Приложение 1. Системы счисления

Будь благословенно божественное число,
породившее богов и людей.
Пифагор Самосский, VI век до н.э.

Системой счисления (numeral system) называется способ записи (а иногда ещё и наименования) чисел. Нас будет интересовать только запись чисел, сейчас запись десятичных чисел одинакова практически во всех языках, а вот их наименование, конечно, разное. Кроме того, здесь не будут рассматриваться машинные системы счисления, это тема курса по архитектурам ЭВМ.

Сначала отметим, что количество чего-либо можно представить двумя способами: аналоговым (непрерывным) и цифровым (дискретным). Для аналогового представления используется какая-нибудь физическая величина: длина отрезка, напряжение на концах сопротивления, положение стрелок модных часов (совсем без цифр) и т.д. Для цифрового представления используются числа – строки символов («цифр»). Каждая цифра имеет в числе позицию, на которой она находится. Традиционно позиции цифр в числах нумеруется для целой части справа налево, начиная с нуля, а для дробной части слева направо, начиная с −1 (скоро поймём, почему так):

$$a_n a_{n-1} \dots a_1 a_0 , a_{-1} a_{-2} \dots \qquad (1)$$

Итак, запись каждого числа состоит из цифр и, возможно, дополнительных символов. Например, в 10-й системе счисления это цифры `0..9` и знаки `+`, `–` и десятичная точка (ну, или запятая). Цифра называется значащей, если при её удалении из записи числа его величина меняется. Например, в числе `0120` первый ноль незначащий, а остальные цифры значащие, а в числе 0 только одна цифра, и она значащая.

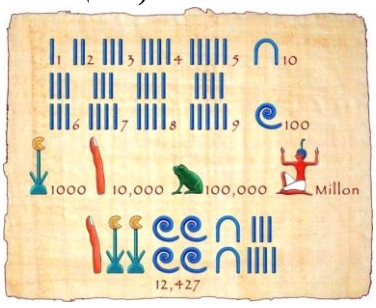
П 1.0. Не позиционные системы счисления

Числа не управляют миром, но они показывают, как управляется мир.
Иоганн Вольфганг фон Гёте

Историки выяснили, что у всех народов сначала появилась письменность, а числа изобрадались словами, например, «три», «двенадцать», «двадцать» и т.д. Затем в языке появилась особые правила записи чисел, т.е. возникла система счисления. Интересно, что часто в качестве цифр использовалась первая буква в написании соответствующего числа, например, в Аттической системе счисления древней Греции это были цифры I – 1, П – 5, Δ – 10, Н – 100, Х – 1000 и М – 10000.

Первой, однако, всюду появилась так называемая унарная (или единичная) система счисления. В этой системе были только неотрицательные целые числа и каждое число «записывалось» нужным числом палочек, зарубок на деревяшке, чёрточек на стене пещеры, узелков на нити и т.д. Таким образом, была только одна «цифра» (отсюда и название). Здесь очень лёгкими были операции сложения и вычитания (хотя, «стирать» при вычитании чёрточки с каменной стены, наверно, было нелегко 😕), а операции умножения и деления чаще всего были «без надобности» и поэтому неизвестны, а при необходимости выполнялись последовательными операциями сложения и вычитания. Далее, никому не приходило в голову как-то по особому изображать число ноль (ни одной единицы, пустое масто на стене 🙄), в этом опять не было необходимости.

Унарная система счисления сейчас используется редко, например, при обучении детей арифметики (счётные палочки), значение костяшки домино, на старых деревянных счётах и т.д.. В математике унарная система счисления применяется в так называемом кодировании Чёрча (интернет Вам в помощь 🙂).



Дальнейшее развитие систем счисления привела к появлению уже нескольких цифр. Так, в древнеегипетской с системе счисления были цифры-иероглифы со значениями | – 1, ∩ – 10, 𓆼 или 𓍢 – 100, 𓆐 – 1000, 𓂭 10000, 𓆈 – 100000 и 𓁨 – 1000000. Неотрицательное число записывалось набором этих цифрами, в любом порядке, например, ||∩ и ∩|| обозначали число 12, хотя обычно «для красоты» цифры записывались в убывающем порядке, а также записывались друг над другом, скажем, число 4321 изображалось так 𓆐𓍢𓍢𓍢∩∩|. Например, слева показано

изображение числа 12427.

