Министерство образования и науки Российской Федерации
Федеральная заочная физико-техническая школа
при Московском физико-техническом институте
(государственном университете)

ИНФОРМАТИКА и ИКТ

Системы счисления.
Способы представления чисел

Задание №1 для10-х классов

(2010 – 2011 учебный год)

г. Долгопрудный, 2011

Составитель: В.В. Мерзляков, ассистент кафедры информатики СУНЦ МГУ.

Информатика: задание №1 для 10-х классов (2010 – 2011 учебный год). – М.: 2011, 28с.

Составитель:

Мерзляков Василий Владимирович

Подписано 04.04.10. Формат 60×90 1/16.

Федеральная заочная физико-техническая школа
при Московском физико-техническом институте
(государственном университете)

141700, Москов. обл., г. Долгопрудный, Институтский пер., 9
ФЗФТШ при МФТИ, тел/факс (495) 408-51-45 – **заочное отделение**
тел./факс (498) 744-63-51 – **очно-заочное отделение**
тел. (498) 744-65-83 – **очное отделение**

e-mail: zftsh@mail.mipt.ru

Наш сайт: www.school.mipt.ru

§ 1. Системы счисления

1.1. *Основные понятия и определения*

С древнейших времён числа сопутствовали человеку во многих сферах его жизни. Что только людям не приходилось подсчитывать. И, соответственно, нужно было каким-то образом фиксировать результаты своих подсчётов. Это можно сделать самыми разными способами (отмечать дни палочками, использовать картинки, иероглифы или числа, которыми пользуемся мы сегодня). Способ записи (обозначения) чисел — это и есть **система счисления**. Символы, с помощью которых записываются целые неотрицательные числа, называются **цифрами**. А совокупность цифр называется **алфавитом** системы счисления.

Давайте попробуем установить некие общие свойства для различных систем счисления. Прежде всего, поделим их на два класса – позиционные и непозиционные.

Определение: Система счисления называется **непозиционной**, если значение каждой цифры не зависит от её положения в записи числа.

Примером непозиционной системы счисления является простейшая палочковая система счисления, где всего одна цифра «|». Другой пример – это римская система счисления. Хотя в ней и есть некоторые правила формирования чисел (например, IX – это 9, а XI – это 11), но всё равно, где бы мы ни написали в числе цифру X, она всегда обозначает 10. К сожалению, неотрицательных чисел бесконечно много, а поскольку у каждой цифры есть своё фиксированное значение, значит, для записи произвольного числа в непозиционной системе счисления потребуется бесконечный алфавит. Поэтому в дальнейшем непозиционные системы счисления мы рассматривать не будем.

Определение: Система счисления называется **позиционной**, если количественный эквивалент каждой цифры зависит от её положения в записи числа.

Примером позиционной системы счисления является привычная нам десятичная система. Рассмотрим число 5654. Очевидно, что количественные эквиваленты цифр 5 в этом числе различны. Эквивалент первой пятёрки – это 5000, а второй – всего лишь 50.

Определение: **Базисом** позиционной системы счисления называется последовательность чисел, каждое из которых задаёт вес соответствующего разряда.

Для знакомой нам десятичной системы счисления базисом будет являться последовательность {1, 10, 100, 1000 ,…}.

Число в позиционной системе счисления формируется следующим образом: значение каждой цифры умножается на соответствующий элемент базиса и полученные значения складываются. Например, число 5654 формируется так: 4*1+5*10+6*100+5*1000. Такой способ формирования чисел называется аддитивно-мультипликативным. Данный способ позволяет записать практически в любой позиционной системе счисления произвольное число, используя конечный алфавит цифр.

Исходя из определения базиса, очевидно, что можно придумать любую последовательность чисел, объявить её базисом и записывать числа в полученной системе счисления, но будет ли это удобно? Оказывается, что далеко не всегда. Для того чтобы система счисления помогала нам жить, её базис должен соответствовать некоторым требованиям.

Определение: Позиционная система счисления называется **традиционной**, если её базис является геометрической прогрессией, знаменатель которой – натуральное число, большее единицы, а значения цифр – целые неотрицательные числа.

В дальнейшем мы будем изучать исключительно традиционные системы счисления.

Определение: Знаменатель геометрической прогрессии, члены которой образуют базис позиционной традиционной системы счисления, называется **основанием** этой системы счисления. Традиционные системы счисления с основанием P называются P-ичными (например, десятичная, двоичная, восьмеричная, шестнадцатеричная и т. д.).

Любопытным свойством традиционных систем счисления является то, что размер алфавита в таких системах равен основанию. Так, в десятичной системе счисления 10 цифр, в двоичной – 2 , в троичной – 3, в восьмеричной – 8, а в шестнадцатеричной – аж целых 16. Существуют стандартные правила формирования алфавита. Первые 10 символов в нём – это всегда арабские цифры: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9. Таким образом, с системами счисления, основание которых меньше десяти, всё ясно: их алфавит – это первые P арабских цифр. Следующие 26 символов алфавита – это заглавные латинские буквы. Таким образом, в шестнадцатеричной системе счисления алфавитом будет являться множество {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}. Если же основание системы счисления больше 36, то в качестве цифр обычно используют десятичные числа, заключённые в квадратные скобки. Например [45], [178], [239875983498759] и т. д. Однако это правило является нестрогим. Возможны и другие способы формирования алфавита для систем счисления с основанием большим 36.

1.2. *Представление чисел в традиционных системах счисления*

Прежде всего отметим, что здесь и в дальнейшем в рамках параграфа 1 мы будем рассматривать только неотрицательные числа. Зато будем говорить как о целых, так и о вещественных (в курсе математики вещественные числа назывались действительными). Далее по тексту, вещественные числа будем иногда называть дробями. Теперь начнём рассмотрение представления целых чисел.

Вспомним, как в математике определяется множество натуральных чисел: «Натуральные числа – это числа вида 1, 2, 3, 4, … ». То есть данное множество задаётся путём перечисления элементов. Метод перечисления можно использовать и для определения множества натуральных чисел в любой традиционной системе счисления. Приведём правила перечисления натуральных чисел, обозначив буквой P основание системы счисления.

