ISSN 2221-1993 • **Informatics in School** • 2025 • Volume 24 No 1

DOI: 10.32517/2221-1993-2025-24-1-10-17

П. А. Корнилов
победитель конкурса ИНФО-2024 в номинации «40 лет школьной информатике: Олимпиады, конкурсы, проекты по информатике», Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия

Н. П. Федотова
победитель конкурса ИНФО-2024 в номинации «40 лет школьной информатике: Олимпиады, конкурсы, проекты по информатике», Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия

ЗАДАЧИ С МАТЕМАТИЧЕСКИМ СОДЕРЖАНИЕМ НА ОЛИМПИАДАХ И КОНКУРСАХ ПО ИНФОРМАТИКЕ

Аннотация

Математические понятия и алгоритмы присутствуют в базовом и углубленном курсах информатики основной и средней школы, в задачах единого государственного экзамена и задачах олимпиад и конкурсов по программированию. Содержательные линии математики и информатики во многом пересекаются, главным образом в разделах дискретной математики: математическая логика, теория графов, комбинаторика, системы счисления, модульная арифметика и др. В статье приводятся примеры математических задач, которые оказываются полезными при изучении различных разделов курса информатики. Рассматривается использование полуинвариантов и инвариантов при решении математических задач и верификации программ. Приведены примеры задач на нетрадиционные системы счисления, которые используются при создании аппаратного и программного обеспечения персональных компьютеров. Обсуждаются вопросы арифметики остатков, которые связаны с современными стандартами шифрования информации. Большинство задач, включенных в данную статью, рассчитаны на хорошо подготовленных и мотивированных школьников. Они предлагались авторами на различных олимпиадах и конкурсах по информатике для школьников Ярославской области.

Ключевые слова: математические основы информатики, инварианты, верификация программ, арифметика остатков, системы счисления.

Информация об авторах

Корнилов Петр Анатольевич, канд. физ.-мат. наук, доцент, доцент кафедры теории и методики обучения информатике, физико-математический факультет, Ярославский государственный педагогический университет им. К. Д. Ушинского, г. Ярославль, Россия; *e-mail:* kornilovpa@yandex.ru

Федотова Наталья Петровна, канд. физ.-мат. наук, доцент кафедры компьютерной безопасности и математических методов обработки информации, математический факультет, Ярославский государственный университет им. П. Г. Демидова, г. Ярославль, Россия; *e-mail:* natatulik@yandex.ru

P. A. Kornilov
Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia
N. P. Fedotova
Yaroslavl State University named after P. G. Demidov, Yaroslavl, Russia

TASKS WITH MATHEMATICAL CONTENT IN OLYMPIADS AND COMPETITIONS IN INFORMATICS

Abstract

Mathematical concepts and algorithms are present in the basic and advanced informatics courses of primary and secondary schools, in the tasks of the Unified State Exam and tasks of Olympiads and competitions in programming. The content lines of mathematics and informatics overlap in many ways, mainly in the sections of discrete mathematics: mathematical logic, graph theory, combinatorics, number systems, modular arithmetic, etc. The article provides examples of mathematical tasks that are useful when studying various sections of the informatics course. The use of semi-invariants and invariants in solving mathematical tasks and verifying programs is considered. Examples of tasks on non-traditional number systems that are used in creating hardware and software for personal computers are given. The issues of remainder arithmetic related to modern information encryption standards are discussed. Most of the tasks included in the article are created for well-prepared and motivated students. They were proposed by the authors at various Olympiads and competitions in informatics for schoolchildren in the Yaroslavl region.

Keywords: mathematical foundations of informatics, invariants, program verification, remainder arithmetic, number systems.

Information about the authors

Petr A. Kornilov, Candidate of Sciences (Physics and Mathematics), Docent, Associate Professor at the Department of Theory and Methods of Teaching Informatics, Faculty of Physics and Mathematics, Yaroslavl State Pedagogical University named after K. D. Ushinsky, Yaroslavl, Russia; *e-mail:* kornilovpa@yandex.ru

Natalia P. Fedotova, Candidate of Sciences (Physics and Mathematics), Associate Professor at the Department of Computer Security and Mathematical Methods of Data Processing, The Mathematics Faculty, Yaroslavl State University named after P.G. Demidov, Yaroslavl, Russia; *e-mail:* natatulik@yandex.ru

ISSN 2221-1993 • **Информатика в школе** • 2025 • Том 24 № 1

Введение

Информатика вводилась в школы СССР как предмет, где основное содержание составляло изучение программирования. В настоящее время, по мнению некоторых исследователей, главной содержательной линией информатики являются теоретические (математические) основы информатики [1, 8, 14]. Математические понятия и алгоритмы присутствуют в базовом и углубленном курсах информатики основной и средней школы, в задачах единого государственного экзамена и задачах олимпиад и конкурсов по программированию. Содержательные линии математики и информатики во многом пересекаются, главным образом в разделах дискретной математики: математическая логика, теория графов, комбинаторика, системы счисления, модульная арифметика и др. Изучение содержания перечисленных разделов может проходить на разных уровнях, но наиболее полно реализуется при подготовке одаренных и мотивированных школьников для успешного участия в олимпиадах и конкурсах по информатике.

