bmod(m)$ невозможно, если $L$ не делится на $d$.
- При $L$, кратном $d$, сравнение имеет $d$ решений.

Рассмотрим сравнение (5.3.3). Из китайской теоремы об остатках получим

$$M_2 = \frac{m_1 \ldots m_n}{m_2}. \tag{5.3.6}$$

Поэтому

$$\text{НОД}(a, m) = \text{НОД}(M_2 N_2, m_1 \ldots m_n) = \text{НОД}(N_2, m_1)\frac{m_1 \ldots m_n}{m_2}. \tag{5.3.7}$$

Достаточно ограничиться случаем НОД $(N_2, m_2) = 1$.

Из теоремы следует, что сравнение (5.3.5) имеет решения только при $L$, кратном $d$, т. е.

$$L = k\frac{m_1 \ldots m_n}{m_2}. \tag{5.3.8}$$

Подставляя в (5.3.3) значение $M_2$ и $L$, получим

$$\frac{m_1 \ldots m_n}{m_1} N_1 b_1 \equiv k\frac{m_1 \ldots m_n}{m_1} \bmod(m_1 \ldots m_n). \tag{5.3.9}$$

Сокращаем все части сравнения на $\frac{m_1 \ldots m_n}{m_2}$, получим

$$N_2 b_2 \equiv k \bmod(m_2), \tag{5.3.10}$$

или для нашего случая (4.3.4б)

$$44\, b_2 \equiv k \bmod (63). \tag{5.3.11}$$

Выбирая диапазон $L_D$ такой, чтобы выполнялось условие

$$k \frac{m_1 \ldots m_n}{m_2} < L_D, \tag{5.3.12}$$

находим все $k$, при которых выполняется неравенство (5.2.13), и для каждого из них находим координату в первой строке таблицы решений.

Возьмем $L_D = 317$, значения $k \dfrac{53 \cdot 63}{63} = k \cdot 53 < 317$. Таким образом, $k$ = 0, 1, 2, 3, 4, 5.

Для всех допустимых $k$ решаем сравнение (5.3.11).

| $n$ | Представление |
|---|---|
| 1 | 63 / 44 = 1 · 44 + 19 |
| 2 | 44 / 19 = 2 · 19 + 6 |
| 3 | 19 / 6 = 3 · 6 + 1 |
| 4 | 6 / 1 = 6 · 1 + 0 |

Находим $P_{n-1}$, последовательно вычисляя $P_s = q_s\, P_{s-1} + P_{s-2}$.

| $n$ | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| $q_s$ | – | 1 | 2 | 3 | 6 |
| $P_s$ | 1 | 1 | 3 | **10** | 63 |

Находим решение с помощью выражения $b_2 = (-1)^{n-1} P_{n-1}\, k \pmod{63}$:

$k = 0$ $\quad b_2 = 0;$

$k = 1$ $\quad b_2 = -10 \pmod{63} = 53;$

$k = 2$ $\quad b_2 = -20 \pmod{63} = 43;$

$k = 3$ $\quad b_2 = -30 \pmod{63} = 33;$

$k = 4 \qquad b_2 = -40 \pmod{63} = 23;$

$k = 5 \qquad b_2 = -50 \pmod{63} = 13.$

Таким образом, минимальный разброс между координатами допустимых диагоналей составит 10, и ошибка $\pm 5$ в исходных данных в определении координат может быть скорректирована.

Устранение фазовой неоднозначности для методов интерферометрии и для методов, основанных на проекции синусоидальных картин, одинаковое.

В следующем разделе рассмотрим устранение фазовой неоднозначности только при структурированном освещении.

## 5.4. РАСШИРЕНИЕ ОБЛАСТИ ФАЗОВОЙ НЕОДНОЗНАЧНОСТИ В МЕТОДАХ СТРУКТУРИРОВАННОГО ОСВЕЩЕНИЯ

Работа с расширенным массивом $n[i]$, в котором будут записаны номера допустимых витков, позволяет легко выполнять коррекцию при наличии сбойных (ошибочных) начальных данных.