Правила перечисления натуральных чисел

1) Если последняя (крайняя справа) цифра числа ***a*** меньше, чем ***P*–1**, то в следующем по порядку натуральном числе все цифры, кроме последней, будут совпадать с цифрами числа ***a**,* а последняя цифра числа ***a*+1** будет следующим элементом алфавита системы счисления, по сравнению с последней цифрой числа ***a***; (или можно сказать, что последняя цифра числа ***a*+1** будет на единицу больше последней цифры числа ***a***).
2) Если последняя (крайняя справа) цифра числа ***a*** равна ***P*–1**, то последняя цифра числа ***a*+1** будет равна **0**, и происходит перенос единички в следующий разряд. Если цифра в следующем разряде тоже равна ***P*–1**, то она становится равной **0** и снова происходит перенос единички в следующий разряд. Перенос завершается, когда очередная цифра не равна ***P*–1** или когда все разряды закончились, и был создан новый (например, переход от числа 9999 к числу 10000 в десятичной системе счисления).

Прежде чем приводить примеры перечисления натуральных чисел, условимся здесь и в дальнейшем после числа с помощью нижнего индекса указывать основание системы счисления, в которой записано число (например, число 34 в восьмеричной системе счисления будет выглядеть как 34_8).

Приведём пример перечисления первых 6 чисел в двоичной системе счисления:

$1=1_2$;
$2=10_2$;
$3=11_2$;
$4=100_2$;
$5=101_2$;
$6=110_2$.

Теперь, приведём пример перечисления чисел с 10 до 18 в шестнадцатеричной системе счисления:

$10=A_{16}$;
$11=B_{16}$;

$12=C_{16}$;
$13=D_{16}$;
$14=E_{16}$;
$15=F_{16}$;
$16=10_{16}$;
$17=11_{16}$;
$18=12_{16}$.

Итак, мы научились перечислять натуральные числа. Зачастую (например, в ЕГЭ) встречаются задачи, которые легко решаются методом выписывания числового ряда.

Задача 1. *Выписать в семеричной системе счисления все числа меньшие 20, которые кратны 5.*

Для решения этой задачи выпишем ряд натуральных чисел в семеричной системе счисления от 1 до 20 (напоминаю, что в семеричной системе счисления есть цифры: 0, 1, 2, 3, 4, 5 и 6):

1, 2, 3, 4, **5**, 6, 10, 11, 12, **13**, 14, 15, 16, 20.

Числа, кратные пяти, расположены в каждой пятой позиции этого ряда (выделены жирным шрифтом).

Таким образом, в ответе мы напишем числа: 5 и 13. Следующее семеричное число, кратное пяти – это 21, но оно не входит в диапазон, рассматриваемый в задаче.

Теперь поговорим о свёрнутой и развёрнутой формах записи чисел. Говорят, что число записано в свёрнутой форме, если оно записано в виде перечисления его цифр без пробелов от старших разрядов к младшим. То есть мы с первого класса привыкли записывать числа в свёрнутой форме. Однако существует ещё развёрнутая форма записи чисел, которую удобно использовать, например, во многих алгоритмах перевода. Развёрнутая форма записи явным образом показывает аддитивно-мультипликативное формирование числа в позиционной системе счисления, когда каждая цифра умножается на соответствующий элемент базиса и полученные значения складываются. Доказано, что у каждого числа в любой традиционной системе счисления существует и единственна развёрнутая форма.

Пример. Запишем в развёрнутой форме число 654_9 (напоминаем, что нижний индекс показывает основание системы счисления, то есть в данном случае речь идёт о девятеричной системе) $654_9 = (4*9^0 + 5*9^1 + 6*9^2)_9$.

Теперь представим в развёрнутой форме нецелое число, например $543{,}23_8$. Развёрнутая форма дроби практически не отличается от целого числа, только степени у основания будут отрицательные. Итак, $543{,}23_8 = (5*8^2 + 4*8^1 + 3*8^0 + 2*8^{-1} + 3*8^{-2})$.

Использование развёрнутой формы записи позволяет с лёгкостью решать, например такие задачи.

Задача 2. *В какой системе счисления число* 371_{10} *выглядит как* 173*?*

Для решения этой задачи необходимо записать интересующее нас число в неизвестной системе счисления в развёрнутой форме. Обозначив основание системы счисления буквой $\boldsymbol{P}$, получаем: $3*P^0 + 7*P^1 + 1*P^2$. И нам известно, что в десятичной системе счисления данное выражение равно 371. Значит, осталось решить простое квадратное уравнение: $P^2 + 7*P + 3 = 371$. Учтите, что основание системы счисления – всегда натуральное число, поэтому отрицательные, нецелые и нулевые корни надо отбраковывать. Данное уравнение имеет всего 1 натуральный корень – это число 16. Значит, число 371_{10} выглядит как 173 в шестнадцатеричной системе счисления.

1.3. *Арифметика в традиционных системах счисления*

Во всех позиционных системах счисления арифметические операции выполняются по одним и тем же правилам, согласно соответствующим таблицам сложения и умножения. Для всех систем счисления справедливы одни и те же законы арифметики: коммутативный, ассоциативный, дистрибутивный, а также правила сложения, вычитания, умножения и деления столбиком.

В *P*-ичной системе счисления таблица сложения представляет результаты сложения каждой цифры алфавита *P*-ичной системы с любой другой цифрой этой же системы. Составление подобной таблицы не составляет труда. Наиболее простыми являются таблицы сложения в двоичной и троичной системах счисления.