В данной статье рассматривается несколько разделов, в которых математика и информатика переплетаются, поддерживают и развивают друг друга. Большинство примеров и задач, рассматриваемых в работе, взято из различных олимпиад и конкурсов по информатике для школьников, которые авторы проводили в течение последних лет.

1. Инварианты и верификация программ

При решении некоторого класса математических задач применяются полуинварианты и инварианты. ***Полуинвариант*** — это некоторая характеристика системы в целом, которая монотонно изменяется в ходе изучаемых процессов [16]. Полуинварианты помогают доказывать конечность процессов, обосновывают невозможность их зацикливания. ***Инварианты*** — это характеристики системы, которые сохраняются в ходе преобразований системы. С их помощью часто делают вывод о невозможности перехода системы из начального состояния в конечное, а иногда применяют для доказательства корректности преобразований системы.

В информатике полуинварианты и инварианты используют при верификации программ. Например, полуинварианты применяют для доказательства конечности работы циклов с пост- или предусловием while и repeat, а инварианты применяют для доказательства корректности работы программ.

Рассмотрим некоторые примеры. Первую задачу наш авторский коллектив давал на областную олимпиаду по информатике 1993/1994 учебного года в Ярославской области [7].

Задача 1.1.

В лототроне N шаров, помеченных натуральными числами. Каждую секунду из лототрона вываливаются два шара. Если они помечены одинаковыми числами, то один выбрасывается, а второй возвращается обратно без изменений. Если числа разные, то большее из них заменяется на разность и оба шара возвращаются в лототрон.

1) Докажите конечность алгоритма.

2) Определите, чем все закончится и зависит ли результат от случая.

Решение.

В данном случае ответ на первый вопрос достаточно простой. В качестве полуинварианта можно выбрать сумму всех чисел, написанных на шарах в лототроне. Она монотонно убывает, остается целой и не может стать отрицательной. Поэтому описываемый процесс закончится. Понятно, что в этот момент в лототроне останется один шар.

Чтобы ответить на второй вопрос, можно сначала посмотреть на ситуацию с двумя шарами. Тогда процесс из неопределенного превращается в хорошо известный алгоритм Евклида нахождения наибольшего общего делителя (НОД) двух чисел. Это позволяет выдвинуть гипотезу, что и для любого количества чисел в конце на оставшемся шарике будет написан НОД всех чисел, бывших там изначально. Обоснование этой гипотезы производится точно так же, как и обоснование самого алгоритма Евклида. Для этого рассматриваем один шаг процесса и замечаем, что при переходе от пары (a, b) к паре $(a - b, b)$ общие делители чисел не могут как пропасть, так и появиться. При удалении же дубликата числа из набора НОД всех чисел тем более не может измениться.

Задача 1.2.

Определите, что делает данная программа. Ответ обоснуйте.

```
алг X(вещ a, y, нат n)
  арг a, n
  рез y
нач нат k, вещ t, b
  b := 1; t := a; k := n
  нц пока k<>0
    если mod(k, 2) = 0
      то t := t * t; k := div(k, 2)
      иначе b := b * t; k := k - 1
    все
  кц
  y := b
кон
```

Решение.

Для того чтобы появилась гипотеза о том, что делает эта программа, можно пошагово смоделировать ее работу при значениях аргументов, например, $n = 10$ и $a = 2$. Выпишем значения переменных b, t, k, которые меняются в цикле:

№ шага	b	t	k
0	1	2	10
1	1	4	5
2	4	4	4
3	4	16	2
4	4	256	1
5	1024	256	0

Видим, что ответом будет число 1024, которое сразу же помогает выдвинуть гипотезу, что данная программа вычисляет значение a в степени n.

ISSN 2221-1993 • **Informatics in School** • 2025 • Volume 24 No 1

Однако доказывать, что программа всегда работает корректно, какими-то обычными рассуждениями весьма затруднительно. А при использовании инвариантов доказательство будет убедительным. В данном случае, внимательно посмотрев на текст программы или на результаты ее исполнения на нашем примере, мы можем заметить, что неизменной в ходе выполнения цикла остается следующая комбинация изменяющихся параметров: $b \cdot t^k$. Действительно, на каждой итерации цикла в случае четного числа k имеем:

$$b \cdot t^k = b \cdot (t^2)^{k/2},$$

а в случае нечетного k:

$$b \cdot t^k = (b \cdot t) \cdot t^{k-1}.$$

Остается заметить, что в начале исполнения программы эта величина равна $b \cdot t^k = 1 \cdot a^n = a^n$, а в конце работы программы эта же величина равна $b \cdot t^k = y \cdot t^0 = y$. Тем самым, $y = a^n$.