Для этого необходимо предварительно установить, какой максимальный диапазон имеет измеряемый объект. Пусть, например, это пять витков. Максимальный диапазон равен $5m_1 + 52 = 317$. В этом случае мы можем задать массив $n[i]$, в котором все значения больше пяти будут равны нулю, а также все диагонали и таблицы, начинающиеся с нуля, тоже будут равны нулю. Таким образом, будут подсчитываться только значения точек, попадающих на диагонали (рис. 5.4.1).

| | | | | | | | | | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 | 0 |

*Рис. 5.4.1.* Начальный фрагмент массива $n[i]$ с модулями $m_1 = 53$ и $m_2 = 63$.
Серым цветом показаны диагонали, меньшие или равные пяти

Необходимо скорректировать сбойные точки. Это можно сделать следующим образом. Нулевые промежутки в массиве $n[i]$ заполняем ближайшим значением допустимой диагонали (рис. 5.4.2).

| | | | | | | | | | b2 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |

*Рис. 5.4.2.* Начальный фрагмент массива $n[i]$ с модулями $m_1 = 53$ и $m_2 = 63$,
дополненный допустимыми значениями диагоналей

Значение чисел в таблице можно вычислять так:

$$X = \left[ N_2 (b_2 - b_1) \bmod m_2 \right] m_1 + b_1, \qquad \text{если} \quad b_2 - b_1 \geq 0 \qquad (5.4.1\text{а})$$

$$X = \left[ N_2 (b_2 - b_1 + m_2) \bmod m_2 \right] m_1 + b_1, \qquad \text{если} \quad b_2 - b_1 < 0 \qquad (5.4.1\text{б})$$

Пусть $m_1 = 167$ и $m_2 = 241$ (взаимно простые числа), таблица решений для этого случая показана на рис. 5.4.3. Выбраны первые 13 диагоналей. Области грубых сбоев показаны различным цветом. (Цветной вариант рис. 5.4.3 см. на форзаце.)

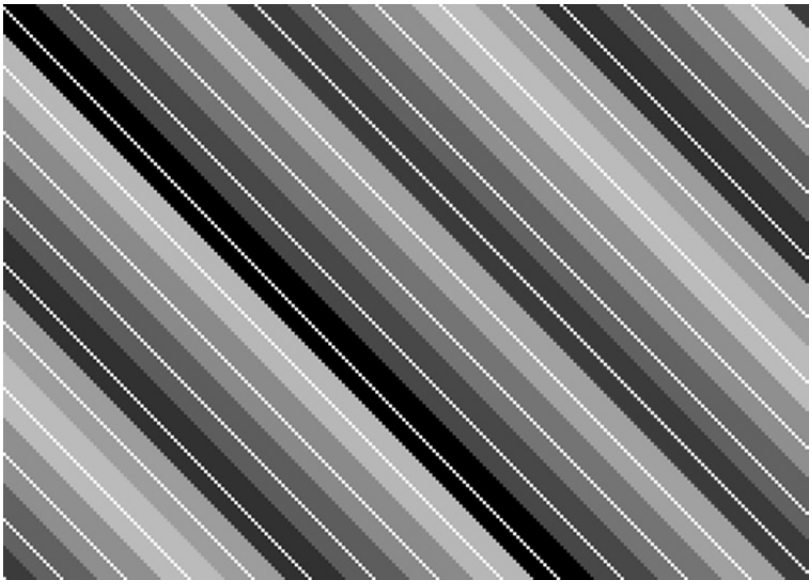


*Рис. 5.4.3.* Таблица решений с модулями $m_1 = 167$ и $m_2 = 241$

На рисунке разным цветом показаны области, при попадании в которые значения $(b_1, b_2)$ корректируются в сторону ближайшей диагонали.

На рис. 5.4.4 показаны два поля фаз одного объекта, полученные при изменении периода полосы ($m_1 = 167$ и $m_2 = 241$). Диапазон фазовых значений меняется от нуля до $2\pi$. Если привести эти значения для левого поля $(0, 241)$ и для правого $(0, 167)$, то каждое значение в точке $(x, y)$ для левого поля будет определять $b_2$, а для правого – $b_1$.

Поскольку $m_1 = 167$ и $m_2 = 241$ взаимно простые числа, то максимальный диапазон измерений может составить 241 полосу по 167 ($241 \cdot 167 = 40\,247$). Если, например, размер полосы, соответствующей 167, имеет размер 1 см, то максимальный размер значений профиля (по координате *Z*) может быть 241 см.