+	**0**	**1**
0	0	1
1	1	10_2

+	**0**	**1**	**2**
0	0	1	2
1	1	2	10_3
2	2	10_3	11_3

Пример. Сложение столбиком в двоичной, троичной и шестнадцатеричной системах счисления.

$$\begin{array}{r} 101{,}01_2 \\ +\ \ 1{,}11_2 \\ \hline 111{,}00_2 \end{array} \qquad \begin{array}{r} 21_3 \\ +\ 2{,}1_3 \\ \hline 100{,}1_3 \end{array} \qquad \begin{array}{r} F2A_{16} \\ +\ E9_{16} \\ \hline 1013_{16} \end{array}$$

Поскольку мы привыкли мыслить в десятичной системе счисления, покажем, как надо рассуждать в последнем примере (шестнадцатеричная система). Выпишите слагаемые из последнего примера и, читая комментарии, записывайте соответствующие цифры ответа. Итак. «Складываем младшие цифры A и 9. Цифра A в десятичной системе – это 10, получаем 19. Получилось больше, чем 16, значит, происходит перенос в следующий разряд. Пишем 3, а 1 в уме. Складываем цифры второго разряда 2 и E. Получаем 16 (E=14), да ещё 1 было в уме, итого 17, следовательно, 1 пишем и 1 в уме. В третьем разряде только одна цифра F (F=15) да 1 было в уме, получаем 16, значит, пишем 0 и 1 в следующем разряде. Ответ: 1013.»

Приведённые рассуждения показывают, что можно все цифры переводить в десятичную систему счисления и все действия выполнять только в привычной и знакомой десятичной системе счисления, а перенос 1 в следующий разряд выполнять, когда в результате сложения в очередном разряде получилось число, большее или равное основанию системы счисления.

ВНИМАНИЕ! При выполнении сложения в системах счисления с основанием, меньшим 10, можно случайно записать в ответ цифры, которые не входят в систему счисления. Например, при выполнении действий в семеричной системе в ответе не должны появляться цифры 7, 8, 9. Проверяйте все цифры ответа на принадлежность к рассматриваемой системе счисления!

Вычитание из большего числа меньшего в *P*-ичной системе счисления можно производить столбиком подобно вычитанию в десятичной системе. Аналогично операции сложения можно все цифры переводить в десятичную систему и выполнять вычитание. Но надо помнить, что когда мы «занимаем» единицу из старшего разряда, в младший переходит не «десять», а соответствующее основание системы счисления.

Пример. Вычитание в двоичной, троичной и шестнадцатеричной системах счисления.

$$\begin{array}{r} 101_2 \\ -\ 10{,}1_2 \\ \hline 10{,}1_2 \end{array} \qquad \begin{array}{r} 210_3 \\ -102_3 \\ \hline 101_3 \end{array} \qquad \begin{array}{r} A10_{16} \\ -102_{16} \\ \hline 90E_{16} \end{array}$$

Давайте снова посмотрим, как надо рассуждать при решении примера на вычитание в шестнадцатеричной системе счисления. «Рассматриваем младшие разряды. Из 0 вычесть 2 нельзя, значит, занимаем единицу во втором разряде, соответственно, в первый разряд переходит 16. 16 – 2 = 14, что соответствует цифре E, которую мы и пишем в ответ. Смотрим на второй разряд. После занимания единицы получилось 0 – 0, поэтому в ответ также идёт 0. Смотрим на третий, последний разряд. Цифра A в десятичной системе имеет значение 10. 10 – 1 = 9, что мы и записываем в ответ. **Ответ:** 90E.» Для интереса можете проверить все ответы из примеров на вычитание с помощью операции сложения.

Замечание. Поскольку мы не рассматриваем отрицательные числа, то соответственно мы не рассматриваем операцию вычитания из меньшего числа большего. Ни в одной задаче на вычитание не получится отрицательного ответа.

Для того чтобы выполнять операции умножения и деления, необходимо знать таблицу умножения для соответствующей системы счисления. Но это слишком трудоёмко. Гораздо проще перевести оба множителя (или делимое и делитель) в десятичную систему счисления, выполнить действие, а затем результат перевести в исходную систему счисления. Применять такой способ для сложения и вычитания невыгодно. Эти операции проще выполнять напрямую.

1.4. *Алгоритмы перевода чисел из одной системы счисления в другую*

Для перевода чисел из одной системы счисления в другую необходимо будет пользоваться арифметикой одной из этих систем счисления. Причём придётся выполнять много раз операции умножения и деления. Поскольку эти операции неудобно выполнять в какой-либо системе счисления, кроме десятичной, то из соображений наглядности и удобства восприятия мы будем все алгоритмы перевода формулировать именно для десятичной системы счисления. То есть, мы научимся переводить числа из P-ичной системы в десятичную, и обратно. Если же нам потребуется перевести, например, семеричное число в тринадцатеричное, то мы сначала переведём его из семеричной системы в десятичную, а потом из десятичной в тринадцатеричную.

При переводе нужно помнить, что целое число всегда превращается в целое, а рациональное в рациональное, то есть конечная дробь не может превратиться в бесконечную непериодическую.

Начнём изучение алгоритмов перевода с алгоритма перевода целых чисел из P-ичной системы счисления в десятичную. Для решения этой задачи представим исходное число в развёрнутой форме $a = a_nP^n + a_{n-1}P^{n-1} + \ldots + a_1P + a_0$. Для того чтобы получить значение этого многочлена, записанное в десятичной системе счисления, следует число P и коэффициенты при степенях P (цифры P-ичного числа) записать в виде десятичных чисел и все вычисления провести в десятичной системе. Данный способ можно сформулировать в виде следующего алгоритма.

Алгоритм перевода целых чисел из *P*-ичной системы счисления в десятичну:

1) *каждая цифра P-ичного числа переводится в десятичную систему*;

2) *полученные числа нумеруются справа налево, начиная с нуля*;

3) *десятичное число, соответствующее каждой P-ичной цифре, умножается на P^k, где k — номер этого числа* (*п.* 2), *и результаты складываются*, *причём все арифметические действия проводятся в десятичной системе*.