Особые иероглифы были для изображения наиболее употребительных дробей, например, так  изображалась дробь 2/3. Любопытно, что текст записывался иероглифами слева-направо, а вот числа, наоборот, справа-налево, как в современном арабском языке. Ясно, что умножение, а тем более деление, были трудными операциями. Отметим, что позже появились и иероглифы для записи и других чисел: 10000000, 10, 20 и т.д. до 90, 200, 300 и т.д. до 900 и другие.

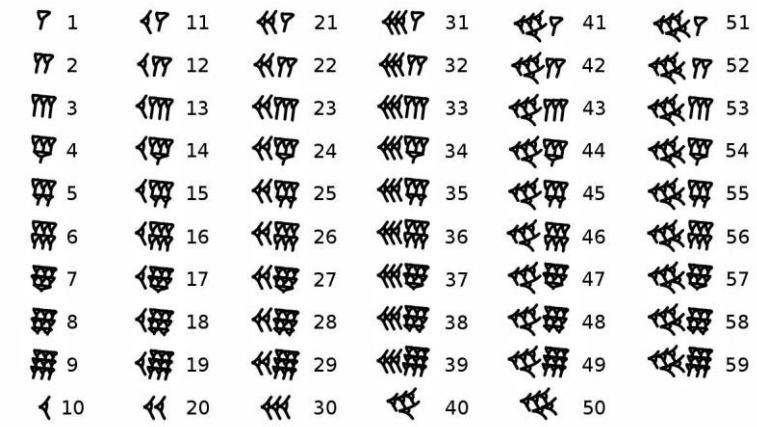
Каждая значащая цифра вносит свой вклад в величину числа. Система счисления называется позиционной, если вклад каждой (значащей) цифры зависит не только от её значения, но и от её позиции в числе, иначе система называется не позиционной. Например, в отличие от нашей 10-й системы, в римской системе счисления в числе XXIV вклад всех цифр, кроме I, не зависит от их позиции, значит это *не позиционная* система счисления.

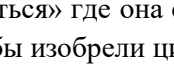
П 1.1. Позиционные системы счисления

Мысль выражать все числа немногими знаками, придавая им, кроме значения по форме, еще значение по месту, настолько проста, что именно из-за этой простоты трудно понять, насколько она удивительна. Как нелегко было прийти к этому методу, мы видим на примере величайших гениев греческой учености Архимеда и Аполлония, от которых эта мысль осталась скрытой.

Пьер Симон маркиз де Лаплас

Первой «почти настоящей» позиционной системой счисления была, наверное, 60-ная система счисления, изобретённая в III тысяцелении до нашей эры шумерами. Само число записывалось в позиционной системе счисления, но каждая 60-ная «цифра» сама была числом (со значением от 0 до 59, и это число записывалось уже «более мелкими» цифрами в непозиционной системе счисления. В качестве этих цифр использовались значки (стоячий клин) – 1 и (лежачий клин) – 10, ниже приведены все цифры от 1 до 59:



Например, вот запись числа $1*60^2+12*60+23=3600+720+23=4343$:  . К сожалению, цифра «ноль» сначала никак не обозначалась и приходилось «догадываться» где она стоит, особенно неприятно это было в средине числа. Понадобилось более 1000 лет, чтобы изобрели цифру ноль  (но только в средине числа). Отзвуки этой системы счисления по-прежнему многочисленны, мы делим час на 60 минут, круг на 360 градусов и т.д.

Отметим, что хорошая система счисления должна обладать двумя не совсем очевидными свойствами:

1). Каждое число должно записываться в системе счисления однозначно.

2). Каждая последовательность цифр обозначает какое-то число, нет «бессмысленных» строк цифр (дополнительные знаки +, – и т.д. должны использоваться только в определённых местах).

«Человеческие» системы счисления свойством однозначности почти всегда не обладают. Например, в общепринятой сейчас десятичной системе считаются равными числа в разной записи:

+2,3≡2,3 02,3≡2,300 и т.д.

Здесь у нас появляются незначащий знак «плюс» и незначащие цифры «ноль», мы говорим, что числа равны *с точностью* до незначащих символов. Неоднозначной является и «римская» система счисления: VIIII≡IX, и система счисления на основе рациональных дробей m/n, например, 1/3≡2/6. В последнем случае мы говорим о *сократимых* и *несократимых* дробях.