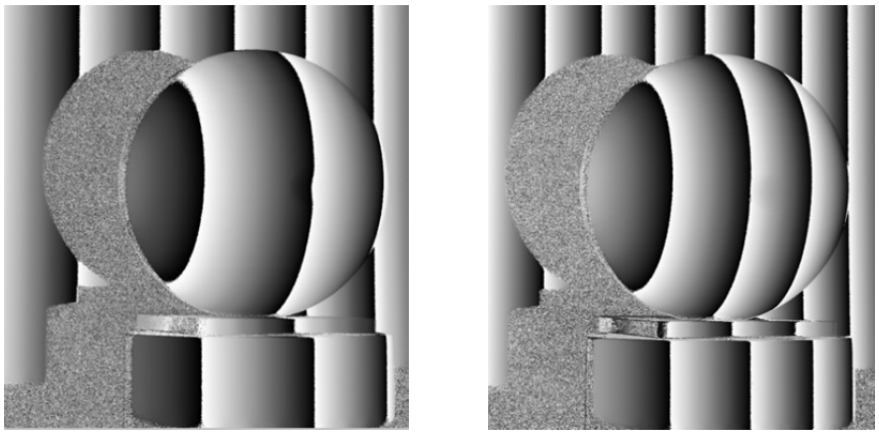


*Рис. 5.4.4.* Два поля фаз одного объекта (шар с выемкой), полученные с периодом $m_2 = 241$ (слева), с периодом $m_1 = 167$ (справа)

Если откладывать значения $b_2$ по координате *X*, а $b_1$ – по координате *Y* для всех точек приведенных полей фаз, то получим таблицу решений (рис. 5.4.5).

*Рис. 5.4.5.* Таблица решений для фазовых значений, показанных на рис. 5.4.4

Из рисунка видно, что из-за погрешности измерения фазовых значений полосы перекрываются, и отделить их достаточно сложно. Однако если ограничить диапазон измерений, то эту проблему можно решить.

Пусть максимальный диапазон составит 13 полос (максимальный размер значений по координате $Z$ – 13 см), таблица решений будет иметь вид, показанный на рис. 5.4.6. Полосы хорошо разделяются, и эти значения можно скорректировать к ближайшей диагонали.

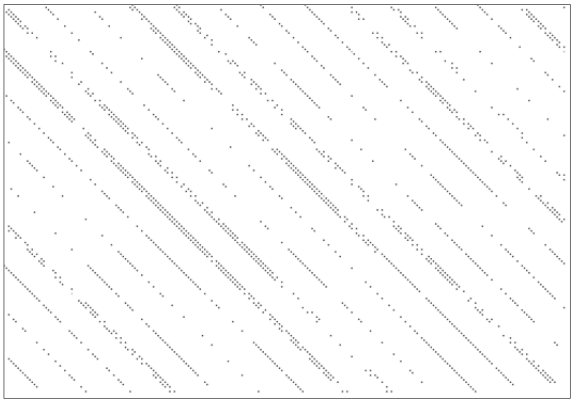
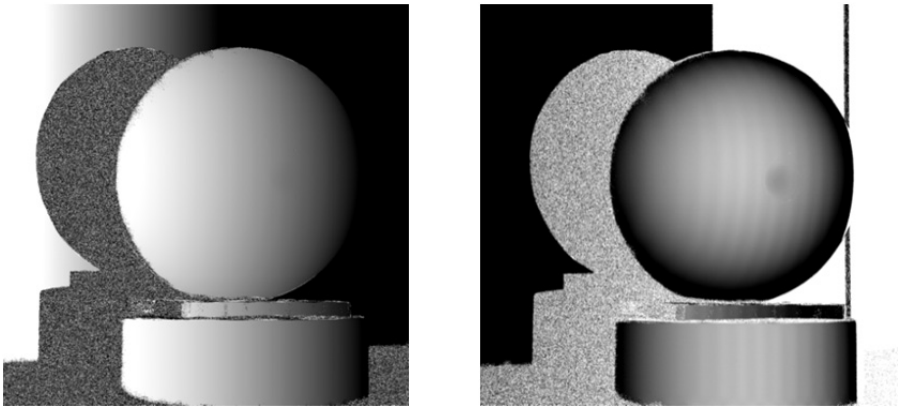


*Рис. 5.4.6.* Таблица решений для фазовых значений, ограниченная 13 диагоналями

Результирующее поле полных фаз показано на рис. 5.4.7.