Пример. Переведём число $B0F9_{16}$ в десятичную систему.

$$B0F9_{16} = [11_{10}][0][15_{10}][9] = 11_{10}\cdot(16_{10})^3 + 0\cdot(16_{10})^2 + 15_{10}\cdot(16_{10})^1 + $$
$$+9\cdot(16_{10})^0 = 45305_{10}.$$

Цифра B была заменена на десятичное число 11, а цифра F – на 15.

Теперь рассмотрим алгоритмы перевода конечной P-ичной дроби в десятичную дробь. Их существует два.

Алгоритм № 1

Представим дробь в развёрнутой форме

$$b = b_{-1} P^{-1} + b_{-2} P^{-2} + \ldots + b_k P^{-k} .$$

Для того чтобы вычислить значение многочлена в десятичной системе счисления, следует число P и коэффициенты многочлена (цифры P-ичного числа) записать в виде десятичных чисел и все вычисления проводить в десятичной системе. Запишем эти правила в виде алгоритма.

Алгоритм перевода конечной *P*-ичной дроби в десятичную:

1) *целая часть числа переводится в десятичную систему отдельно;*

2) *каждая цифра дробной части P-ичного числа переводится в десятичную систему;*

3) *полученные числа (в дробной части) нумеруются слева направо, начиная с единицы*;

4) *десятичное число, соответствующее каждой P-ичной цифре, умножается на P^{-k}, где k – номер этого числа, результаты складываются, причём все арифметические действия проводятся в десятичной системе.*

Пример. Переведём число $0,B0F9_{16}$ в десятичную систему счисления.

$$0,B0F9_{16} = 0,[11_{10}][0][15_{10}][9] = 11\cdot16^{-1} + 15\cdot16^{-3} + 9\cdot16^{-4} =$$
$$= 0,69129943847656625_{10}$$

Здесь, согласно пункту (3) алгоритма, числа были пронумерованы так: 11 — номер один (коэффициент при P в степени «минус один»), 15 — номер три, 9 — номер четыре.

Алгоритм № 2

Представим конечную P-ичную дробь в виде обыкновенной дроби. Числителем этой дроби будет число, стоящее после запятой, а знаменателем — P^n, где n есть количество значащих цифр в дробной части. Далее числитель запишем в десятичной системе (знаменатель по определению записан в десятичной системе), и мы получим обыкновенную дробь в десятичной системе счисления. При необходимости её можно записать в виде десятичной дроби (конечной или периодической, выделяя непериодическую часть и период).

Пример. Переведем число $0{,}13_{15}$ в десятичную систему счисления

$$0{,}13_{15} = \frac{13_{15}}{{15_{10}}^2} = \frac{18_{10}}{225_{10}} = \frac{2_{10}}{25_{10}} = 0{,}08_{10}.$$

Второй алгоритм наиболее эффективен в случаях, когда среди простых делителей основания системы счисления P содержатся какие-нибудь числа, кроме 2 и 5, и соответствующую конечную P-ичную дробь невозможно представить в виде конечной десятичной.

Пример. Переведем число $0{,}1A_{15}$ в десятичную систему счисления $0{,}1A_{15} = \dfrac{1A_{15}}{{15_{10}}^2} = \dfrac{25_{10}}{225_{10}} = \dfrac{1_{10}}{9_{10}} = 0{,}(1)_{10}.$

Теперь перейдём к рассмотрению обратных алгоритмов. То есть будем переводить числа из десятичной системы счисления в P-ичную. Начнём с целых чисел. Запишем исходное число a в P-ичной системы счисления в развёрнутой форме

$$a=a_n\cdot P^n + a_{n-1}\cdot P^{n-1} + \ldots + a_1\cdot P + a_0,$$

где a_n, a_{n-1}, …, a_1, a_0 пока неизвестны. Тогда, разделив a на P с остатком, получаем целое частное

$$a_n\cdot P^{n-1} + a_{n-1}\cdot P^{n-2} + \ldots + a_1$$

и a_0 в остатке, который не превышает P–1. Таким образом, мы получили последнюю цифру в P-ичной записи числа a. Разделим полученное частное вновь на P, получив в остатке значение a_1 и новое частное: $a_n \cdot P^{n-2} + a_{n-1} \cdot P^{n-3} + \ldots + a_2$. Таким образом, мы получили уже предпоследнюю цифру в P-ичной записи числа a. Продолжаем этот процесс, пока результат целочисленного деления не станет равен нулю. Более формально данный способ можно записать так.

Алгоритм перевода целого числа из десятичной системы счисления в *P*-ичную:

1) *делим исходное число а на Р нацело в десятичной системе и записываем в качестве нового значения десятичного числа а целую часть результата от деления*;
2) *остаток от деления заменяем на соответствующую цифру в Р-ичной системе счисления и приписываем её слева к полученным ранее цифрам Р-ичной записи числа а* (*первая полученная цифра соответствует младшему разряду и её мы просто записываем*);
3) *выполняем пункты* 1 *и* 2 *до тех пор, пока число а не станет равным* 0.

Пример. Переведем число 123 в троичную систему счисления.

```
123:3=41   (0)
 41:3=13   (2)
 13:3= 4   (1)
  4:3= 1   (1)
  1:3= 0   (1)
```

В скобках указаны остатки от целочисленного деления, которые являются соответствующими цифрами в троичном представлении числа.

Ответ: $123 = 11120_3$.

Переведем это же число в шестнадцатеричную систему счисления.

```
123:16=7(11)
7:16=0(7)
```

Заменим число 11 на шестнадцатеричную цифру B.

Ответ: $123=7B_{16}$.

Осталось рассмотреть лишь алгоритм перевода конечных десятичных дробей в P-ичную систему счисления. Описанный ниже алгоритм применяется только к дробной части числа. Если число имеет целую часть, то она переводится в *P*-ичную систему счисления отдельно по алгоритму, описанному выше.

Сформулируем правила перевода десятичных дробей в *P*-ичную систему в виде алгоритма.

