# Оглавление

# Введение

**Что такое компьютерная алгебра.** Компьютерная алгебра — область математики, лежащая на стыке алгебры и вычислительных методов. Для нее, как и для любой области, лежащей на стыке различных наук, трудно определить четкие границы. Часто говорят, что к компьютерной алгебре относятся вопросы слишком алгебраические, чтобы содержаться в учебниках по вычислительной математике, и слишком вычислительные, чтобы содержаться в учебниках по алгебре. При этом ответ на вопрос о том, относится ли конкретная задача к компьютерной алгебре, часто зависит от склонностей специалиста.

Термин "компьютерная алгебра" возник как синоним терминов "символьные вычисления", "аналитические вычисления", "аналитические преобразования" и т. д. Даже в настоящее время этот термин на французском языке дословно означает "формальные вычисления".

В чем основные отличия символьных вычислений от численных и почему возник термин "компьютерная алгебра"?

Когда мы говорим о вычислительных методах, то считаем, что все вычисления выполняются в поле вещественных или комплексных чисел. В действительности же всякая программа для ЭВМ имеет дело только с конечным набором рациональных чисел, поскольку только такие числа представляются в компьютере. Для записи целого числа отводится обычно 16 или 32 двоичных символа (бита), для вещественного — 32 или 64 бита. Это множество не замкнуто относительно арифметических операций, что может выражаться в различных переполнениях, например, при умножении достаточно больших чисел или при делении на маленькое число. Еще более существенной особенностью вычислительной математики является то, что арифметические операции над этими числами, выполняемые компьютером, отличаются от арифметических операций в поле рациональных чисел, — более того, для компьютерных операций не выполняются основные аксиомы поля (ассоциативности, дистрибутивности). Эти особенности компьютерных вычислений оцениваются в

терминах погрешности или точности вычислений. Оценка погрешности представляет одну из основных проблем вычислительной математики. Каждую задачу требуется решить с использованием имеющихся ресурсов ЭВМ, за обозримое время, с заданной точностью.

Набор объектов, применяемых в символьных вычислениях, весьма разнообразен, в частности, в них используется значительно большее множество рациональных чисел. Это множество все равно остается конечным, но ограничения на допустимые размеры числа (количество знаков в его записи) связаны обычно с размерами оперативной памяти ЭВМ, что позволяет пользоваться практически любыми рациональными числами, операции над которыми выполняются за приемлемое время. При этом компьютерные операции над рациональными числами совпадают с соответствующими операциями в поле рациональных чисел. Таким образом, снимается одна из основных проблем вычислительных методов — оценка погрешности вычислений.

В компьютерной алгебре вещественные и комплексные числа практически не применяются, зато широко используется алгебраические числа. Алгебраическое число задается своим минимальным многочленом, а иногда для его задания требуется указать интервал на прямой или область в комплексной плоскости, где содержится единственный корень данного многочлена. Многочлены играют в символьных вычислениях исключительно важную роль. На использовании полиномиальной арифметики основаны теоретические методы аналитической механики, они применяются во многих областях математики, физики и других наук. Кроме того, в компьютерной алгебре рассматриваются такие объекты, как дифференциальные поля (функциональные поля), допускающие показательные, логарифмические, тригонометрические функции, матричные кольца (элементы матрицы принадлежат кольцам достаточно общего вида) и другие. Даже при арифметических операциях над такими объектами происходит разбухание информации, и для записи промежуточных результатов вычислений требуется значительный объем памяти ЭВМ.

Ограничения на алгоритмы решаемых компьютерной алгеброй задач накладываются имеющимися ресурсами ЭВМ и обозримостью времени счета. Однако ограничения по времени счета и по используемой памяти в символьных вычислениях существенно более обременительны, чем в вычислительных методах.

В научных исследованиях и технических расчетах специалистам приходится гораздо больше заниматься преобразованиями формул, чем собственно численным счетом. Тем не менее, с появлением

ЭВМ основное внимание уделялось автоматизации численных вычислений, хотя ЭВМ начали применяться для решения таких задач символьных преобразований, как, например, символьное дифференцирование, ещё в 50-х годах прошлого века. Активная разработка систем компьютерной алгебры началась в конце 60-х годов. С тех пор создано значительное количество различных систем, получивших различную степень распространения; некоторые системы продолжают развиваться, другие отмирают, и постоянно появляются новые.