*а* *б*

*Рис. 5.4.7.* Восстановленное поле полных фаз (*а*); поле полных фаз после устранения наклона (*б*)

В результате мы получили алгоритм, основанный на решении целочисленных систем сравнений, который достаточно устойчив. Время расшифровки определяется только выборкой значений из одномерного массива.

Рассмотрен алгоритм для системы из двух сравнений. Диапазон увеличивается на порядок. Если необходимо большее увеличение диапазона, то необходимо использовать количество модулей больше двух.

# ЗАКЛЮЧЕНИЕ

Использование модульной арифметики позволяет значительно упростить архитектуру вычислительных систем. Однако существует ряд проблем при выполнении арифметических операций и преобразовании чисел из модульной системы представления в позиционную систему и обратно. Решение некоторых задач приведено в первой главе.

Во второй главе представлена история развития компьютерных систем на основе модульной арифметики. Большой интерес к этим системам в 70-х годах XX века привел к появлению целого ряда отечественных и зарубежных компьютеров, которые работали на модульном принципе. К сожалению, затруднения, связанные с модульными вычислениями, привели к снижению интереса к подобным разработкам.

В третьей главе приведены некоторые примеры алгоритмов получения простых чисел и алгоритмов шифрования, основанных на модульной арифметике. В настоящее время благодаря развитию различных систем криптовалют к этим методам привлечено внимание инженерных кадров.

В четвертой и пятой главах приведены решения некоторых задач экспериментальной механики, напрямую не связанные с вычислениями, но использующие описанные в книге методы.

Автор надеется, что проблемы модульной арифметики, изложенные в монографии, привлекут широкие слои специалистов, создающих вычислительные системы для практического использования в своей работе.

## БИБЛИОГРАФИЧЕСКИЙ СПИСОК


1. История развития системы остаточных классов в компьютерной технике / М. Н. Калимолдаев, Р. Г. Бияшев, С. Е. Нысанбаева, Е. Е. Бегимбаева, М. М. Мағзом // Вестник КазНИТУ. – 2017. – № 2. – С. 416–419.

2. *Huang C. H.* A memory compression scheme for modular arithmetic / C. H. Huang, F. J. Taylor // IEEE Transactions on Acoustics, Speech and Signal Processing. – 1979. – Vol. 27 (6). – P. 608–611. – DOI: 10.1109/TASSP.1979.1163312.

3. *Smith W.* Symposium on Very High Speed Computing Technology / W. Smith // IEEE ICASSD Conference. – 1980.

4. *Taylor F. J.* A VLSI residue arithmetic multiplier / F. J. Taylor // IEEE Transactions on Computers. – 1982. – Vol. C-31 (6). – P. 540–546.

5. *Schinianakis D.* Residue number systems in cryptography: design, challenges, robustness / D. Schinianakis, T. Stouraitis // Secure System Design and Trustable Computing. – Springer, 2016. – P. 115–161.

6. An RNS implementation of an *Fp* elliptic curve point multiplier / D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas, T. Stouraitis // IEEE Transactions on Circuits and Systems I: Regular Papers. – 2009. – Vol. 56 (6). – P. 1202–1213.

7. *Schinianakis D.* A RNS Montgomery multiplication architecture / D. Schinianakis, T. Stouraitis // IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE, 2011. – P. 1167–1170.

8. *Schinianakis D.* Hardware-fault attack handling in RNS-based Montgomery multipliers / D. Schinianakis, T. Stouraitis // IEEE International Symposium on Circuits and Systems (ISCAS). – IEEE, 2013. – P. 3042–3045.

9. *Sousa L.* Combining residue arithmetic to design efficient cryptographic circuits and systems / L. Sousa, S. Antão, P. Martins // IEEE Circuits and Systems Magazine. – 2016. – Vol. 16 (4). – P. 6–32.