Алгоритм перевода правильной конечной десятичной дроби в *P*-ичную систему счисления:

1) *умножим исходное число на P* (*основание новой системы счисления*)*, целая часть полученного произведения является первой цифрой после запятой в искомом числе* (*целая часть может быть равна нулю, но она всегда меньше чем P, это позволяет записать её в виде ровно одной цифры P-ичной системы счисления*);
2) *дробную часть произведения снова умножим на P, целую часть полученного числа заменим на цифру в P-ичной системе и припишем её справа к результату*;
3) *выполняем пункт* 2 *до тех пор, пока дробная часть произведения не станет равной нулю или не выделится период* (*дробная часть окажется равной уже получавшейся ранее дробной части произведения*).

Пример. Переведём число 0,375 в двоичную систему счисления.

0,375·2=0,75	**0** – первая цифра результата
0,75 ·2=1,5	**1** – вторая цифра результата
0,5 ·2=1,0	**1** – последняя цифра результата

Ответ: $0{,}375 = 0{,}011_2$.

Переведём число 0,515625 в четверичную систему счисления.

0,515625·4=2,0625	**2**
0,0625 ·4=0,25	**0**
0,25 ·4=1,0	**1**

Ответ: $0{,}53125 = 0{,}201_4$.

Переведём число 0,109375 в шестнадцатеричную систему.

0,109375·16=1,75	**1**
0,75· 16=12,0	$12_{10} = \boldsymbol{C}_{16}$

Ответ: 0,109375=0,1C_{16}.

Переведём в пятеричную систему счисления число 0,123.

0,123·5=0,615	**0**
0,615·5=3,075	**3**
0,075·5=0,375	**0**
0,375·5=1,875	**1**
0,875·5=4,375	**4**

Дробная часть последнего произведения равна уже встречавшейся ранее дробной части, следовательно, последние две цифры образуют период пятеричной дроби.

Ответ: 0,123 = $0,030(14)_5$.

Мы рассмотрели универсальные алгоритмы перевода, которые можно применять к любым традиционным системам счисления. Однако есть частный случай, когда перевод можно осуществить более простым способом. Это случай, когда основания систем счисления P и Q связаны следующим соотношением: $P^m = Q$, где m – натуральное число.

Тогда для того, чтобы перевести целое число из системы счисления с основанием P в систему счисления с основанием $Q = P^m$, достаточно запись числа в P-ичной системе разбить на группы по m цифр, начиная с **правой** цифры, и каждую группу из m цифр заменить одной цифрой в Q-ичной системе.

Например, 10101_2=10|101=25_8 (P=2; Q=8; m=3).

Если в последней группе получилось меньше m цифр, то можно дополнить число слева ведущими нулями.

Для того чтобы перевести целое число из системы счисления с основанием $Q = P^m$ в систему счисления с основанием P необходимо каждую Q-ичную цифру перевести в систему с основанием P и дополнить, если это необходимо, полученные числа слева нулями так, чтобы каждое число, за исключением левого, состояло ровно из m цифр. Например, 73_{16} = 111|0011=1110011_2 (P = 2; Q = 16; m = 4).

Перевод дробной части из Q-ичной системы в P-ичную осуществляется, как и для целых чисел. Незначащими в дробной части теперь являются правые нули в P-ичном представлении самой правой цифры дробной части Q-ичного числа. При обратном же переводе цифры P-ичной дроби группируются по m штук слева направо, начиная с первой цифры после запятой, если последняя группа содержит менее m цифр, то к ней добавляют соответствующее количество нулей.

Пример. Переведём число $A{,}1C_{16}$ из шестнадцатеричной системы в четверичную.

Для шестнадцатеричной и четверичной систем счисления выполняется соотношение $Q = P^m$, т. к. $16 = 4^2$. Поэтому заменим каждую 16-ричную цифру ее 4-ричным представлением, для чего используем десятичную систему в качестве промежуточной: $A_{16} = 10_{10+} = 22_4$; $C_{16} = 12_{10} = 30_4$. $A{,}1C_{16} = 22{,}|01|30_4$ (последний незначащий 0 можно опустить).

Ответ: $A{,}1C_{16} = 22{,}013_4$.

Пример. Переведём двоичное число $1010{,}00011011_2$ в восьмеричную систему.

Для двоичной и восьмеричной систем счисления выполняется соотношение $Q = P^m$, т. к. $2^3 = 8$. Следовательно, при переводе будем группировать по три цифры двоичного числа (в целой части – справа налево, в дробной части – слева направо): $1|010{,}|000|110|11_2 = 12{,}066_8$ (последняя группа двоичных цифр была дополнена нулем справа).

Ответ: $1010{,}00011011_2 = 12{,}066_8$.

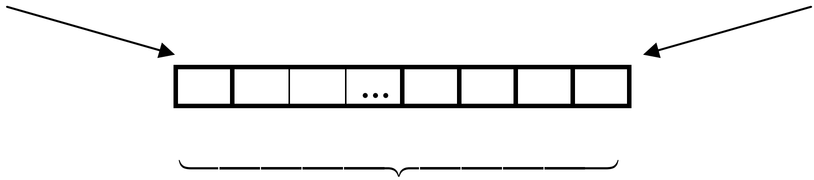
Замечание. Перед использованием данных алгоритмов бывает удобно выписать кодировочную таблицу, где каждой Q-ичной цифре ставится в соответствие m P-ичных цифр.

§ 2 Представление чисел в компьютере. Числа в языке программирования

Со второй половины XX века, века компьютеризации, человечество ежедневно пользуется двоичной системой счисления, так как вся информация, обрабатываемая современными компьютерами, хранится в них в двоичном виде.

Каждая ячейка памяти представляет собой физическую систему, состоящую из некоторого числа однородных элементов, обладающих двумя устойчивыми состояниями, одно из которых соответствует нулю, а другое — единице. Каждый такой элемент служит для изображения одного из разрядов двоичного числа. Именно поэтому каждый элемент ячейки называют *разрядом*.

(k – 1)-й разряд 0-й разряд



ячейка из k разрядов

Для оценки объёма памяти компьютера используются такие единицы измерения, как бит и байт. Битом обозначается каждый элемент ячейки, поэтому говорят, что 1 бит может хранить только ноль или единицу. Байтом называется ячейка, состоящая из 8 битов, то есть можно сказать, что байт – это восьмиразрядная ячейка.