При переходе к «компьютерным» системам счисления это уже будет недопустимо. Отводить несколько битовых значений для записи одного и того же числа является расточительством. Ясно также, что если, глядя на запись числа в виде набора бит в памяти ЭВМ, компьютер станет «сомневаться», что это за число, то ничего хорошего не будет. В курсе по архитектурам ЭВМ у Вас будет две целочисленные и одна вещественная машинные системы счисления, в каждой из них набор бит заданной длины будет представлять число однозначно. [1]

Далее мы рассмотрим пример системы счисления, в которой не выполняется второе свойство, т.е. не всякая цепочка цифр будет записью какого-нибудь числа.

Число различных цифр, используемых для записи (неотрицательных целых) чисел называется основанием системы счисления. В математике для точного указания основания его записывают внизу после числа, например, 1056_8 есть число в 8-й системе счисления (*по основанию* 8).

В языках программирования нет подстрочных символов, поэтому там другие правила задания основания чисел. Например, число 255_{10} будет записано в языке Free Pascal как

$FF (16-ая система) &377 (8-ая система) %11111111 (2-ая система)

в языке Ассемблера MASM как

0FFh (16-ая система) 377q (8-ая система) 11111111b (2-ая система)

а в языке C как

0xFF (16-ая система) 0377 (8-ая система) 2-ая система не предусмотрена 😕

В истории человечества широко использовались 10-ая, 12-ая, 20-ая и 60-ая системы счисления. Мы широко используем 60-ую систему счисления для записи времени:

14ч 26м 58с = $\overline{14}\ \overline{26}\ \overline{58}_{60}$

Это 3-хзначное число, правда, значение каждой цифры 60-й цифры записано как десятичное число с надчёркиванием (см. далее), а значение первой цифры не больше, чем 23.

В каждой системе счисления есть правило, как по записи числа вычислить его величину, например, в 10-й системе величина числа (1) вычисляется по простой формуле

$$A=a_n*10^n+a_{n-1}*10^{n-1}+\ldots+a_1*10+a_0+a_{-1}*10^{-1}+\ldots+a_{-K}*10^{-K}=\sum_i a_i\cdot 10^i$$

Эта формула и определяет, что это именно позиционная система счисления.

Будем называть такую систему счисления канонической (обычной), её цифры задают в величине числа геометрическую прогрессию основания. Внедрение 10-й позиционной системы счисления в Европе началось с книги «Liber Abaci» знаменитого математика Леонардо Фибоначчи (Пизанского), постепенно она вытеснила непозиционную римскую систему счисления.

Для не канонических систем эта формула может быть и более сложной, например (для целых чисел)

$A = f_n(a_n)+f_{n-1}(a_{n-1})+\ldots+f_1(a_1)+f_0(a_0)$,

где $f_i(a_i)$ некоторые целочисленные функции от одной переменной, или ещё более сложной

$A = f_n(a_0,a_1,\ldots,a_n)+f_{n-1}(a_0,a_1,\ldots,a_n)+\ldots+f_0(a_0,a_1,\ldots,a_n)$,

где f_i – целочисленные функции многих переменных.

В компьютерных системах счисления всего две цифры "0" и "1", но можно представлять любые числа (положительные и отрицательные, целые и дробные), там тоже «хитрые» формулы вычисления величины числа по его записи, это тема курса по архитектуре ЭВМ.

Как Вы думаете, в какой системе счисления пронумерованы этажи на кнопках в лифте? Здесь можно вспомнить анекдот о программисте, которому надо было ехать на лифте на 12-й этаж. Он сна-

---

[1] К сожалению, в машинной системе счисления для вещественных чисел не всё будет так хорошо. Во-первых, у нас будут два (равных между собой) нуля +0≡−0, кроме того, будет «хитрое» вещественное число NaN (Not a Number), представленное многими битовыми наборами.

чала нажал кнопку с номером 1, затем кнопку с номером 2, а потом долго ругался, так как не мог найти кнопку Enter…

Рассмотрим теперь системы счисления с основанием, отличным от 10. Пусть, например, в системе счисления N цифр (числа записываются по основанию N). Как изображать эти цифры? Можно было бы использовать для записи цифр какие-нибудь экзотические символы α, β, γ и т.д., но это только запутает дело. Договоримся, что для $N \le 10$ оставим обычные 10-е цифры, а для $N>10$ будем записывать цифру как десятичное число с надчёркиванием, например $\overline{10}$, $\overline{11}$ и т.д.