Данный алгоритм имеет различные практические приложения, носит название ***«алгоритм быстрого возведения в степень»*** и применяется не только при возведении чисел в очень большие степени (что требуется в шифровании), но и в ряде других ситуаций. Например, при вычислении произведения двух чисел по очень большому модулю его заменяют на выполнение небольшого количества операций сложения чисел по данному модулю. Также при вычислении далеких членов рекуррентных последовательностей, задаваемых линейными соотношениями (простейший пример — числа Фибоначчи), можно свести задачу к возведению в большую степень некоторой матрицы и воспользоваться для этого данным алгоритмом.

Задача 1.3.

Определите, что делает данная программа. Ответ обоснуйте.

```
алг П(нат a, b, c)
  арг a,b
  рез c
нач нат d, e
  d := b; c := 0
  нц пока a>0
    e := mod(a, 2)
    a := div(a, 2)
    c := c + d * e
    d := d * 2
  кц
кон
```

Решение.

Поступим аналогично решению предыдущей задачи. Сначала рассмотрим работу алгоритма на примере $a = 12$, $b = 7$:

№ шага	e	a	c	d
0	0	12	0	7
1	0	6	0	14
2	0	3	0	28
3	1	1	28	56
4	1	0	84	112

Возникает вполне логичная гипотеза, что этот алгоритм вычисляет произведение двух чисел. Действительно, в Древней Руси именно так вычисляли произведение натуральных чисел до появления алгоритмов умножения столбиком. Сущность старинного русского способа умножения состоит в том, что умножение любых двух чисел сводилось к ряду последовательных делений одного числа пополам (последовательное раздвоение) при одновременном удвоении другого числа. Например, если в произведении $24 \cdot 5$ множимое 24 уменьшить в два раза (раздвоить), а множитель увеличить в два раза (удвоить), т. е. взять произведение $12 \cdot 10$, то произведение остается равным числу 120.

Данный пример тоже относится к алгоритмам, имеющим практическое применение. В некоторых компиляторах он положен в основу умножения натуральных чисел, поскольку операции умножения и целочисленного деления на 2 — это просто сдвиг двоичной записи числа влево или вправо на 1 бит, которые выполняются чрезвычайно быстро.

Мы предоставляем читателю самостоятельно догадаться по записи алгоритма или по рассмотренному примеру, какая именно комбинация меняющихся в цикле параметров является инвариантом, и с помощью него обосновать корректность работы программы.

Рассмотрим задачу, которую авторы предлагали на областном командном чемпионате школьников по программированию в 2022/2023 учебном году.

Задача 1.4.

С парой чисел (x, y) многократно проводят следующую операцию:

$$\begin{cases} x = x\dfrac{\sqrt{3}}{2} + \dfrac{y}{2}; \\ y = y\dfrac{\sqrt{3}}{2} - \dfrac{x}{2}. \end{cases}$$

Требуется выяснить, какими будут два числа после проведения K операций (исходные значения x, y по модулю не превосходят 1000, K принимает значения от 1 до 10^9).

Решение.

Есть несколько подходов к решению данной задачи. Самый простой из них — просто записать данное преобразование в цикле от 1 до K. Но, ввиду достаточно больших ограничений в задаче на значение K, это решение набирало небольшое количество баллов. В условии задачи был приведен пример, в котором показывалось, что из пары чисел (7, 13) после проведения 2022 операций получится пара (–7, –13). Это была подсказка о том, что решение задачи носит математический характер, хотя можно было, используя данную подсказку, довести решение до полного уже и без знания математики. На самом деле, можно заметить, что инвариантом задачи является сумма квадратов чисел. Если после этого интерпретировать задачу геометрически, то можно было понять, что описанное преобразование — это просто поворот вектора. Если участник был знаком с матричной формулой поворота вектора или просто посмотрел выполнение одного шага на примерах, то он понимал, что за один шаг происходит поворот вектора на 30 градусов,

ISSN 2221-1993 • **Информатика в школе** • 2025 • Том 24 № 1

после чего задача становилась элементарной. Если участник не мог интерпретировать задачу геометрически, но был хорошо знаком с алгеброй, то он мог понять, что один шаг представляет собой умножение вектора на одну и ту же матрицу. В этом случае задача сводилась к возведению данной матрицы в большую степень K, для чего можно воспользоваться алгоритмом быстрого возведения в степень, который мы обсуждали в задаче 1.2.

2. Системы счисления

Системы счисления изучаются и в математике, и в информатике. В курсе информатики основное внимание уделяется изучению двоичной системы счисления и родственных ей восьмеричной и шестнадцатеричной систем счисления. Это объясняется тем, что в современных компьютерах вся информация представляется в виде двоичного кода. Такой выбор во многом объясняется физическими принципами хранения и обработки информации, например, есть сигнал — нет сигнала, участок намагничен — не намагничен и т. п. Между тем двоичная система счисления не является самой выгодной в смысле производительности и скорости вычислений.