Для компьютерного представления целых чисел обычно используется несколько различных способов представления, отличающихся друг от друга количеством разрядов и наличием или отсутствием знакового разряда. *Беззнаковое* представление можно использовать только для неотрицательных целых чисел, отрицательные числа можно представлять только в *знаковом* виде.

При беззнаковом представлении все разряды ячейки отводятся под само число. При представлении со знаком самый старший (левый) разряд отводится под знак числа, остальные разряды — под собственно число. Если число положительное, то в знаковый разряд помещается 0, если число отрицательное, то 1. Очевидно, в ячейках одного и того же размера без знака можно представить больший диапазон целых чисел, чем со знаком. Например, в одном байте (8 разрядов) можно записать положительные числа от 0 до 255, а со знаком — только до 127, поскольку под само число отводится всего 7 разрядов. Зато при знаковом представлении в том же самом байте ещё можно записать отрицательные числа до –128. Таким образом, в одном байте можно в любом случае представить 256 целых чисел (от 0 до 255 в беззнаковом представлении и от –128 до 127 в знаковом представлении).

Для получения компьютерного представления беззнакового целого числа в k-разрядной ячейке памяти достаточно перевести его в двоичную систему счисления и дополнить полученный результат слева нулями до k разрядов. Понятно, что существует ограничение на значения чисел, которые мы можем записать в k-разрядную ячейку.

Максимальное значение целого неотрицательного числа достигается в случае, когда во всех разрядах ячейки хранятся единицы. Для k-разрядного представления оно будет равно $2^k - 1$. Минимальное число соответствует k нулям, хранящимся в k разрядах ячейки, и оно всегда равно нулю. Ниже приведены максимальные значения для беззнаковых целых k-разрядных чисел:

Количество разрядов	Максимальное значение
8	255 ($2^8 - 1$)
16	65535 ($2^{16} - 1$)
32	4294967295 ($2^{32} - 1$)
64	18446744073709551615 ($2^{64} - 1$)

Для получения компьютерного представления знакового целого числа в k-разрядной ячейке памяти достаточно перевести его в двоичную систему счисления, дополнить полученный результат слева нулями до k –1 разрядов и записать в левый разряд 0, если число положительное, и 1, если оно отрицательное. Таким образом, получается прямой код отрицательного числа. Сейчас мы не будем рассматривать представление отрицательных чисел в виде дополнительного кода. (Подробно об этом можно прочитать, например, в книге «Математические основы информатики» авторов Андреевой Е.В., Босовой Л.Л., Фалиной И.Н.)

В k-разрядной ячейке памяти можно представлять целые числа со знаком из диапазона $[-2^{k-1}, 2^{k-1} - 1]$. Так, в восьмиразрядной ячейке можно представить числа из диапазона [–128,127] (при этом число минус 128 в прямом коде представляется в виде единицы и 7 нулей), в шестнадцатиразрядной ячейке – из диапазона [–32768, 32767] и т. д.

Для представления вещественных чисел, отводимые под них разряды делятся на 2 группы. Первая из них называется мантиссой, и в данных разрядах хранятся значащие цифры вещественного числа. Вторая группа разрядов называется порядком и содержит смещение мантиссы относительно десятичной запятой.

Теперь перейдём к использованию чисел в программировании. Для удобства и простоты мы будем рассматривать язык программирования, который лучше всего подходит для обучения – это язык **Паскаль (Pascal).** Сейчас нас интересует часть языка, связанная с числами. При программировании, естественно, используются как целые, так и вещественные числа. И под каждое из них в оперативной памяти отводится соответствующее количество байт. Сколько именно байт отвести под число, зависит от его типа. В языке Паскаль существует несколько типов целых чисел и несколько типов вещественных.

Нас будут интересовать два целых типа, которые называются **integer** и **longint**. Оба этих типа являются знаковыми (то есть числа представляются с использованием знакового бита). В типе integer под каждое число отводится 2 байта или 16 разрядов соответственно, в этом типе можно работать с числами из диапазона [–32768, 32767]. В типе longint под каждое число отводится 4 байта или 32 разряда, соответственно, в этом типе можно работать с числами из диапазона [–2147483648, 2147483647]. Также мы будем работать с одним вещественным типом, который называется **real** (напомним, что вещественные числа также называются действительными). Вещественные числа в Паскале записываются с использованием десятичной точки, например 4.34.

При работе с числами нам обязательно придётся выполнять над ними арифметические операции. В языке программирования Паскаль существует шесть операций: сложение (обозначается знаком «+»), вычитание (обозначается знаком «–»), умножение (обозначается знаком «*»), деление (обозначается знаком «/»), деление нацело (обозначается словом «div») и взятие остатка от деления нацело (обозначается словом «mod»).

Теперь у нас есть числа, есть операции, осталось научиться составлять из всего этого арифметические выражения. Важным понятием в арифметике является понятие операнда. Операндами называются те объекты, над которыми выполняется арифметическая операция. В математике различные операции могут иметь разное количество операндов, но все арифметические имеют два операнда. Операндом для операции может являться как одиночное число, так и целое арифметическое выражение. Рассмотрим выражение (2+2)*2. У операции сложения операндами являются два числа 2, а у операции умножения правый операнд – это число 2, а левый – это целое выражение в скобках (2+2). Прежде чем выполнять операцию, необходимо вычислить оба её операнда.

Приоритет операций в Паскале точно такой же, как и в математике. Сначала выполняются операции умножения, деления, div и mod (это тоже операции деления), а потом операции сложения и вычитания. Операции одного приоритета выполняются слева направо. Для изменения порядка действий можно использовать круглые скобки. Операции в скобках имеют более высокий приоритет, чем операции вне скобок. Так, при вычислении выражения 2+2*2 получается число 6, потому что операция умножения имеет более высокий приоритет, чем сложение, и, следовательно, выполняется первой. Если же записать выражение (2+2)*2, то при вычислении получается число 8, потому что сложение в скобках выполняется раньше умножения.