Сначала рассмотрим те системы счисления с основанием N, в которых формула для вычисления значения числа в привычной нам 10-й системе имеет канонический вид

$$A_{10} = (a_n*N^n+a_{n-1}*N^{n-1}+ \dots +a_1*N+a_0+a_{-1}*N^{-1}+a_{-2}*N^{-2}+ \dots)_{10}=\sum_i a_i \cdot N^i$$

где под a_i имеется ввиду 10-ое число, со значением цифры a_i, т.е., скажем, $a_i=\overline{11}=11$.

П 1.2. Перевод чисел из одной системы счисления в другую

Число составляет всю суть каждой вещи.
Платон, V век до н.э.
«Теэтет» Диалоги

– *Что общего между Хэллоуином и Рождеством?*
– *Каждый программист знает, что 31oct=25dec.*

Программистский фольклор

Нам надо научится переводить числа из одной «канонической» системы счисления в другую. Здесь важно понять, что число является объективной реальностью и не зависит от того, в какой бы системе счисления оно не было записано (пять яблок всегда и везде остаются пятью яблоками ⚠). Так что $12_{10}=1100_2=10_{12}=0C_{16}=XII=\dots$

Обратим теперь внимание на такое важное свойство наших «канонических» систем счисления: изменений любой цифры в дробной части числа не может повлиять на величину целой части числа, и наоборот. Вспомним, что в математике этим часто пользуются при решении уравнения, отдельно приравнивая целые и дробные части этого уравнения (из одного уравнения сразу получается два). Поэтому достаточно перевести из одной системы в другую отдельно целую и дробную части, а потом просто соединить их запятой, отделяющей их друг от друга. Аналогично обстоит дело и с отрицательными числами, достаточно перевести в другую систему счисления абсолютную величину числа, а затем просто приписать впереди знак минус.

Итак, научимся переводит из системы счисления с основанием N в систему счисления с основанием K сначала целые (неотрицательные) числа $A_N \to X_K$. При этом нам надо постараться использовать для всех вычислений привычную нам 10-ую систему, таблицу сложения и умножения которой мы знаем с первого класса школы. Выполнять сложение и умножение в другой системе счисления тяжело, надо знать таблицу сложения и таблицу умножения этой другой системы. Та система счисления, которая используется в вычислениях, называется базовой системой счисления. Для использования при вычислениях только 10-й системы, применим хитрость, будем производить перевод по схеме $A_N \to Y_{10} \to X_K$.

Итак, необходимо перевести

$$A_N = Y_{10} = (a_n*N^n+a_{n-1}*N^{n-1}+\dots+a_1*N+a_0)_{10}$$

Здесь нет никаких трудностей, все действия производятся в 10-й системе, например

$$A=1043_5=(1*5^3+0*5^2+4*5+4)_{10}=149_{10}$$

Теперь надо перевести $Y_{10}=X_K=a_n*K^n+a_{n-1}*K^{n-1}+\dots+a_1*K+a_0$, где a_i пока неизвестные цифры в записи числа в системе счисления с основанием K (число этих цифр n+1 тоже пока неизвестно). Разделим обе части этого уравнения на основание K:

$$Y_{10}/K = a_n*K^{n-1}+a_{n-1}*K^{n-2}+\dots+a_1+a_0/K$$

Приравнивая отдельно целую и дробную части уравнения, получаем, что a_0 есть остаток от деления Y_{10}/K. Далее, приравнивая целые части уравнения, сводим задачу к предыдущей, затем получаем a_1 и т.д. Например, $Y_{10}=523=X_6=2231_6$. Аналогично $Y_{10}=526=X_{17}=1\,\overline{13}\,\overline{16}_{17}$ (см. рис. П1.1).

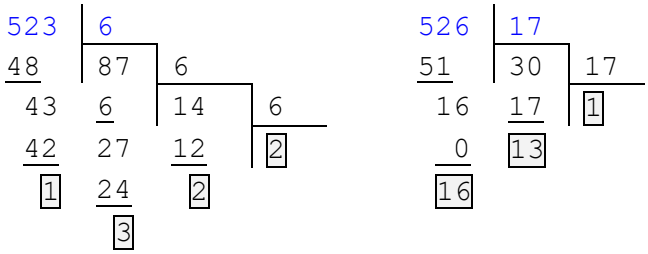


Рис. П1.1. Перевод целой части числа из 10-й в другую систему счисления.