**Системы компьютерной алгебры.** Системы компьютерной алгебры можно условно разделить на системы общего назначения и специализированные. К системам общего назначения относятся MACSYMA, REDUCE, MATEMATICA, MAPLE, AXIOM и другие системы. В 80-е годы прошлого века широкое распространение в СССР получила система REDUCE. Она первоначально предназначалась для решения физических задач, разрабатывалась на наиболее широко распространенных компьютерах, разработка до определенного времени не носила коммерческого характера (система до конца 80-х годов распространялась бесплатно), открытый характер системы позволил привлечь к ее разработке огромную армию пользователей, обогативших систему многочисленными пакетами для решения отдельных задач. MACSYMA, так же, как и REDUCE, является "старой" системой. В отличие от системы REDUCE, MACSYMA разрабатывалась с самого начала как коммерческий продукт. В ней более тщательно проработаны алгоритмические вопросы, ее эффективность существенно выше, но меньшее ее распространение можно объяснить двумя обстоятельствами: длительное время она была реализована только на малом числе "экзотических" компьютеров и распространялась только на коммерческой основе. Система MAPLE, созданная в 80-х годах прошлого века в Канаде, с самого начала была задумана как система для персональных компьютеров, учитывающая их особенности. Она развивается "вширь и вглубь", даже ее ядро переписывалось с одного алгоритмического языка на другой. В настоящее время MAPLE широко применяется во многих странах (в частности, в США и Канаде) в учебном процессе, а также в различных областях научных и технических исследований. В конце прошлого века получила широкое распространение и сейчас быстро развивается система MATEMATICA. Ее успех в значительной степени объясняется ее широкими графическими возможностями, а также электронной документацией, которую можно рассматривать как

электронную библиотеку, посвященную различным разделам математики и информатики. Особое место среди систем компьютерной алгебры занимает система AXIOM. В отличие от остальных систем, представляющих собой пакеты программ, общение с которыми осуществляется на некотором алголоподобном языке, система AXIOM, развившаяся из системы SCRATCHPAD-II, имеет дело с более привычными для математиков объектами. В частности, в ней ключевым понятием является понятие категории: здесь можно рассматривать, например, категории множеств, полугрупп, дифференциальных колец, левых модулей и т. д. Система имеет высокую степень универсальности, требует для своей реализации мощных компьютеров, распространяется за достаточно высокую плату, поэтому используется только в ограниченном числе мощных университетских и научных центров.

Специализированные системы отличаются более высокой эффективностью, но область их применения ограничена. К специализированным системам относятся такие системы, как CALEY и GAP — специализированные системы для вычислений в теории групп, MACAULEY, CoCoA, Singular — системы разной степени универсальности для вычислений в кольце многочленов, SCHOONSHIP — специализированная система для вычислений в физике высоких энергий, muMATH и ее правонаследница DERIVE — системы, широко используемые в учебном процессе (в частности, в Австрии лицензия на установку системы DERIVE приобретена для всех средних школ), и многие другие.

**Алгоритмы компьютерной алгебры.** Компьютерная алгебра имеет дело с алгоритмами, которые существенно отличаются от алгоритмов, используемых в вычислительной математике. Алгоритмы вычислительной математики должны давать возможность решать задачу с требуемой точностью при заданных вычислительных ресурсах, наиболее существенными из которых являются ограничения по используемой памяти и времени счета. В компьютерной алгебре вычисления обычно производятся без округления, анализу сходимости уделяется значительно меньше внимания, но используется значительно более широкий набор математических, в первую очередь алгебраических, объектов сложной структуры, и ограничения по времени счета, а особенно по используемой памяти, становятся гораздо более обременительными.

Применение компьютеров в алгебраических исследованиях поставило перед специалистами ряд новых задач и в то же время заставило заново пересмотреть задачи, считавшиеся решенными полностью и окончательно. В частности, к ним относились задачи, для которых был предложен метод, позволяющий решать их "за конечное число шагов". При этом методы решения конкретных задач обычно отличались большим разнообразием, и универсальные методы для конкретных вычислений практически не использовались. С применением компьютеров для алгебраических вычислений потребовалось реализовать универсальные алгоритмы в виде программ для ЭВМ, и оказалось, что они позволяют решать только очень небольшие задачи. С увеличением размера задачи резко возрастало время счета и необходимая память компьютера. Это сделало актуальным поиск более эффективных алгоритмов решения алгебраических задач.

В конце прошлого века бурно развивались исследования в трех областях компьютерной алгебры — теории базисов Грёбнера, факторизации многочленов и интегрирования в конечном виде. Именно эти вопросам, в первую очередь, посвящено настоящее пособие.

Оно начинается с рассмотрения проблемы представления данных, которую можно сформулировать в следующем виде. *Имеется множество объектов $T$ и на нем отношение эквивалентности $\sim$. Требуется в каждом классе эквивалентных объектов выбрать единственного представителя этого класса*, для основных алгебраических областей: кольца целых чисел, поля рациональных чисел, конечных полей, кольца многочленов и поля рациональных функций, алгебраических и трансцендентных расширений полей. Кроме того, в главе 1 рассматриваются арифметические операции в этих областях.

Отдельная глава (глава 2) посвящена алгоритмам вычисления наибольших общих делителей целых чисел и многочленов. Здесь же приводятся некоторые оценки для коэффициентов делителя многочлена от одной переменной.

Задача представления данных для факторколец кольца многочленов приводит к введению понятия базисов Грёбнера полиномиальных идеалов. Рассматривая образующие полиномиального идеала как систему нелинейных алгебраических уравнений, базис Грёбнера можно трактовать как некоторую каноническую форму этой системы. Для случая многочленов от одной переменной над некоторым полем в качестве такой "канонической формы" можно рассматривать наибольший общий делитель этих многочленов, который может быть

получен, например, с помощью алгоритма Евклида. Для системы линейных уравнений от многих переменных в качестве "канонической формы" можно взять диагональную форму системы (т. е. систему вида $y_i = c_i$, $i = 1, \ldots, n$), которая может быть получена с помощью метода Гаусса. В общем случае алгоритмы построения базиса Грёбнера можно считать обобщением алгоритма Евклида и метода Гаусса.