Для иллюстрации приведем пример задачи, которая является частным случаем так называемой задачи Баше—Менделеева о гирях.

Задача 2.1.

Придумайте минимальный набор гирь, с помощью которого можно на чашечных весах взвесить любое целое число граммов лекарств от 1 до 40 граммов.

Решение.

Один из первых способов решения дает ответ 6 гирь, имея в виду представление чисел от 1 до 40 в двоичной системе счисления, для чего потребуются гири весом 1, 2, 4, 8, 16 и 32 грамма.

На самом деле этот ответ не является оптимальным. Здесь есть красивая нестандартная идея, что гири можно класть на разные чашки весов — и туда, где нет груза, и туда, где он находится. Это идея троично-симметричной системы счисления. В данном примере оказывается достаточным набора всего лишь из 4 гирек весом 1, 3, 9 и 27 граммов, что несложно проверяется непосредственно.

Еще одним преимуществом троично-симметричной системы счисления [11] является то, что при ее применении не возникает проблем с представлением в компьютере отрицательных чисел, для чего в современных компьютерах применяется дополнительный код [4, с. 68]. В троично-симметричной системе для представления отрицательного числа нужно всего лишь инвертировать цифры в записи модуля числа. Интересный исторический факт состоит в том, что в конце 1950-х годов в МГУ была разработана, реализована и запущена в серию ЭВМ «Сетунь», архитектура которой была основана именно на более экономной троично-симметричной системе.

Далее рассмотрим пример задачи о необычной системе счисления, идея которой пришла одному из авторов статьи при решении задачи заключительного этапа ВСОШ по математике.

Задача 2.2.

В стране математиков решили ввести в действие позиционную систему счисления с необычным основанием. Это иррациональное число x, которое является большим корнем уравнения $x^2 + x - 7 = 0$. Оказалось, что все натуральные числа можно представить в этой системе счисления, используя только цифры от 0 до 6, например, число 13 представляется в ней как 116. А как в этой системе представляется число 2024?

Решение.

Пример представления числа 13 в этой системе счисления с помощью цифр от 0 до 6, приведенный в условии задачи, является существенной подсказкой к составлению алгоритма перевода натуральных чисел в эту необычную систему счисления. Он показывает, что 7 единиц некоторого разряда можно заменить на одну единицу следующего и одну единицу следующего через один разряда вместе. Это можно увидеть и алгебраически, если данное в условии уравнение домножить на x в любой степени. Тогда имеем:

$$(x^2 + x - 7) \cdot x^k = 0 \quad \text{или} \quad x^{k+2} + x^{k+1} = 7 \cdot x^k.$$

В таком случае, начав с 2024 единиц в младшем разряде и применяя описанное преобразование переноса в старшие разряды поочередно справа налево, мы получим ответ к задаче: 133634121.

Как нам кажется, эта задача хорошо сочетается с подходом к формированию представления чисел в позиционных системах счисления, изложенных в статье Н. И. Заводчиковой [10], и будет полезна для мотивированных школьников.

Еще одной необычной системой счисления, которая применяется в информатике для проведения большого количества вычислений с большими числами, является ***система остаточных классов*** (СОК). В ней представление натуральных чисел основано на китайской теореме об остатках.

Для представления натуральных чисел в СОК выбирается базис системы — набор взаимно простых чисел $n_1, n_2, \ldots, n_k$, а любое число x представляется в ней как набор остатков $x_1, x_2, \ldots, x_k$ от деления x на числа базиса системы. При этом, по китайской теореме об остатках, доказательство которой было проведено в трактате Цинь Цзюшао «Математические рассуждения в 9 главах», датируемом 1247 годом, представление любого числа в диапазоне от 0 до произведения всех чисел базиса (не включительно) единственно [12]. Далее, арифметические операции в СОК определяются как операции с остатками в модульной арифметике по отдельности.

К преимуществам СОК надо отнести высокую скорость выполнения арифметических операций, особенно операций умножения. По сравнению с операциями умножения в обычных позиционных системах, где каждую цифру надо умножать на каждую и при этом следить за переносами, здесь каждый остаток умножается только на один такой же остаток у другого числа, причем отсутствие переносов позволяет распараллеливать процесс выполнения любой арифметической операции, что также заметно увеличивает производительность системы.

ISSN 2221-1993 • **Informatics in School** • 2025 • Volume 24 No 1

Но следует отметить и недостатки системы. К ним стоит отнести ограничения на обрабатываемые числа, которые закладываются при выборе базиса системы, неудобство сравнения или деления чисел. В течение последних лет ведется большая работа по устранению этих недостатков. Еще одним недостатком является непростой алгоритм получения самого числа по имеющимся остаткам.

Разберем один из вариантов восстановления числа по известным остаткам.

Задача 2.3.