Переведём теперь в систему счисления с основанием K только дробную часть десятичного числа (в записи этой дробной части у нас будет конечное, хотя и заранее неизвестное, число цифр). Итак, надо перевести:

$$A = Y_{10} = a_{-1}*K^{-1}+a_{-2}*K^{-2}+a_{-3}*K^{-3}+\ldots$$

Здесь a_i неизвестные цифры в системе счисления с основанием K. Умножая обе части на основание K получаем:

$$Y_{10}*K = a_{-1}+a_{-2}*K^{-1}+a_{-3}*K^{-2}+\ldots$$

Таким образом, a_{-1} равно целой части от $Y_{10}*K$. Приравнивая затем дробные части уравнения, сводим задачу к предыдущей, снова умножаем обе части уравнения на K и получаем следующую цифру a_{-1} и т.д. Редко когда при переводе получится конечное число цифр после запятой. Из рассмотренного алгоритма можно легко доказать, что получится обязательно *периодическая* дробь (докажите это сами), но период может быть большим. Так сколько же цифр a_i в системе счисления с основанием K имеет смысл получить?

Здесь надо понять следующее. *Целое* число представляет величину точно (два карандаша, четыре землекопа и т.д.), а вот *дробное* число, наоборот, представляет величину приближённо. Действительно, например, длину никогда нельзя измерить точно, это всегда округлённое значение. Так $0,53$ метра округлено либо из $0,534\ldots$, либо из $0,525\ldots$ Впрочем, заметим, что в числе $0,530$ последняя цифра 0 может быть *значащей*, если число округлено до 3-х цифр после запятой либо из $0,5304$, либо из $0,5295$.

Здесь можно возразить, что $0,5$ года это точно 6 месяцев, однако следует заметить, что первые 6 месяцев по длине не совпадают со вторыми... С другой стороны, с точки зрения современной физики, непрерывных величин вообще не существует, так как есть наименьшие кванты времени, пространства, заряда и всех других величин, в единицах этих квантов каждая величина задаётся целым числом. Правда, физика говорит, что эти кванты очень маленькие. Например, квант времени (хронон) равен $5,3910^{-44}$ с., а квант длины (Плантовская длина) равен $1,616^{-35}$ м. Так что из-за крайней малости этих квантов, на это можно не обращать внимания.

Можно сказать, что каждое дробное число записано с погрешностью, равной половине единицы последней цифры этого числа. При переводе чисел из одной системы счисления в другую, естественно, надо потребовать, чтобы новое число имело погрешность, не большую, чем исходное число (можно сказать и наоборот, что его точность должна быть не меньше, чем исходное число). Например, десятичное число $0,53$ имеет погрешность $\varepsilon_{10}=0,01/2$, при переводе в систему счисления с основанием K погрешность $\varepsilon_{10}\geq\varepsilon_K=K^{-P}/2$, где P – число цифр после запятой у числа в системе счисления с основанием K. Например, для числа $0,53$ при переводе в систему счисления с основанием 3 надо $0,01\geq3^{-P}$, отсюда $P=5$.

Так мы и будем поступать при переводе, оставляя в результате только необходимое число знаков после запятой. Здесь надо понять, что брать бóльшее число цифр бессмысленно, они будут не значащие, т.е. просто мусор. Правда следует учесть, что в нашей схеме перевода $A_N=Y_{10}=X_K$ придётся два раза переводить дробную часть числа, поэтому для обеспечения нужной погрешности при первом переводе $A_N=Y_{10}$ лучше взять одну дополнительную цифру в дробной части десятичного числа.

Когда вещественный результат получается через цепочку промежуточных вычислений, то может возникнуть неприятная ситуация, когда на каждой операции мы потеряем около половины последнего разряда. В этом случае ответ будет неточным, чтобы этого избежать, промежуточные результаты проводят с *повышенной точностью*. Вот и мы взяли для этого одну дополнительную десятичную цифру.

Так же может поступать и программист на компьютере. Например, пусть вычисления производятся с 64-разрядными вещественными числами (типа `double`, так называемая двойная точность). В этом случае перед началом вычислений такие числа загружаются на специальные вещественные регистры длиной уже 80 бит, и все вычисления производятся на таких регистрах. После вычислений ответ округляется из 80 снова в 64 бита и записывается в результирующую переменную.