Теория базисов Грёбнера рассматривается в главе 3. Изложение не ограничивается случаем полиномиальных идеалов: строится более общая теория, применимая также к подмодулям свободных полиномиальных, дифференциальных и разностных модулей. Наряду с классическими базисами Грёбнера рассматривается их специальный случай, называемый инволютивными базисами.

Базисы Грёбнера имеют многочисленные приложения. В частности, они позволяют определить, совместна ли система нелинейных алгебраических уравнений, и, если система совместна, то определить, сколько эта система имеет решений над алгебраически замкнутым полем (если множество решений бесконечно, то определить размерность многообразия решений). Эти вопросы изучаются в рамках теории размерностных многочленов (многочленов Гильберта). Свойства таких многочленов и алгоритмы их вычисления приведены в главе 4.

Глава 5 посвящена задаче разложения многочленов на неприводимые множители. Мы рассматриваем эту задачу в следующей постановке: дан многочлен $f(x) \in \mathbb{Z}[x]$ с целыми коэффициентами от одной переменной, требуется разложить его на неприводимые множители. С точки зрения "чистого" математика эта задача давно решена полностью и окончательно: получен алгоритм, позволяющий находить требуемое разложение "за конечное число шагов". Один из таких алгоритмов получен в 1882 году Кронекером, чьим именем он и называется в настоящее время, хотя за 100 лет до Кронекера этот алгоритм был известен австрийскому астроному Шуберту. Следующий шаг в исследовании алгоритмов факторизации был сделан только в 60-х годах прошлого столетия, когда был найден достаточно эффективный алгоритм для разложения на множители многочленов с коэффициентами из конечного поля. Использование этого алгоритма в сочетании с леммой Гензеля позволило получить алгоритмы факторизации многочленов с целыми коэффициентами, пригодные для практической реализации. С конца 60-х годов прошлого века появляется большое количество работ по факторизации. Предлагаются усовершенствования алгоритмов, направленные на увеличение

их быстродействия, на расширение области их применения, в частности, рассматривается задача факторизации многочленов от одной и многих переменных с коэффициентами из конечных полей, из полей алгебраических чисел и т. д. Крупным вкладом в теорию факторизации многочленов явилась работа [**24**], позволившая получить алгоритм факторизации, сложность которого оценивается полиномом от степени исходного многочлена.

Наконец, глава 6 посвящена одной из проблем дифференциальной компьютерной алгебры.

Дифференциальные кольца и поля, в которых наряду с арифметическими операциями имеется одна или несколько операций дифференцирования, — это активно исследуемые объекты компьютерной алгебры. Важным частным случаем дифференциальных полей являются поля элементарных функций, получающиеся из поля рациональных функций от одной переменной последовательным присоединением алгебраических элементов, экспонент и логарифмов (таким образом формализуется основная масса формул, с которыми мы привыкли иметь дело в курсе анализа). Алгебраические зависимости, которые могут существовать между образующими таких расширений, описывает так называемая структурная теорема. (В общем случае дифференциальные поля имеют очень сложную структуру и проблема представления данных для них алгоритмически неразрешима.)

Операция дифференцирования легко описывается алгебраически, и ее реализация не представляет трудностей. Гораздо сложнее обстоит дело с обратной операцией — интегрированием. Если $F$ — дифференциальное поле и $f \in F$, то не обязательно существует $g \in F$ такой, что $g' = f$. Задача интегрирования в конечном виде в общем случае формулируется следующим образом. Пусть $\mathcal{A}$ и $\mathcal{B}$ — два класса дифференциальных полей. Требуется построить алгоритм, который для любого элемента $f$ любого дифференциального поля $F$ из класса $\mathcal{A}$ либо находит дифференциальное поле $G$ в классе $\mathcal{B}$ и элемент $g \in G$ такой, что $g' = f$, либо доказывает, что такого элемента не существует ни в каком поле класса $\mathcal{B}$. В классической постановке задачи $\mathcal{A} = \mathcal{B}$ — класс полей элементарных функций. Различные методы интегрирования изучаются в курсе математического анализа, но до недавнего времени они не были оформлены в виде алгоритмов, применимых к широкому классу функций, в частности, эти методы часто позволяли проинтегрировать функцию, если элементарный интеграл у нее существует, но далеко не всегда позволяли доказать отсутствие элементарного интеграла.

Алгоритм интегрирования в конечном виде функций из чисто трансцендентного расширения поля рациональных функций, порожденного экспонентами и логарифмами, был сформулирован в 1969 году Ришем [**29**]. Проверка, принадлежит ли подынтегральная функция данному классу, осуществляется с помощью структурной теоремы. Далее теорема Лиувилля позволяет определить вид элементарного интеграла, если такой существует. Вычисление интеграла или доказательство его отсутствия производится методом неопределенных коэффициентов с использованием рекурсии. Алгоритм интегрирования алгебраических функций (элементов алгебраического расширения поля рациональных функций от одной переменной) был сформулирован Дж. Дэвенпортом [**6**] в 1982 г. Этот алгоритм использует глубокие результаты алгебраической геометрии и весьма сложен для реализации. Дальнейшее развитие алгоритмы интегрирования в конечном виде получили в работах различных математиков, из которых следует отметить Б. Трейгера и М. Бронштейна. Основные направления исследований — повышение эффективности алгоритмов, расширение области их применения и исследование более широкого класса дифференциальных полей, чем поля элементарных функций.