Теперь возникает следующий вопрос. Как определить тип результата при вычислении арифметического выражения? Для ответа на этот вопрос необходимо знать особенности выполнения всех арифметических операций. Итак, операции сложения, вычитания и умножения выдают целый результат, если оба их операнда целые, и вещественный, если хотя бы один из операндов – вещественный.

Операция деления «/» **всегда** выдаёт вещественный результат. Даже если мы 4 делили на 2, всё равно в итоге получается нецелое число. На первый взгляд это кажется странным, но в отличие от математики в программировании каждое число, кроме значения, ещё имеет тип, и если типы у чисел не совпадают, то они НЕ считаются равными. Нужно уяснить, что **1 ≠ 1.0.** Операции div и mod всегда выдают целый результат и, в отличие от всех остальных арифметических операций, могут иметь только целые операнды. Попытка применить данные операции к вещественным числам приводит к критическим ошибкам…

Давайте подробнее познакомимся с двумя последними операциями. Операция *a* div *b* выдаёт целую часть от деления числа *a* на число *b.* То есть 5 div 2 = 2, а 3 div 7 = 0. Операция *a* mod *b* выдаёт остаток от деления *a* на *b* по следующему закону

```
a mod b = a – ((a div b) * b).
```

Приведём примеры выполнения этих их операций для всех возможных знаков операндов:

```
 5 div  3 =  1;  5 mod  3 =  2;
-5 div  3 = -1; -5 mod  3 = -2;
 5 div -3 = -1;  5 mod -3 =  2;
-5 div -3 =  1; -5 mod -3 = -2.
```

Следующее очень важное понятие в программировании – это понятие переменной. Переменной называется объект, который может изменять своё значение в ходе вычислений. Под каждую переменную отводится именованная область в оперативной памяти. Для каждой переменной всегда определены три вещи:

1) Уникальное имя. 2) Тип.

3) Значение в текущий момент времени.

Именем переменной может быть любая последовательность латинских букв и цифр, начинающаяся с буквы, например «D2», «VV», «ABRAcADaBRA» являются правильными именами, а «d-f», «4d», «gggggg+()*OIIJ_K» – это примеры неправильных имён переменных. Учтите, что в Паскале не различаются большие и малые латинские буквы (то есть имена ABc и abC обозначают одну и ту же переменную). Ещё одна важная особенность – символ подчёркивания («_») считается латинской буквой, поэтому имя переменной может начинаться и с данного символа.

Переменная может принадлежать к одному из известных нам типов (integer, real, longint) и может принимать значение только в соответствии со своим типом. То есть переменная типа integer не может иметь значения 40000, так как оно выходит за допустимый диапазон, или 0.5, так как это число вообще не является целым.

Значение переменных можно изменять при помощи оператора присваивания. Выглядит он следующим образом: сначала записывается имя переменной, потом знак присваивания (в Паскале знаком присваивания является «:=»), а затем значение, которое мы хотим присвоить данной переменной. Например, запись X:=5 означает, что в переменную X мы записали число 5, и если в дальнейших вычислениях мы используем переменную X, то вместо неё будет подставляться число 5. Например, если написать Y:=X–1, то в переменную Y запишется число 4, поскольку X было равно 5. Использование переменных позволяет записывать в программе математические и физические формулы один раз в общем виде, а затем, задавая конкретные начальные значения, производить бесконечное количество вычислений, не меняя текста программы!

Переменные наряду с числами могут быть операндами в арифметическом выражении. Также операндами могут быть стандартные математические функции, которые приведены в таблице на следующей странице:

Функция	**Комментарий**	**Тип аргумента**	**Тип результата**
`abs(x)`	$\lvert x\rvert$ — модуль x	`integer`, `real`	совпадает с типом аргумента
`sqr(x)`	x^2	`integer`, `real`	совпадает с типом аргумента
`sqrt(x)`	$\sqrt{x}$ — корень квадратный из x	`integer`, `real`	`Real`
`Pi`	`3.1415926535897932385`	нет	`real`
`sin(x)`	$\sin x$	`integer`, `real`	`real`
`cos(x)`	$\cos x$	`integer`, `real`	`real`
`arctan(x)`	$\operatorname{arctg} x$	`integer`, `real`	`real`
`trunc(x)`	отсекание дробной части x	`real`	`integer`
`round(x)`	округление x до ближайшего целого. Половины округляются в сторону увеличения модуля.	`real`	`integer`

Необходимо отметить, что функциям sin и cos угол следует подавать в радианах, а не в градусах! Также функция arctan возвращает результат в радианах.

При использовании операторов присваивания необходимо соблюдать правило совместимости типов, то есть тип выражения должен строго соответствовать типу переменной, в которую мы хотим присвоить значение этого выражения. Есть исключения из этого правила: в вещественную переменную можно присвоить целое значение; в переменную типа longint можно присвоить значение типа integer; в переменную типа integer можно присвоить значение типа longint, если оно укладывается в соответствующий диапазон.

Сейчас мы рассмотрим несколько примеров задач на программирование. Условимся, что если в решении придётся писать более одного оператора присваивания, то они отделяются друг от друга точкой с запятой.

Задача 1. Пусть A и B – это переменные, значениями которых являются длины сторон прямоугольника. Требуется в переменную S записать величину площади этого прямоугольника, а в переменную P – его периметр.

Решение. Эта задача решается путём записи двух операторов присваивания с соответствующими формулами.

```
S: = A*B;
P: = 2*A + 2*B.
```

Задача 2. Пусть R – радиус окружности. Требуется в переменную L записать длину этой окружности, а в переменную S площадь круга, ограниченного данной окружностью.

Решение.

```
L: = 2*pi*R;
S: = pi*sqr(R).
```

Задача 3. Пусть A и B – длины сторон треугольника, а X – величина угла между ними в радианах. Требуется в переменную C записать величину третьей стороны.