Сейчас вычисление с вещественными числами обычно проводится на специальных векторных регистрах, у которых нет таких дополнительных разрядов. Там для повышения точности приходится идти на разные ухищрения, например, выполнять в одной команде сразу две арифметические операции (см. сноску в конце главы 8 этого пособия). Работа с векторными регистрами описывается в главе 17 пособия для второго семестра по курсу «Архитектура ЭВМ и язык Ассемблера» на сайте `http://arch64.cs.msu.ru`.

Итак, переведём число, меньшее единицы, например, $Y_{10}=0,53=X_3$. Как мы уже определили, надо получить в результате пять верных цифр, следовательно, надо взять шесть цифр и последнюю из них округлить. Оформим «в столбик» последовательные умножения дробного числа на основание (см. рис. П1.2).

Итак, $0,53_{10}=X_3\approx 0,112000_3\approx 0,11200_3$. Ещё один пример $Y_{10}=0,153=X_{31}$. Здесь у нас $\varepsilon_{10}=0,001\geq\varepsilon_{31}=31^{-P}$, отсюда $P=2$ и $0,153=0,4\,\overline{23}\,1_{31}\approx 0,4\,\overline{23}_{31}$.

```
0,53          0,153         0,776
   3             31            19
1,59          4,590        14,744
   3             31            19
1,67         23,033        14,136
   3             31            19
2,01          1,023         2,584
   3
0,03
   3
0,09
   9
0,81
```

Рис. П1.2. Перевод дробной части числа из 10-й в другую систему счисления.

Теперь рассмотрим общий пример $162,53_7=X_{19}$. Сначала переведём число в 10-ую систему, точность $\varepsilon_7=1/49\geq\varepsilon_{10}=10^{-P}$, $P=2+(1$ про запас$)=3$.

$$Y_{10} = 1*7^2+6*7^1+2+5*7^{-1}+3*7^{-2} = 93+38/49 \approx 93,776_{10}$$

потом переведём в систему с основанием 19. Точность $\varepsilon_7=1/49\geq\varepsilon_{19}=19^{-P}$, отсюда $P=2$. Здесь типичная ошибка студентов, они берут $\varepsilon_{10}=0,001\geq\varepsilon_{19}=19^{-P}$, откуда неверное $P=3$.

$$93,776_{10}\approx 58,\overline{14}\;\overline{14}\,2\approx 4\overline{14},\overline{14}\;\overline{14}_{19}$$

Ещё раз напоминаем, что число является объективной реальностью, а в разных системах счисления меняется не само число, а только форма его записи. В математике нет никаких систем счисления! В связи с этим попробуйте ответить на вопрос, есть ли такая (каноническая) система счисления с основанием N, при записи в которой число 2_{10} является делителем числа 9_{10}?

П 1.3. Неканоническая троичная система счисления

Глядя на мир, нельзя не удивляться!
Козьма Прутков.
«Безвыходное положение»

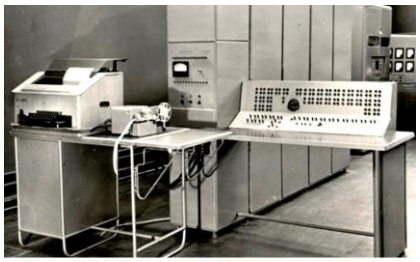
Рассмотрим систему счисления, в которой величина числа вычисляется по другим правилам, чем мы рассматривали ранее. В этой системе счисления три цифры, но, в отличие от «обычной» троичной

системы с цифрами 0, 1 и 2, цифры новой системы счисления обозначим как 0, 1 и $\bar{1}$. Их значения в 10-й системе соответственно равны 0, 1 и −1. Обозначим эту систему как 3^*, тогда, например, число

$$\bar{1}01\bar{1}_{3^*} = (-1)*3^3+1*3+(-1) = -25_{10}$$

Разработайте алгоритм преобразования чисел из 10-й системы счисления в эту систему 3^*.

В этой системе счисления не нужен знак «минус», так как знак числа определяется первой значащей цифрой. По аналогии с разрядами двоичного числа, которые называются битами, разряды троичного числа называются тритами. Отметим, что в этой системе счисления очень просто делается округление чисел: лишние цифры просто отбрасываются, так как они уменьшают и увеличивают значение округляемого числа с одинаковой вероятностью. Такое округление даёт наименьшую среднюю ошибку, исходя из этого, эту систему счисления называют *уравновешенной*. Кроме того, эта система являеися самой экономичной, в ней с помощью фиксированного числа символов можно записать наибольшее количество чисел [1]. Далее, Джон фон Нейман, один из родоначальников теории информации, даже доказал теорему о том, что троичная система счисления позволяет наиболее эффективно кодировать вещественные числа.