Обобщением задачи интегрирования можно считать задачу нахождения в классе $\mathcal{B}$ дифференциальных полей решений линейных дифференциальных уравнений с коэффициентами из дифференциального поля, принадлежащего классу $\mathcal{A}$. Частный случай этой задачи для уравнения первого порядка приходится решать при интегрировании элементарных функций (в этом случае мы ищем решения в том же поле, в котором лежат коэффициенты исходного уравнения). Алгоритм нахождения решения в классе полей элементарных функций для уравнений второго порядка с коэффициентами из поля рациональных функций был предложен Ковасиком в 1978 году и вскоре был реализован в различных системах компьютерной алгебры. Обобщение алгоритма на уравнения произвольного порядка было получено М. Сингером. Сложность алгоритма Сингера очень высока, и трудно рассчитывать на его реализацию в обозримом будущем (есть реализация для дифференциальных уравнений 3-го порядка). Задачу о разрешимости линейного дифференциального уравнения в элементарных функциях можно разделить на два этапа. Вначале исследуется разрешимость уравнения в классе лиувиллевых функций (лиувиллевы функции получаются путем расширения исходного дифференциального поля последовательным присоединением элементов, являющихся либо алгебраическими, либо интегралами, либо экспонентами интегралов). Вопрос о разрешимости уравнения в

лиувиллевых функциях решается дифференциальной теорией Галуа: линейное однородное дифференциальное уравнение разрешимо в лиувиллевых функциях тогда и только тогда, когда разрешима связная компонента его группы Галуа, являющейся линейной алгебраической группой. Далее задача сводится к уже рассмотренной задаче интегрирования. Рассмотрение задач, сформулированных в этом абзаце, выходит за рамки настоящего курса.

Компьютерная алгебра используется также при нахождении решений дифференциальных уравнений в виде степенных рядов, при анализе устойчивости решений, поиске периодических решений и в других задачах теории дифференциальных уравнений. Широко применяется компьютерная алгебра при построении разностных схем для численного решения дифференциальных уравнений.

**Вопросы для самопроверки.**

(1) Чем отличается компьютерная алгебра от вычислительной математики?
(2) Какие типы чисел применяются в компьютерной алгебре?
(3) Назовите несколько универсальных и специализированных систем компьютерной алгебры. Опишите область применения специализированных систем.
(4) Сформулируйте проблему представления данных.
(5) Для решения каких задач применяется алгоритм Евклида?
(6) Для решения каких задач применяется метод Гаусса?
(7) Какие задачи могут быть решены с использованием базисов Грёбнера?
(8) Для решения каких задач применяется алгоритм Кронекера?
(9) Сформулируйте задачу интегрирования в конечном виде.
(10) Назовите известные вам применения систем компьютерной алгебры.

# Определения и обозначения

Следующие обозначения считаются фиксированными на протяжении всей книги:

$\mathbb{Z}$ — кольцо целых чисел;
$\mathbf{Z}_n$ — кольцо вычетов по модулю натурального числа $n$;
$\mathbb{Z}_+$ — множество неотрицательных целых чисел;
$\mathbb{Q}$ — поле рациональных чисел;
$O_p$ — кольцо целых $p$-адических чисел;
$R_p$ — поле рациональных $p$-адических чисел;
$F$ — произвольное поле;
$F_q$ — конечное поле из $q$ элементов;
$\mathbb{R}$ — поле вещественных чисел;
$\mathbb{C}$ — поле комплексных чисел;
$\mathcal{Z}$ — кольцо гауссовых чисел (вида $a + bi$, $a, b \in \mathbb{Z}$, $i^2 = -1$);
$\mathcal{Q}$ — поле рационально-комплексных чисел;
$\mathcal{L}$ — логическая переменная (типа "да/нет").

Запись алгоритмов осуществляем в форме, по возможности близкой к тем, которые используются в курсе информатики для средней школы и на механико-математическом факультете МГУ в курсе программирования [**2**]. Алгоритм снабжаем именем, за которым в скобках следует список параметров с указанием их типа. В записи алгоритмов // означает, что далее в строке следуют комментарии.