10. *Айерлэнд К.* Классическое введение в современную теорию чисел / К. Айерлэнд, М. Роузем. – Москва : Мир, 1987. – 415 с.

11. *Виноградов И. М.* Основы теории чисел / И. М. Виноградов. – Москва ; Ленинград : Гос. изд-во технико-теоретической литературы, 1952. – 180 с.

12. *Кнут Д. Э.* Искусство программирования. Т. 2. Получисленные алгоритмы : пер. с англ. / Д. Э. Кнут. – 3-е изд. – Москва : Вильямс, 2007. – 832 с. – ISBN 978-5-8459-0081-4.

13. *Gushov V. I.* Automatic processing of fringe patterns in integer interferometers / V. I. Gushov, Yu. N. Solodkin // Optics and Lasers in Engineering. – 1991. – Vol. 14 (4–5). – P. 311–324.

14. *Арнольд И. В.* Теоретическая арифметика / И. В. Арнольд. – Москва : Учпедгиз, 1938. – 480 с.

15. *Гильберт Д.* Наглядная геометрия / Д. Гильберт, С. Кон-Фоссен. – Москва : Наука, 1981. – 344 с.

16. *Гужов В. И.* Использование модулярной арифметики при фазовых измерениях / В. И. Гужов, Е. С. Кабак, И. С. Орлов // Автоматика и программная инженерия. – 2015. – № 1 (1). – С. 97–107.

17. Сравнение чисел и анализ переполнения в модулярной арифметике / В. И. Гужов, И. О. Марченко, Е. Е. Трубилина, Д. С. Хайдуков // Системы анализа и обработки данных. – 2021. – № 3 (83). – С. 75–86. – DOI: 10.17212/2782-2001-2021-3-75-86.

18. *Бухштаб А. А.* Теория чисел / А. А. Бухштаб. – Москва : Просвещение, 1966. – 384 с.

19. *Грегори Р.* Безошибочные вычисления. Методы и приложения / Р. Грегори, Е. Кришнамурти. – Москва : Мир, 1988. – 208 с.

20. *Шнайер Б.* Прикладная криптография: протоколы, алгоритмы, исходные тексты на языке Си = Applied Cryptography. Protocols, Algorithms and Source Code in C / Б. Шнайер. – Москва : Триумф, 2002. – 816 с. – ISBN 5-89392-055-4.

21. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн. – 2-е изд. – Москва : Вильямс, 2005. – 1292 с.

22. *Valach M.* Vznik kodu a číselné soustavy zbytkových tříd // Stroje na Zpracováni Informaci. – 1955. – Sb. 3. – S. 211–245.

23. *Малашевич Б. М.* Система остаточных классов и модулярные супер-ЭВМ // История отечественной электронной вычислительной техники. – Москва : Столичная энциклопедия, 2014. – С. 179–201.

24. *Акушский И. Я.* Машинная арифметика в остаточных классах / И. Я. Акушский, Д. И. Юдицкий. – Москва : Советское радио, 1968. – 439 с.

25. *Jullien G. A.* Residue number scaling and other operations using ROM arrays / G. A. Jullien // IEEE Transactions on Computers. – 1978. – Vol. C-27 (4). – P. 325–336.

26. Exploiting residue number system for power-efficient digital signal processing in embedded processors / R. Chokshi, K. S. Berezowski, A. Shrivastava, S. J. Piestrak // Proceedings of the CASES '09, Grenoble, France. – ACM, 2009. – P. 19–28.

27. *Schönhage A.* Schnelle multiplikation grosser zahlen / A. Schönhage, V. Strassen // Computing. – 1971. – Vol. 7 (34). – P. 281–292.

28. *Fürer M.* Faster integer multiplication / M. Fürer // Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, 11–13 June 2007. – ACM, 2007. – P. 57–66.

29. Fast integer multiplication using modular arithmetic / A. De, P. P. Kurur, C. Saha, R. Saptharishi // Symposium on Theory of Computation (STOC) 2008. – arXiv:0801.1416.

30. The RSA challenge numbers / RSA Laboratories. – URL: https://web.archive.org/web/20061209135708/http://www.rsasecurity.com/rsalabs/node.asp?id=2093 (accessed: 20.09.2023).