ЭВМ Сетунь, 1959 год

Существовали компьютеры, которые работали в такой троичной системе счисления, например, серийная ЭВМ Сетунь, разработанная в нашем университете ещё в 1958 году под руководством Н.П. Брусенцова. Троичная ЭВМ работает не в двоичной логике, где есть только значения **истина** и **ложь** или **true** и **false** (обычно они имеют числовые значения 1 и 0), а в троичной логике, где за числовые значения логических констант можно взять именно значения −1, 0 и 1. Это позволяет решать задачи с меньшим числом сравнений, например, переход на три вычислительные ветви по знаку числа X за два сравнения:

```
if X=0 then goto Equal else
if X<0 then goto Less else goto Greater
```

В языке с троичной логикой можно сделать это за одно сравнение, просто писать что-то вроде

```
if sign(X) then goto (Less,Equal,Greater)
```

Эта так называемая сокращённая (или симметричная) система счисления. Исторически она восходит к средневековой «задаче о взвешивании», в которой требовалось с наименьшим количеством гирь уметь взвешивать предметы от 1 до 40 кг (ну, или фунтов, унций и т.д.). Гири можно класть на обе чаши весов. Решением являются гири весом в 1, 3, 9 и 27, т.е. степени тройки. При помещении на чашу весов с товаром вес такой гири надо было вычитать из веса груза (т.е. вес гири является как бы отрицательным). Поймите, что это и есть описанная выше троичная система счисления, а с помощью гирь мы «записываем» нужное число, равное весу товара. Эта система счисления предложена математиком Фибоначчи ещё в XIII веке.

Любопытно, что отдельный знак «минус» для представления отрицательных чисел не нужен и в системе счисления с неканоническим основанием −2:

$$(a_n a_{n-1}\ldots a_1 a_0, a_{-1}a_{-2}\ldots)_{-2} = \sum_i a_i \cdot (-2)^i$$

Любопытно, что ещё в 1840 году деревянное вычислительное устройство, работающее в такой сбалансированной троичной системе счисления построил английский изобретатель Томас Фаулер. Сохранилось только двухстраничное описание этой машины.

Ещё более необычным основанием системы счисления является `-1+i=-1+sqrt(-1)`. В этой системе счисления без знака «минус» и обозначения мнимой единицы `i` можно записать как вещественные, нак и комплексные числа. В качестве (достаточно трудной) задачи попытайтесь найти алгоритм перевода (вещественных) десятичных чисел в такую систему счисления.

---

[1] См., например, Фомин С.В. «Системы счисления». – М., Наука, 1987, 48с.

П 1.4. Двоично-десятичная система счисления

У всего есть своя красота, но не каждый может её увидеть.

Конфуций, V век до н.э.

Как Вы уже, вероятно, знаете, работа ЭВМ, и, в частности, представление чисел, базируется на двоичной системе счисления. Человеку, однако, много удобнее работать в привычной ему 10-й системе. Таким образом, много задач решаются по такой схеме. Сначала программа вводит много чисел в 10-й системе, преобразуя их в 2-ую систему, потом начинает их обрабатывать. После обработки результаты преобразуются из 2-й системы обратно в 10-ую и представляются человеку.

Перевод в другую систему счисления, как мы уже знаем, содержит многочисленные операции умножения и деления, которые являются трудоёмкими даже для современных ЭВМ. Когда собственно обработка данных занимает мало времени, то большая часть работы состоит в преобразовании чисел из одной системы счисления в другую и обратно. Первые ЭВМ работали медленно, и такие «накладные расходы» на вычисления представляли большу́ю проблему. Тогда и была разработана машинная система счисления, прямой и обратный перевод между которой и 10-й системой был очень лёгким.