При необходимости вводим новые типы данных, в частности, для коммутативного кольца $R$ с единицей введем типы "многочлен" и "разложение" следующим образом:

```
R[x] — многочлен:
        запись(степень: ℤ+
                коэффициенты: вектор элементов типа R
                              с индексом 0..степень);
        разложение:
        запись(число_множителей: ℤ+
                множители: вектор элементов типа многочлен
                           с индексом 1..число_множителей).
```

Глава 1

# Проблема представления данных

## 1. Задача представления данных

Проблему представления данных можно сформулировать в общем виде следующим образом. *Имеется множество объектов $T$ и на нем отношение эквивалентности $\sim$. Требуется в каждом классе эквивалентных объектов выбрать единственного представителя этого класса.* В такой постановке могут быть сформулированы очень многие математические задачи, например:

(1) В качестве $T$ возьмем множество натуральных чисел $> 1$, каждое из которых можно записать в виде арифметического выражения. В качестве канонического выбираем выражение, которое является произведением простых чисел, расположенных в порядке неубывания. Основная теорема арифметики утверждает, что любое натуральное число представляется в таком виде и такое представление единственно. Таким образом, задача разложения натурального числа на простые множители может рассматриваться как задача представления данных.

(2) Предыдущий пример непосредственно обобщается на любое кольцо с однозначным разложением на множители, элементы которого можно упорядочить. В частности, таковым является кольцо $\mathbb{Z}[x]$, и мы получаем задачу факторизации многочленов, рассматриваемую в главе 5.

(3) Основная теорема алгебры утверждает, что любой многочлен от одной переменной $z$ с комплексными коэффициентами может быть представлен в виде $a \cdot (z - \alpha_1)(z - \alpha_2) \cdots \cdot (z - \alpha_n)$. Таким образом, в этом случае задача представления данных — это задача нахождения всех корней данного многочлена, т. е. задача решения алгебраического уравнения.

1.1. Упражнения. Переформулировать в виде задачи представления данных следующие задачи:

(1) решения системы линейных уравнений с коэффициентами из некоторого поля;
(2) нахождения НОД некоторого множества целых чисел;
(3) нахождения НОД некоторого множества многочленов от одной переменной с комплексными коэффициентами.

Естественно, что в рамках данного курса мы не можем обсуждать проблему представления данных в полном объеме.

Основная цель этой главы — сформулировать и дать некоторое решение задачи для систем многочленов с коэффициентами из некоторого поля (т. е. для систем нелинейных алгебраических уравнений).

Разнообразие структур данных, используемых в компьютерной алгебре, выдвигает задачу представления данных в ЭВМ на первый план. При аналитических вычислениях (вручную или с использованием ЭВМ) используются элементы таких множеств, как кольцо целых чисел $\mathbb{Z}$, поле рациональных чисел $\mathbb{Q}$, кольцо вычетов по некоторому модулю $\mathbb{Z}_n$, кольца многочленов, различные элементарные функции. Объекты этих множеств допускают неоднозначную запись в виде алгебраических выражений, т. е. представление в памяти ЭВМ. Из различных вариантов представления нужно, по возможности, выбрать оптимальное относительно некоторых критериев. Какими же критериями руководствуются математики при выборе представления конкретных элементов?

Наиболее существенным является требование о том, чтобы выбор представления был *каноническим*, т. е. в множестве всех эквивалентных выражений нужно выбрать единственное выражение, которое представляло бы этот класс эквивалентности. При этом предполагается, что известен алгоритм проверки эквивалентности двух выражений. В действительности такой алгоритм имеется не всегда. Это является одной из причин, почему иногда на представления налагаются более слабые требования, чем то, что они канонические.

Одним из таких условий является условие *нормальности*.

Как правило, рассматриваемая структура данных снабжена некоторым набором арифметических операций, часть из которых определена не для всех значений аргументов. В частности, недопустимо деление на нуль. Тем самым нуль приобретает некоторое особое положение, и возрастает значение задачи определения равенства эле-

мента нулю. Представление, в котором все эквивалентные нулю выражения представляются одним и тем же образом (0), называется *нормальным*. Любое каноническое представление является нормальным, но обратное верно не всегда. Однако во многих случаях наличие нормального представления позволяет построить каноническое. Если же рассматриваемая структура данных такова, что в ней имеются, кроме нуля, и другие "особые" элементы, то определение нормального представления должно быть усложнено.

Ключевое для канонического представления понятие эквивалентности объектов может быть самым различным. Могут использоваться различные определения эквивалентности объектов даже на одном и том же множестве. Например, на множестве многочленов с коэффициентами из конечного поля можно рассматривать отношение функционального равенства, а можно — отношение эквивалентности между многочленами, рассматриваемыми как элементы кольца многочленов с коэффициентами из заданного поля.

Другим требованием, предъявляемым к выбору представления, является требование *естественности*. Что понимается под этим требованием? Рассмотрим пример неестественного представления, основанного на *методе Брауна*. Предположим, что мы умеем определять, являются ли два выражения эквивалентными, и что наша ЭВМ обладает неограниченной памятью. В процессе появления выражений мы будем сравнивать каждое новое выражение с уже встречавшимися нам, которые хранятся в памяти ЭВМ. Если среди ранее встречавшихся выражений имеется эквивалентное исходному, то исходное выражение переписывается в форме эквивалентного ему выражения, уже хранящегося в памяти ЭВМ, в противном случае его форма объявляется каноническим представителем в данном классе эквивалентности и запоминается. К преимуществам такого метода следует отнести его универсальность — метод работает всегда, когда есть алгоритм проверки эквивалентности двух выражений. Недостатком метода является его неестественность, т. е. представление конкретного элемента зависит от того, в какой последовательности элементов он появляется (и в каком месте). Представление называется естественным, если представление каждого элемента определяется одними и теми же правилами, не зависящими от того, в какой последовательности появляется этот элемент.