31. Сравнение чисел и анализ переполнения в модулярной арифметике / В. И. Гужов, И. О. Марченко, Е. Е. Трубилина, Д. С. Хайдуков // Системы анализа и обработки данных. – 2021. – № 3 (83). – С. 75–86. – DOI: 10.17212/2782-2001-2021-3-75-86.

32. *Николенко С.* Разложение чисел на множители / С. Николенко. – Криптография – АУ РАН, 2011. – URL: https://logic.pdmi.ras.ru/~sergey/oldsite/teaching/aucryp11/08-factoring.pdf (дата обращения: 20.09.2023).

33. *Diffie W.* New directions in cryptography / W. Diffie, M. E. Hellman // IEEE Transactions on Information Theory. – 1976. – Vol. 22 (6). – P. 644–654. – DOI: 10.1109/TIT.1976.1055638.

34. *Rivest R.* A method for obtaining digital signatures and public-key cryptosystems / R. Rivest, A. Shamir, L. Adleman // Communications of the ACM. – 1978. – Vol. 21 (2). – P. 120–126. – DOI: 10.1145/359340.359342.

35. *Gardner M.* A new kind of cipher that would take millions of years to break / M. Gardner // Scientific American. – 1977. – Vol. 237 (2). – P. 120–124. – DOI: 10.1038/SCIENTIFICAMERICAN0877-120.

36. *Menezes A.* RSA public-key encryption / A. Menezes, P. van Oorschot, S. Vanstone // Handbook of Applied Cryptography. – CRC Press, 1996. – Ch. 8.2. – P. 285.

37. *Ян С. Й.* Криптоанализ RSA / С. Й. Ян. – Москва ; Ижевск : НИЦ РХД ; Ижевский ин-т компьютер. исслед., 2011. – 312 с.

38. *Гужов В. И.* Проекционный метод измерения рельефа объекта / В. И. Гужов, С. П. Ильиных, А. И. Уберт // Научный вестник НГТУ. – 2012. – № 1 (46). – С. 23–28.

39. *Гужов В. И.* Методы измерения 3D-профиля объектов: фазовые методы / В. И. Гужов. – Новосибирск : Изд-во НГТУ, 2016. – 83 с.

40. *Гужов В. И.* Оптические измерения. Компьютерная интерферометрия : учеб. пособие / В. И. Гужов, С. П. Ильиных. – 2-е изд. – Москва : Юрайт, 2018. – 258 с. – ISBN 978-5-534-06855-9.

41. *Гужов В. И.* Влияние разрядности при квантовании интенсивности на погрешность определения фазы в системах с управляемым фазовым сдвигом / В. И. Гужов, Б. С. Котарский // Автометрия. – 1990. – № 2. – С. 70–72.

42. *Zhang S.* High-resolution, real-time 3-D shape measurement : A diss. for the Degree of Dr. of philosophy in mechanical engineering / S. Zhang. – Stony Brook University, 2005. – 127 p.

43. *Schmidt-Weinmar H. G.* Spatial distribution of magnitude and phase of optical-wave fields / H. G. Schmidt-Weinmar // Journal of the Optical Society of America. – 1973. – Vol. 63 (5). – P. 547–555.

44. *Ильиных С. П.* Обобщенный алгоритм расшифровки интерферограмм с пошаговым сдвигом / С. П. Ильиных, В. И. Гужов // Автометрия. – 2002. – № 3. – С. 123–126.

45. Универсальный алгоритм расшифровки / В. И. Гужов, С. П. Ильиных, Д. С. Хайдуков, А. Р. Вагизов // Научный вестник НГТУ. – 2010. – № 4 (41). – С. 51–58.

46. Generic algorithm of phase reconstruction in phase-shifting interferometry / V. Guzhov, S. Ilinykh, R. Kuznetsov, D. Haydukov // Optical Engineering. – 2013. – Vol. 52 (3). – P. 030501-1–030501-2.

47. Digital wavefront measuring interferometer for testing optical surfaces, lenses / J. H. Bruning, D. R. Herriott, J. E. Gallagher, D. P. Rosenfeld, A. D. White, D. J. Brangaccio // Applied Optics. – 1974. – Vol. 13 (11). – P. 2693–2703.