Решение. Используем теорему косинусов:

```
C: = sqrt(sqr(A) + sqr(B) –2*A*B*cos(X)).
```

Задача 4. Пусть в переменной K записано число от 100 до 999. Требуется в переменную X записать сумму его цифр.

Решение. Эта задача решается с помощью операций div и mod. Нужно выделить цифры из исходного числа и сложить их. В предложенном решении цифры выделяются от младшей к старшей.

X:=K mod 10+K div 10 mod 10+K div 100.

Действительно, взяв остаток от деления на 10, мы получаем младшую цифру в числе, а разделив нацело на 100 – старшую цифру. Для того чтобы узнать вторую цифру (количество десятков), мы выполняем две операции: сначала делим нацело на 10, отбрасывая последнюю цифру, а потом у результата (двузначного числа) берём остаток от деления на 10 и получаем ровно то, чего добивались.

Задача 5. Пусть переменные A и B имеют некоторые значения. Требуется поменять их местами.

Р**ешение**. Это очень важная задача на один из фундаментальных алгоритмов в программировании. Очевидно, что просто написать A:=B, а затем B:=A нельзя, поскольку после первого оператора присваивания переменные A и B будут иметь одинаковое значение. Необходимо сначала сохранить значение одной из переменных в какой-нибудь дополнительной. Например, в переменной C. После этого переменную, значение которой мы сохранили, можно «испортить», присвоив ей значение второй. И, наконец, во вторую переменную мы присваиваем значение первой, которое сохранено в дополнительной переменной C. Решение может выглядеть, например, так:

C: = A;
A: = B;
B: = C.

Контрольные вопросы

1(1). Какую систему счисления называют позиционной?

2(1). Дайте определение традиционной системы счисления. Что называют основанием традиционной системы счисления?

3(1). Сколько цифр нужно для записи чисел в двенадцатеричной системе счисления?

4(1). Верно ли записаны числа в семеричной системе счисления: 2360_7, 35721_7, 608512_7?

5(1). Запишите в развёрнутом виде числа 65_7; $10203{,}405_7$; $0{,}15A_{16}$; $1AF1H{,}A9_{20}$.

6(1). В какой системе счисления число 50_{10} выглядит как 35?

7(1). В каких системах счисления $5_P + 5_P \neq 10_p$?

8(1). Подсчитайте количество троичных чисел в диапазоне от 12_3 до 1000_3.

9(1). Как будет выглядеть в двоичной системе счисления десятичное число 0,125?

10(2). Какие из приведённых ниже наборов символов могут быть именами переменных в Паскале, а какие нет и почему: *a*, *a*1, 2*d*, 123, *KTO-TO*, *CCCP*, sin(*x*), *aaaaaaaa*1, sin, _1, *a*_?

11(3). Вычислите результаты следующих операций или сообщите о невозможности вычислений:
20 div 6; 20 mod 6; 20 div 4; 20 mod 4; (–20) div 6; (–20) mod 6; 2 div 5; 2 mod 5; 12 div 0.
abs(–2); abs(–2.0); sqr(–5); sqrt(16); sqrt(16.0); sin(0); sin(0.0); trunc(6.9); round(6.9); trunc(6.2); round(6.2); trunc(–1.8); round(–1.8); trunc(2); round(–2); trunc(0.5); round(0.5); trunc(–0.5); round(–0.5).

12(3). Уберите из выражения лишние скобки (то есть те, удаление которых не изменит порядок выполнения операций):

a) ((a*b) div c) mod (a + b);

b) (sin(x + y))/2 – cos((x + y)/2);

c) (a*x)/(b*y) + (a/x)/(b/y).

13(2). Сколько операций выполняется при вычислении выражения (x + 1/2)*(y + 7/10) –3/4? Как сократить число операций?

14(4). Если *y* вещественная переменная, а *n* – целая, то какие из следующих операторов присваивания правильные, а какие нет и почему?

a) y: = n + 1; b) n: = y –1; c) n: = 4.0; d) y: = trunc(y);
e) n: = n div 2; f) y: = y div 2; g) n: = n/2; h) n := sqr(sqrt(n));

15(4). Правильны ли следующие операторы присваивания? Ответ обосновать.

a) *k*: = *k* mod 3 + *k**cos(0); b) x: = x*2 div 6+x/4.

Задачи

1(3). Выполните действия сложения и вычитания над следующими

ми

парами чисел: 10010000_2 и 1110_2; 4322_5 и 3_5; $AB06_{12}$ и A_{12}.

2(3). Переведите следующие числа в десятичную систему счисления: 123_4; $123{,}4_5$; $203{,}5_6$.

3(3). Переведите десятичное число 52 в двоичную, восьмеричную и 11-ричную системы счисления.

4(4). Переведите десятичное число 6174,45 в двоичную систему счисления.

5(4). Переведите число $1234{,}5678_9$ в 27-ричную систему счисления, а число $ABCD{,}EF_{16}$ — в восьмеричную.

6(2). Чему равна сумма чисел 26_8 и 58_{16}? Ответ представьте в двоичной системе счисления.

7(2). Даны числа $a=3A_{16}$ и $b=76_8$. Выпишите все двоичные числа, лежащие на отрезке [a,b].

8(2). Выпишите все десятичные числа, не превосходящие 40, запись которых в пятеричной системе счисления оканчивается цифрой 4.

9(2). Протоссы из вселенной Starcraft используют для представления чисел ячейки из 35 разрядов, в каждой из которых первый бит обозначает, является ли число целым, а второй — показывает знак числа. Какой диапазон целых чисел используют Протоссы?

10(2). Пусть A и B – это длины катетов прямоугольного треугольника. Требуется в переменную P записать его периметр, а в переменную S – его площадь.

11(3). Пусть даны координаты трёх вершин треугольника (придумайте сами им имена). Найдите его площадь и периметр.

12(3). Пусть в переменной C записано пятизначное число. Найти произведение его цифр и записать его в переменную X.

13(2). Выразите функцию round и операцию div через функцию trunc./.