Базис СОК состоит из четырех чисел: 5, 7, 11 и 13. Остатки от деления числа x на них равны 3, 6, 10 и 10 соответственно. Восстановите число x (в диапазоне от 0 до $5 \cdot 7 \cdot 11 \cdot 13 - 1 = 5004$).

Решение.

Будем восстанавливать число по шагам.

На первом шаге возьмем первый остаток 3.

Далее будем перебирать числа $3 + 5 \cdot k$ (чтобы не портить первый из остатков) и увеличивать k, пока не получим правильный второй остаток. При $k = 0$ остаток от деления на 7 равен 3, при $k = 1$ получаем остаток 1, при $k = 2$ получаем нужный остаток 6, а текущее число 13.

Теперь нам надо сохранять два полученных остатка, поэтому к числу 13 будем прибавлять по $5 \cdot 7 = 35$, пока не получим остаток 10 при делении на 11. У числа 13 остаток 2, у 48 — остаток 4, у 83 — остаток 6, у 118 — остаток 8, а у 153 получаем нужный остаток 10.

Далее надо двигаться с шагом $5 \cdot 7 \cdot 11 = 385$. Но уже при $k = 0$ мы получим нужный остаток, поэтому число 153 и будет ответом к задаче.

3. Арифметика остатков

Данная тема занимает важное место как в теории чисел в математике, так и в информатике. Дело в том, что обычно компьютер, когда работает с целыми числами, отводит под их хранение фиксированный объем памяти. Вследствие этого диапазон чисел, с которыми выполняются действия, ограничен и возникают проблемы с переполнением. Например, тип byte в языке Pascal отводит под хранение неотрицательных целых чисел 1 байт, поэтому значения переменных такого типа могут быть только от 0 до 255. Если же сложить 100 и 200, то получится ответ, который не может быть представлен в типе byte. В таком случае компиляторы вместо числа 300 будут считать ответом число $300 - 256 = 44$, что связано с физической реализацией операции сложения. Аналогично, при вычислении, например, разности $20 - 40$ ответом будет число $-20 + 256 = 236$, а при вычислении произведения $50 \cdot 41$ ответом будет число $2050 - 256 \cdot 8 = 2050 - 2048 = 2$. Тем самым выполнение арифметических операций с типом byte полностью соответствует выполнению операций в арифметике остатков (или вычетов) по модулю 256. В арифметике остатков мы можем привычным образом работать с операциями сложения, вычитания, умножения и возведения в степень. А вот с операцией деления могут возникать проблемы. Важное место, как мы увидим дальше, арифметика остатков занимает и в шифровании информации.

Очень близкой темой по содержанию к данной является тема, связанная с делителями чисел, которая также рассматривается в информатике, и задачи на эту тему даже включаются в ЕГЭ по информатике. Здесь тоже много красивых задач, связанных с основной теоремой арифметики, общим видом делителей, количеством и суммой делителей, признаками делимости и т. п. К сожалению, нам не удастся включить даже некоторые из них в данную статью, но любознательные читатели могут ознакомиться с обзором данной темы в работе [9].

В теории чисел существенную роль в доказательстве нескольких теорем играет следующее несложное утверждение.

Лемма. Пусть p — простое число, a — целое число, которое не делится на p. Тогда среди чисел a, $2 \cdot a$, $3 \cdot a$, …, $(p - 1) \cdot a$, рассматриваемых по модулю p, будут представлены все остатки 1, 2, 3, …, $p - 1$ по одному разу.

Доказательство. Предположим, что среди рассматриваемых чисел есть два числа $m \cdot a$ и $n \cdot a$ с одинаковыми остатками. Тогда их разность $(m - n) \cdot a$ должна делиться на p, но ни один из сомножителей не может делиться на p, значит, и их произведение согласно основной теореме арифметики тоже не будет делиться на p. Следовательно, все остатки разные. Но при этом среди них нет остатка 0 по той же причине. Лемма полностью доказана.

Заметим, что требование простоты числа p здесь важно. Для любого составного числа p лемма неверна. Например, при вычислении по модулю 12 мы можем получить $3 \cdot 4 = 2 \cdot 6 = 8 \cdot 6 = 0$, а $5 \cdot 6 = 1 \cdot 6 = 3 \cdot 6 = 6$. Получается, что арифметика вычетов по модулю p «плохая», когда число p составное. Там нельзя сделать вывод, что если произведение равно 0, то один из сомножителей равен 0, нет операции деления. Если же число p простое, то из леммы следует, что у каждого ненулевого элемента есть обратный, так как в каждой строке таблицы умножения присутствуют все числа от 1 до $p - 1$. Это позволяет определить операцию деления и работать с остатками так, как мы привыкли работать в обычной арифметике.