48. *Carré P.* Installation et utilisation du comparateur photoélectrique et interférentiel du Bureau International des Poids et Mesures / P. Carré // Metrologia. – 1966. – Vol. 2 (1). – P. 13–23.

49. *Ильиных С. П.* Алгоритмы анализа и расшифровки интерферограмм методом пошагового фазового сдвига / *С. П. Ильиных* // Модельное исследование методов, алгоритмов и средств индуктивного анализа данных в приоритетных отраслях. – Москва : Русайнс, 2020. – С. 46–76. – ISBN 978-5-4365-5376-4.

50. *Карпюк Б. В.* Анализ погрешностей измерения фазы интерферометров с управляемым фазовым сдвигом / Б. В. Карпюк, Ю. Н. Солодкин // Автометрия. – 1992. – № 6. – С. 16–21.

51. Автоматическая калибровка нелинейности интенсивности в проекционных системах / В. И. Гужов, И. О. Марченко, Д. С. Хайдуков, Е. Е. Серебрякова // Инженерный вестник Дона. – 2019. – № 4 (55). – URL: http://ivdon.ru/ru/magazine/archive/n4y2019/5939 (дата обращения: 21.09.2023).

52. *Гужов В. И.* Использование свойств целых чисел для расшифровки интерферограмм системах / В. И. Гужов, Ю. Н. Солодкин // Оптика и спектроскопия. – 1988. – Т. 65, вып. 5. – С. 1123–1128.

53. *Гужов В. И.* Расширение области фазовой однозначности при интерференционных измерениях // Автометрия. – 1998. – № 3. – С. 99–107.

54. Frequency-multiplex Fourier-transform profilometry: a single-shot three-dimensional shape measurement of objects with large height discontinuities andyor surface isolations / M. Takeda, Q. Gu, M. Kinoshita, H. Takai, Y. Takahashi // Applied Optics. – 1997. – Vol. 36 (22). – P. 5347–5354.

55. *Zhong J.* Phase unwrapping by a lookup table method: application to phase maps with singular points / J. Zhong, M. Wang // Optical Engineering. – 1999. – Vol. 38 (12). – P. 2075–2080.

56. *Zhong J.* Absolute phase-measurement technique based on number theory in multifrequency grating projection profilometry / J. Zhong, Y. Zhang // Applied Optics. – 2001. – Vol. 40 (4). – P. 492–500.

57. Spatial pattern-shifting method for complete two-wavelength fringe projection profilometry / C. Lin, D. Zheng, J. Han, L. Bai, Q. Kemao // Optics Letters. – 2020. – Vol. 45 (11). – P. 3115–3118.

58. 3d reconstruction of objects with occlusion and surface reflection using a dual monocular structured light system / K. He, C. Sui, Z. Wang, Y. Liu, C. Lyu // Applied Optics. – 2020. – Vol. 59 (29). – P. 9259–9271.

## ОГЛАВЛЕНИЕ

## CONTENTS

НАУЧНОЕ ИЗДАНИЕ

**Гужов Владимир Иванович**

**МОДУЛЬНАЯ АРИФМЕТИКА ДЛЯ ИНЖЕНЕРОВ**

**Монография**

Редактор *Е.Е. Татарникова*
Выпускающий редактор *И.П. Брованова*
Художественный редактор *А.В. Ладыжская*
Корректор *Л.Н. Киншт*
Компьютерная верстка *А.В. Сухарева*

Подписано в печать 15.12.2023
Формат 70 × 90 1/16. Бумага офсетная
Уч.-изд. л. 11,7. Печ. л. 10,0.
Тираж 3000 экз. (1-й з-д – 1–30 экз.)
Изд. № 102. Заказ № 322

Налоговая льгота – Общероссийский классификатор продукции
Издание соответствует коду 95 3000 ОК 005-93 (ОКП)

Издательство Новосибирского государственного
технического университета
630073, г. Новосибирск, пр. К. Маркса, 20
Тел. (383) 346-31-87
E-mail: office@publish.nstu.ru

Отпечатано в типографии
Новосибирского государственного технического университета
630073, г. Новосибирск, пр. К. Маркса, 20