Ниже будут рассмотрены некоторые из основных структур данных, используемых в компьютерной алгебре, и для них рассмотрена проблема представления данных. Хорошо известно, что в общем случае эта проблема неразрешима, т.е существуют отношения эк-

вивалентности, для которых нет алгоритма выбора канонического представителя в множестве эквивалентных выражений, и даже проверки эквивалентности двух выражений.

**1.1. Кольцо целых чисел.** Из курса программирования известно, что целое число может быть представлено в памяти компьютера разными способами, в частности, это представление зависит от того, как оно описано: как величина типа `integer`, или `real`, или `string`. При этом в большинстве языков программирования под целыми числами понимаются числа из весьма ограниченного диапазона: типичный случай — от $-2^{15} = -32768$ до $2^{15} - 1 = 32767$. Системы компьютерной алгебры имеют дело с большими целыми числами, в частности, любая такая система умеет вычислять и выводить в десятичной записи числа вида 1000! (более тысячи знаков).

В данном курсе мы будем рассматривать представление целых чисел в символьном виде и не вдаваться в подробности, какая память отводится для записи одного символа (бит, байт или другая).

Наиболее распространенным является представление целых чисел в позиционных системах счисления. Такая система определяется выбором основания счисления, например, 10. Множество десятичных целых чисел обычно описывается следующим образом:

```
целое число        == <натуральное число>|0|
                      -<натуральное число>
натуральное число  == <значащая цифра> |
                      <значащая цифра> <цифры>
значащая цифра     == 1|2|3|4|5|6|7|8|9
цифры              == <цифра>  |
                      <цифра> <цифры>
цифра              == 0 | <значащая цифра>
```

Выписанное определение целых чисел дает однозначность представления каждого такого числа, и аналогичное определение (только, может быть, с другим основанием) используется в большинстве систем компьютерной алгебры. Пользуясь таким представлением, удобно реализовать арифметические операции над целыми числами. При этом сложение и вычитание являются относительно “дешевыми” операциями, а умножение и деление — “дорогими”. При оценке сложности арифметических операций следует учитывать как стоимость элементарной операции (одноразрядной), так и количество одноразрядных операций для выполнения какого-либо действия над многозначными числами. Сложность умножения и деления обусловлена, в первую очередь, тем, что с ростом длины числа (его записи в

какой-либо системе счисления) количество элементарных операций увеличивается по квадратичному закону, в отличие от линейного для сложения и вычитания. К тому же, то, что мы обычно называем алгоритмом деления многозначных чисел, в действительности основано на переборе (часто весьма значительном) возможной очередной цифры частного, и при этом недостаточно просто воспользоваться правилами деления однозначных чисел. При большом основании системы счисления (часто оно может иметь порядок $2^{30}$) этот способ малоэффективен.

Пусть $A$ — натуральное число (записанное в десятичной системе). Чтобы получить его запись $A = \sum_{i=0}^{d_A} b_i k^i$ в $k$-ичной системе счисления, можно воспользоваться следующим алгоритмом ($[A/k]$ обозначает целую часть числа $A/k$):

**Дано:** $A$ — натуральное число в десятичной системе счисления
$k > 1$ — натуральное число
**Надо:** $A$ — запись числа $A$ в $k$-ичной системе счисления

**Начало**
$i := 0$
**цикл пока** $A > 0$
$b_i := A \pmod k$
$A := [A/k]$
$i := i + 1$
**конец цикла**
$d_A := i - 1$
**Конец**

Для восстановления десятичного числа по последовательности $b_{d_A}, b_{d_A-1}, \ldots, b_1, b_0$ его $k$-ичной записи $\sum_{i=0}^{d_A} b_i k^i$ используется следующий алгоритм:

**Дано:** $k > 1$ — натуральное число
последовательность цифр, представляющих число $A$ в $k$-ичной системе
**Надо:** $A$ — запись числа $A$ в десятичной системе счисления

**Начало**
$A := 0$
**цикл пока** не конец последовательности
$b :=$ очередной элемент последовательности
$A := A * k + b$
**конец цикла**
**Конец**

1.2. Упражнение. Объясните, почему для перевода числа из десятичной системы в $k$-ичную используется деление, а для перевода из $k$-ичной системы в десятичную — умножение.

Перемножая «столбиком» два двузначных числа в десятичной системе счисления, мы выполняем следующие операции:

$$(10a + b)(10c + d) = 100ac + 10(ad + bc) + bd,$$

т. е. 4 операции умножения одноразрядных чисел, 3 операции сложения и 2 операции умножения на степень основания счисления, которые сводятся к сдвигу. При оценке сложности можно учитывать все элементарные операции, не разделяя их по весам (в данном примере мы имеем 9 элементарных операций). Задача оптимизации алгоритма сводится при данном подходе к минимизации общего числа элементарных операций. Можно, однако, считать, что умножение является более «дорогой» операцией, чем сложение, которое, в свою очередь, «дороже» сдвига. Учитывая только наиболее дорогие операции, мы получаем, что *мультипликативная* сложность умножения двузначных чисел «столбиком» равна 4.