Хорошо подготовленным школьникам старших классов можно задать следующий сложный вопрос: а можно ли при каких-нибудь составных числах p так определить операцию умножения на множестве остатков при делении на p, чтобы в каждой строке таблицы умножения были представлены все ненулевые остатки? Этот же вопрос можно поставить перед ними сначала в частном случае, когда $p = 4$. Разумеется, очень сильным ограничением здесь является требование, чтобы мы могли, как обычно, работать с арифметическими выражениями, т. е. чтобы можно было переставлять местами сомножители, выполнять операции умножения нескольких сомножителей в любом порядке, выносить общий множитель в сумме слагаемых за скобки. Ответ здесь, как ни странно на первый взгляд, положительный. Для чисел вида p^κ (и только для чисел такого вида), где p — простое число, так сделать можно. В частности, это можно делать для арифметик по модулю $256 = 2^8$, $65536 = 2^{16}$, … . Возникающие при этом «хорошие» арифметики носят название поля Галуа и положены в основание современных стандартов шифрования информации — американского стандарта DES (Data Encryption Standart) и российского стандарта ГОСТ 28147-89 [2, 15]. Оба они были раз-

ISSN 2221-1993 • **Информатика в школе** • 2025 • Том 24 № 1

работаны и реализованы в конце 70-х годов прошлого столетия, но до сих пор обеспечивают надежную защиту информации.

Выведем из леммы две важные теоремы [5, с. 96; 6, с. 117].

Малая теорема Ферма. Пусть p — простое число, a — целое число, не делящееся на p. Тогда $a^{p-1} = 1$ по модулю p.

Доказательство. Действительно, рассмотрим произведение всех чисел a, $2 \cdot a$, $3 \cdot a$, ..., $(p-1) \cdot a$. С одной стороны, оно равно $a^{p-1} \cdot (p-1)!$, а, с другой стороны, по лемме оно равно просто $(p-1)!$. В силу того, что числа p и $(p-1)!$ взаимно простые, мы можем сократить равенство на $(p-1)!$ и получить утверждение теоремы.

Теорема Вильсона. Пусть p — простое число. Тогда $(p-1)! = -1$ по модулю p.

Доказательство. По лемме у каждого элемента есть ровно один обратный к нему элемент, поэтому мы можем разбить все сомножители, кроме 1 и –1, на пары, каждая из которых дает в произведении единицу. Тогда все произведение равно –1, что и требовалось доказать.

Малая теорема Ферма позволяет сильно упростить процесс вычисления больших степеней в арифметике остатков, а теорема Вильсона позволяет во многих случаях быстро вычислять факториалы больших чисел в арифметике остатков. Рассмотрим два примера, иллюстрирующих применение этих теорем.

Задача 3.1.

Может ли быть простым число $122^{2025} + 2025^{80}$?

Решение.

Это число имеет в десятичной записи несколько тысяч знаков, поэтому проверить его простоту чрезвычайно сложно. Заметим, правда, что современные методы могут справиться с этой задачей за разумное время. По формулировке задания похоже, что ответ отрицательный и нам надо найти какой-нибудь делитель.

Попробуем для решения этой задачи воспользоваться малой теоремой Ферма, чтобы избавиться от больших степеней. Для этого надо подобрать подходящее простое число. Оно должно быть связано либо с числом 2025, либо с числом 80, чтобы помочь нам. Но ни число 81, ни число 2026 простыми не являются. Тогда попробуем посмотреть на число 41 и остатки от деления слагаемых на него. При вычислении первого слагаемого по модулю 41 мы можем заменить число 122 на 40 или на –1. Тогда сразу видим, что $(-1)^{2025}$ дает остаток –1 по модулю 41. А ко второму слагаемому мы можем применить малую теорему Ферма и увидеть, что $2025^{80} = (2025^{40})^2 = 1^2 = 1$ по модулю 41. Тогда в сумме получится остаток 0, что означает, что мы нашли делитель 41 у рассматриваемого числа. Ответ: не может.

Задача 3.2.

Найдите остаток от деления 2025! на 2027.

Решение.

Число 2027 является простым, поэтому мы можем применить теорему Вильсона. $2026! = -1 = 2026$ по модулю 2027. Но $2026 = 2026! = 2025! \cdot 2026$ по модулю 2027. Поскольку числа 2026 и 2027 взаимно простые, мы можем сократить равенство на 2026 и получить ответ $2025! = 1$ по модулю 2027.

Малая теорема Ферма является довольно неплохим инструментом в теории чисел. Поэтому предпринимались попытки доказать ее аналог для составных чисел. Это удалось Леонарду Эйлеру. Он ввел функцию, которая теперь называется в теории чисел функцией Эйлера $\varphi(n)$ и равна количеству натуральных чисел, которые меньше или равны n и взаимно просты с ним. Далее, он доказал аналог леммы и, соответственно, аналог малой теоремы Ферма, а именно: для любого натурального n и числа a, которое взаимно просто с n, выполнено $a^{\varphi(n)} = 1 \pmod{n}$.

Разберем следующую задачу.

Задача 3.3.

Докажите, что для любого натурального n число $n^{84} - n^4$ делится на 20400.

Решение.