Эта система счисления называется смешанной или двоично-десятичной BCD (Binary-Coded Decimal). Основная идея состоит в том, чтобы переводить в другую систему счисления не всё число как единое целое (как происходит в изученных нами алгоритмах перевода), а каждую цифру по отдельности. Другими словами, каждая 10-ая цифра всегда записывается в виде четырёх двоичных разрядов:

```
0=0000   1=0001   2=0010   3=0011   4=0100
5=0101   6=0110   7=0111   8=1000   9=1001
```

Например, число $1074_{10}=10000110010_2=0001000001110100_{2\text{-}10}$. Как видно, двоичное число, вообще говоря, не совпадает с двоично-десятичным.

В этой системе счисления в каждом байте машинной памяти можно хранить не одну, а две доично-десятичных цифры (такая система называется упакованной).

Можно доказать теорему, что для двух оснований систем счисления P и Q (P≤Q) число в системе счисления по основанию P будет совпадать с числом в смешанной системе счисления P-Q только тогда, когда $Q=P^k$. Например, $7013_8=111000001011_2=111000001011_{2\text{-}8}$.

Отметим, что из 16-ти комбинаций для кодирования десятичных цифр заняты только 10. При необходимости можно закодировать знаки «плюс» как 1100, «минус» как 1101 и разделитель между целой и дробной частями числа «запятая» (ну, или точка) как 1110. Тогда можно переводить и знаковые дробные числа, например,

$-4047{,}95_{10}=0100\ 0000\ 0100\ 0111\ 1110\ 1001\ 0101_{2\text{-}10}$

Если договорится и обязательно ставить знак «плюс» или «минус», но не в начале, а в конце числа, то можно записывать в памяти ЭВМ числа переменной длины (знак будет задавать конец числа в памяти ЭВМ). Так, процессоры Intel допускают в такой записи 2-10 чи́сла длиной от одной до 31 десятичной цифры. Отметим, что в новых 64-битных ЭВМ такая система счисления уже не используется, так как эти машины умеют выполнять операции над длинными двоичными 64-битными числами.

Видно, что «цифрами» в 2-10 системе являются не отдельные биты, а только четвёрки битов (тетрады или 16-ричные цифры). Кроме того, отметим плохое свойство этой системы: не каждый набор битов будет допустимым числом, например, наборы 10001010 или 100011010001 не соответствуют никакому допустимому числу.

Рассмотрим теперь общий случай (мы всё же в Университете ☺). Пусть P и Q две системы счисления и их основания P≤Q. Переведём $A_Q=X_{P\text{-}Q}$. Сразу возьмём какоё-нибудь конкретный случай, например $1\overline{25}07_{31}=X_{4\text{-}31}$. Сначала определим, сколько разрядов необходимо для представления в системе счисления с основанием 4 любой цифры системы счисления с основанием 31, для этого возьмём самую большую цифру $\overline{30}$ и переведём её: $\overline{30}_{31} = 1\cdot4^2+3\cdot4^1+2\cdot4^0 = 132_4$, т.е. надо три цифры. Тогда

$1\overline{25}07_{31}=001\ 121\ 000\ 013_{4\text{-}31}$

Все рассмотренные нами системы счисления были упорядоченные, математики говорят, что на них определено отношение полного порядка, т.е. для двух любых чисел всегда можно определить,

что одно из них меньше другого, больше другого, или они равны между собой. Не все системы счисления такие, уже известные Вам комплексные числа не упорядочены, их можно сравнивать между собой на равно и не равно, а на больше и меньше уже не всегда. Удивительно, но такими же свойствами обладает и машинная система счисления для вещественных чисел. Там тоже есть числа, которые нельзя сравнивать между собой, а также так называемые условно-сравнимые числа. Поэтому программистов, которые привыкли бездумно писать что-то вроде

```
if x<y then ...
```

может ждать неприятный сюрприз. По счастью, целые числа на ЭВМ всегда упорядочены, но там программиста ожидает другая неожиданность. Например, там есть две целочисленные системы счисления и, соответственно, две операции сравнения, например, на меньше (<), причём по внешнему виду они неотличимы 🐷, и вполне может быть

$x <_1 y$ = **true**, но $x <_2 y$ = **false** 😕

Впрочем, всё это темы курса по архитектурам ЭВМ.

Вопросы и упражнения

1. В чём преимущество позиционных систем счисления от не позиционных?
2. Верно ли, что число знаков для записи числа совпадает с основанием системы счисления?
3. Как записывать цифры в системах счисления с основаниями, больше 10?
4. Для чего надо уметь записывать числа в разных системах счисления?
5. Что такое точность представления вещественного числа?
6. Для чего нужна двоично-десятичная чистема счисления?