В параграфе 5 рассматриваются алгоритмы вычисления наибольших общих делителей и оценивается их сложность.

Рассмотренное представление не является единственным каноническим представлением целых чисел. Как уже отмечалось, для выбора канонического представления можно воспользоваться единственностью разложения натурального числа на простые множители. Такое представление целого числа может быть применено в тех задачах, где используются только операции умножения и деления, так как они становятся очень “дешевыми”, однако несоизмеримо возрастает стоимость операций сложения и вычитания, что препятствует использованию подобного представления. В некоторых задачах отказ от канонического представления дает значительный выигрыш в быстродействии, в частности, может использоваться частичное разложение числа на множители. Особенно полезен аналогичный метод при работе не с числами, а с многочленами.

Если известно, что при работе программы все встречающиеся в вычислениях целые числа ограничены по абсолютной величине некоторой заданной константой, то можно использовать для задания таких чисел их систему вычетов по модулям некоторых взаимно простых чисел, произведение которых превосходит упомянутую константу. Вычисления с классами вычетов выполняются, как правило, быстрее, чем арифметика многократной точности. А арифметикой

многократной точности при таком подходе нужно пользоваться только при вводе или выводе информации.

Отметим, что наряду с каноническими представлениями в системах компьютерной алгебры используются и другие представления. В частности, желательно, чтобы наличие или отсутствие знака '+' перед целым числом не влияло на восприятие его компьютером. Таким образом, для положительных чисел получается неоднозначное представление, хотя форма отрицательных чисел определена однозначно.

```
<положительное целое> == <натуральное число> |
                         + <натуральное число>
<отрицательное целое> == - <натуральное число>
```

Другое требование — на восприятие числа не должно влиять наличие нулей перед первой значащей цифрой.

1.3. Упражнения.

(1) Оценить количество одноразрядных умножений, используемых при умножении столбиком $m$-значного числа на $n$-значное.

(2) Показать, что два двузначных числа можно перемножить, используя только 3 умножения однозначных чисел и увеличив число сложений.

(3) Найти алгоритм деления длинных чисел, не требующий большого перебора при нахождении первой цифры частного.

(4) Описать алгоритм перевода натуральных чисел из $m$-ичной системы счисления в $n$-ичную.

(5) В *римской нумерации* для записи чисел используются следующие символы: I — единица, V — пять, X — десять, L — пятьдесят, C — сто, D — пятьсот, M — тысяча. Символ считается отрицательным, если правее него найдется символ большего числа, и положительным в противном случае. Например, число 1948 в этой системе запишется так: MCMXLVIII. Сформулировать алгоритм перевода числа из римской записи в десятичную и обратно. Реализовать полученный алгоритм на одном из алгоритмических языков (например, C). Ограничения на исходные данные: $1 \leqslant N < 3700$, в записи результата ни один символ не должен появляться больше 3 раз.

(6) Сформулировать алгоритм и написать программу сложения натуральных чисел в римской нумерации.

(7) Будем говорить, что мы имеем дело с системой счисления со *смешанным или векторным основанием*, если нам задан вектор из $n$ натуральных чисел $M = (m_1, \ldots, m_n)$ (основание счисления) и запись $K = (k_0, k_1, \ldots, k_n)$ обозначает число $k = k_0 + m_1(k_1 + m_2(k_2 + \cdots + m_n \cdot k_n) \ldots))$. Написать программу, которая по данным (день недели, часы, минуты, секунды) определяет, сколько секунд прошло с начала недели (понедельник, $0, 0, 0) = 0$, и выполняет обратное преобразование.

**1.2. Кольца вычетов и конечные поля.** Кольца вычетов и конечные поля представляют собой наиболее простые объекты с точки зрения задачи представления данных. Каждому элементу такого кольца или поля, состоящего из $n$ элементов, можно сопоставить, например, взаимно однозначно неотрицательное целое число из отрезка $[0, n-1]$. Для колец вычетов — это сопоставление каждому классу вычетов его единственного элемента, лежащего в $[0, n-1]$, при этом арифметические операции над такими "числами" выполняются как операции над целыми числами по модулю $n$. Часто в качестве системы представителей кольца вычетов $\mathbb{Z}/n\mathbb{Z}$ выбирается отрезок $[-(n-1)/2, (n-1)/2]$ при нечетном $n$ и $[-n/2+1, n/2]$ при четном $n$. Арифметические операции $+$, $-$, $*$ реализуются очевидным образом, для реализации операции деления обычно используется *расширенный алгоритм Евклида* (см. § 5).

Хотя элементы конечного поля из $n$ элементов также находятся во взаимнооднозначном соответствии с целыми числами из отрезка $[0, n-1]$, это соответствие не является таким же естественным, в частности, арифметические операции выполняются по более сложным правилам. Чаще используются другие формы представления, например, для записи элементов *простого поля* из $p$ элементов используется система вычетов по модулю $p$, а поле $GF(p^k)$ представляется в виде факторкольца кольца многочленов $\mathbb{Z}_p[x]$ по идеалу, порожденному некоторым неприводимым по модулю $p$ многочленом степени $k$.