Разложим на множители число $20400 = 16 \cdot 3 \cdot 25 \cdot 17$ и рассмотрим делимость на каждый из них по отдельности. Заметим также, что $n^{84} - n^4 = n^4 \cdot (n^{80} - 1)$. Если n — четное число, то первый множитель делится на 16. В противном случае, учитывая, что $\varphi(16) = 8$, имеем $n^{80} - 1 = (n^8)^{10} - 1 = 1 - 1 = 0$ по модулю 16.

Аналогично, вычисляя по модулю 3, имеем, что либо n делится на 3, либо $n^{80} - 1 = (n^2)^{40} - 1 = 1 - 1 = 0$.

То же самое проделаем по модулю 25, с учетом того, что $\varphi(25) = 20$. Либо n делится на 5, либо $n^{80} - 1 = (n^{20})^4 - 1 = 1 - 1 = 0$ по модулю 25.

И, наконец, по модулю 17 либо n делится на 17, либо $n^{80} - 1 = (n^{16})^5 - 1 = 1 - 1 = 0$.

Поскольку рассматриваемое число делится на каждый из множителей, а они взаимно простые, то оно делится и на их произведение, что нам и требовалось доказать.

Функция и теорема Эйлера играют ключевую роль в алгоритме шифрования с открытым ключом RSA (аббревиатура от фамилий Rivest, Shamir и Adleman).

Система RSA используется для защиты программного обеспечения и в схемах цифровой подписи. Также она используется в открытой системе шифрования PGP и иных системах шифрования (к примеру, DarkCryptTC и формат xdc) в сочетании с симметричными алгоритмами.

Впервые идею шифрования с открытым ключом предложили американские специалисты по криптографии Уитфилд Диффи (Whitfield Diffie) и Мартин Хеллман (Martin Edward Hellman) [17]. Схема, предложенная ими, следующая. Два участника конфиденциальной переписки задумывают свои секретные ключи. Далее первый участник выбирает некоторое сообщение g, простое число p, возводит g в степень своего секретного ключа и передает полученное сообщение A вместе со значениями g и p второму участнику. Тот возводит g в степень своего секретного ключа и передает полученное сообщение B первому участнику. Передача сообщений происходит по открытому каналу связи. Затем первый участник возводит B в степень своего секретного ключа, а второй возводит A в степень своего секретного ключа. Они получают одно и то же число K, поскольку оно является результатом возведения g в степень, равную

произведению двух секретных ключей. Именно число K и является общим секретным ключом этих двух участников. При этом, даже зная величины g, p, A, B, посторонний человек не может вычислить величину K. Причина заключается в том, что возведение числа в большую степень по некоторому модулю считается односторонней функцией. Это означает, что, зная основание и показатель, вычислить результат возведения в степень по известному модулю легко (см. хотя бы задачу 1.2), а, зная результат возведения в степень по известному модулю и основание, определить показатель степени (так называемая задача нахождения дискретного логарифма) практически невозможно за разумное время, если числа достаточно большие.

В алгоритме RSA для шифрования используется операция возведения в степень по модулю большого числа n, которое равно произведению двух больших простых чисел p и q. Для дешифрования надо найти обратное число к открытому ключу по модулю $\varphi(n)$, для чего необходимо уметь вычислять функцию Эйлера от данного большого числа, а для этого необходимо знать разложение числа на простые множители. В настоящее время для шифрования используются простые числа p и q, состоящие примерно из 200–300 цифр каждое. Считается, что, зная произведение двух таких чисел, найти сомножители (задача факторизации большого числа) — это неразрешимая задача за разумное время. В то же время вычислить саму функцию произведения двух чисел можно быстро.

Итак, зная число $n = p \cdot q$ и функцию Эйлера

$$\varphi(n) = (p - 1) \cdot (q - 1),$$

участник выбирает число e, взаимно простое с $\varphi(n)$, и подбирает обратное к нему по модулю $\varphi(n)$ число d, т. е.

$$e \cdot d = 1 \pmod{\varphi(n)}.$$

Обычно это делают с помощью расширенного алгоритма Евклида, но для маленьких чисел мы покажем, как это можно сделать с помощью функции Эйлера: $e^{\varphi(\varphi(n))} = 1 = e^{\varphi(\varphi(n))-1} \cdot e$, поэтому можно взять $d = e^{\varphi(\varphi(n))-1}$. Далее участник передает открытый ключ (e, n) тому, от кого он хочет получить сообщение, отправитель возводит свое сообщение S в степень e по модулю n и передает результат R. Получатель возводит полученное сообщение R по модулю n в степень d, которую знает только он и, поскольку $e \cdot d = 1 \pmod{\varphi(n)}$, то

$$R^d = (S^e)^d = (S)^{e \cdot d} = (S)^{k \cdot \varphi(n)+1} = (S^{\varphi(n)})^k \cdot S = 1^k \cdot S = S \pmod{n},$$

значит, он получает исходное сообщение в первой степени, т. е. оригинал сообщения.

Рассмотрим пример задачи для очень маленьких чисел p и q.

Задача 3.4.

Пусть $p = 5$, $q = 7$, $e = 5$. Найдите закрытый ключ d и расшифруйте пришедшее сообщение 33.

Решение.

Имеем:

$$n = 5 \cdot 7 = 35;$$

$$\varphi(n) = (5 - 1) \cdot (7 - 1) = 24.$$

Тогда закрытый ключ:

$$d = 5^{23} = 5 \cdot 5^{22} = 5 \cdot (5^2)^{11} = 5 \cdot 1 = 5 \pmod{24}.$$

Чтобы восстановить сообщение, надо 33 возвести в степень 5 по модулю 35:

$$33^5 = (-2)^5 = (-32) = 3 \pmod{35}.$$

Ответ: было передано сообщение 3.

Заключение

Изучение теоретических основ информатики невозможно без использования серьезного и разнообразного математического аппарата, это относится в равной степени к школьному и вузовскому курсам информатики. Тем более это относится к олимпиадам и конкурсам по информатике и программированию, в которых участвуют наиболее одаренные и подготовленные школьники.

Как мы видели на многих примерах, задачи из информатики часто имеют аналоги в математике, а многие задачи и методы из математики, как оказывается, имеют прикладное значение в информатике.

Мы желаем нашим читателям встретить еще много красивых и полезных задач, в частности, их можно найти в задачниках [3, 7, 13, 16], а также во многих других источниках.

Список источников

1. *Абдуразаков М. М., Лягинова О. Ю., Цветкова О. Н.* Информатика, математика и логика в аспекте межпредметной и метапредметной образовательной связи // Чебышевский сборник. 2021. Т. 22. Вып. 2. С. 373–388. EDN: RUTLFH. DOI: 10.22405/2226-8383-2021-22-2-373-388.

2. *Алферов А. П., Зубов А. Ю., Кузьмин А. С., Черемушкин А. В.* Основы криптографии. М.: Гелиос АРБ, 2002. 480 с.

3. *Алфутова Н. Б., Устинов А. В.* Алгебра и теория чисел. Сборник задач для математических школ. М.: МЦНМО, 2002. 264 с.

4. *Андреева Е. В., Босова Л. Л., Фалина И. Н.* Математические основы информатики. М.: БИНОМ. Лаборатория знаний, 2012. 312 с. EDN: WPNTBT.

5. *Бухштаб А. А.* Теория чисел. М.: Просвещение, 1966. 384 с.

6. *Виленкин Н. Я.* Алгебра и теория чисел. 2-е изд. М.: Просвещение, 1984. 192 с.

7. *Волченков С. Г., Корнилов П. А., Белов Ю. А., Дашниц Н. Л., Никулин В. А., Заводчикова Н. И.* Ярославские олимпиады по информатике. Сборник задач с решениями. М.: БИНОМ. Лаборатория знаний, 2010. 406 с. EDN: SUGHPX.

8. *Журавлев Ю. И.* Фундаментально-математический и общекультурный аспекты школьной информатики // Вопросы образования. 2005. № 3. С. 192–200. EDN: ZUYHMG.

9. *Заводчиков М. А., Заводчикова Н. И.* Элементы теории чисел как математическая основа решения задач на обработку целочисленной информации // Информатика в школе. 2021. № 8. С. 32–36. EDN: HNRMEM. DOI: 10.32517/2221-1993-2021-20-8-32-36.

10. *Заводчикова Н. И.* Использование графической системы разрядов при изучении темы «Системы счисления» // Информатика в школе. 2023. № 2. С. 55–62. EDN: WDDJTC. DOI: 10.32517/2221-1993-2023-22-2-55-62.

11. *Златопольский Д. М.* Об уравновешенной троичной системе счисления // Мир информатики. 2017. № 9. С. 1–4. https://infojournal.ru/wp-content/uploads/2020/03/mir_info-2-2017-1.pdf

ISSN 2221-1993 • **Информатика в школе** • 2025 • Том 24 № 1

12. *Нестеренко А. Ю.* Введение в современную криптографию. Теоретико-числовые алгоритмы. М.: Изд-во Московского государственного института электроники и математики (технический университет), 2011. 190 с.

13. *Сгибнев А. И.* Делимость и простые числа. 2-е изд., стереотип. М.: МЦНМО, 2013. 112 с.

14. *Семенов А. Л.* Современный курс математики и информатики в школе // Содержание образования. 2004. № 1. С. 103–118. EDN: ZOQKOM.

15. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования данных. ГОСТ 28147-89. М.: ИПК «Издательство стандартов», 1989. 28 с.

16. *Шень А. Х.* Программирование: теоремы и задачи. М.: МЦНМО, 2017. 320 с.

17. *Diffie W., Hellman M. E.* New Directions in Cryptography // IEEE Transactions on Information Theory. 1976. Vol. 22. Is. 6. P. 644–654.