Сформулируем основные результаты о конечных полях.

(1) Любая конечная область целостности является полем (следует из взаимной однозначности умножения на любой ненулевой элемент).

(2) Характеристика конечного поля является простым числом (следует из отсутствия делителей нуля).

(3) Любое конечное поле $GF(q)$ характеристики $p$ состоит из $q = p^k$ элементов, где $k$ — натуральное число (поскольку оно является векторным пространством над $\mathbb{Z}/p\mathbb{Z}$, $k$ — размерность этого пространства).
(4) Мультипликативный порядок любого ненулевого элемента поля $GF(q)$ делит $q-1$ (ненулевые элементы образуют по умножению группу порядка $q-1$).
(5) Мультипликативная группа поля $GF(q)$ является циклической, т. е. существует элемент порядка $q-1$ (следует из однозначности разложения на множители многочленов $x^m - 1$ над любым полем).
(6) Любые два конечных поля, содержащих одинаковое число элементов, изоморфны (следует из однозначности поля разложения для многочлена $x^{q-1} - 1$).
(7) $GF(p^k) \subset GF(p^m) \iff k|m$.

Таким образом, существует два принципиально разных подхода к построению канонического представления элементов конечного поля $GF(p^k)$:

(1) (векторное представление) выбрать элемент $x$ такой, что его степени $x^0 = 1, x, x^2, \ldots, x^{k-1}$ порождают наше поле как векторное пространство над простым подполем $\mathbb{Z}/p\mathbb{Z}$, и любой элемент записывать как вектор в этом базисе;
(2) (степенное представление) найти примитивный элемент $\alpha$, порождающий мультипликативную группу этого поля, и любой элемент поля представлять в виде степени элемента $\alpha$.

Отметим, что переход от степенного представления к векторному достаточно прост, а обратный переход (вычисление дискретного логарифма) — очень сложен. Сложность этой задачи используется в криптографии для построения систем кодирования с открытым ключом.

1.4. Упражнения.

(1) Показать, что кольцо вычетов по модулю $p^2$ не изоморфно конечному полю из $p^2$ элементов.
(2) Составить таблицу умножения и деления для колец $\mathbb{Z}_4$ и $\mathbb{Z}_9$ и для полей $GF(4)$ и $GF(9)$.
(3) Для заданной матрицы размера $p^2 \times p^2$, где $p = 2$ или 3, проверить, является ли она таблицей умножения в поле $GF(p^2)$ при какой-либо нумерации элементов этого поля.
(4) Реализовать алгоритм деления в кольце вычетов $\mathbb{Z}_n$ (учесть возможность получения неоднозначного результата).

(5) **Китайская теорема об остатках**. Дано $k$ взаимно простых натуральных чисел $m_i > 1$. Для любого набора из $k$ целых чисел $a_i$, $1 \leqslant i \leqslant k$, найти $a \in \mathbb{Z}$, $0 \leqslant a < \prod_{i=1}^{k} m_i$, такое, что $a \equiv a_i \pmod{m_i}$ для всех $i$ от 1 до $k$.
(6) Обобщить предыдущую задачу на случай, когда числа $m_i$ не обязательно взаимно просты.
(7) Найти все неприводимые над полем $\mathbb{Z}_p$ многочлены степени $n$ ($n$ небольшое).
(8) Найти число неприводимых над полем $\mathbb{Z}_p$ многочленов степени $n$.

**1.3. Рациональные числа.** Множество рациональных чисел $\mathbb{Q}$ определяется как фактормножество множества пар $(a, b) \in \mathbb{Z} \times \mathbb{Z}$, $b \neq 0$, по отношению эквивалентности: $(a, b) \sim (c, d) \iff ad - bc = 0$. Если у нас фиксирована каноническая форма целого числа, то каноническую форму рационального числа мы можем получить, например, выбирая из эквивалентных пар целых чисел $(a, b)$ такую, у которой $b > 0$ и НОД$(a, b) = 1$. Все сказанное выше о представлении целых чисел относится и к представлению рациональных чисел.

Естественно, приведенная выше каноническая форма рационального числа не является единственно возможной. Из школьного курса известно, что любое рациональное число можно представить в виде бесконечной десятичной периодической дроби. Также известно, что любая бесконечная периодическая дробь представляет рациональное число, причем соответствие между рациональными дробями и бесконечными десятичными периодическими дробями не является взаимно однозначным: рациональные числа, знаменатели которых имеют вид $2^n 5^m$, могут быть представлены периодическими дробями с периодами (0) и (9).

1.5. Упражнения. Пусть $m$ — натуральное число, $m > 1$, рассматриваемое как основание системы счисления.

(1) Доказать, что рациональные числа могут быть представлены бесконечными периодическими $m$-ичными дробями, причем неоднозначно.
(2) Доказать, что любая бесконечная периодическая $m$-ичная дробь представляет некоторое рациональное число.
(3) Установить взаимнооднозначное соответствие между множеством рациональных дробей и некоторым подмножеством бесконечных периодических $m$-ичных дробей.
(4) Написать программу перевода рациональных чисел в бесконечные периодические $m$-ичные дроби и обратно.

[ . . . ]