№ 1 1–15 января 2011 *Основана в 1995 г* inf.1september.ru

УЧЕБНО-МЕТОДИЧЕСКАЯ ГАЗЕТА ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

# ИНФОРМАТИКА

/ тема номера:

## Компьютерная арифметика

1

издательский дом Первое сентября

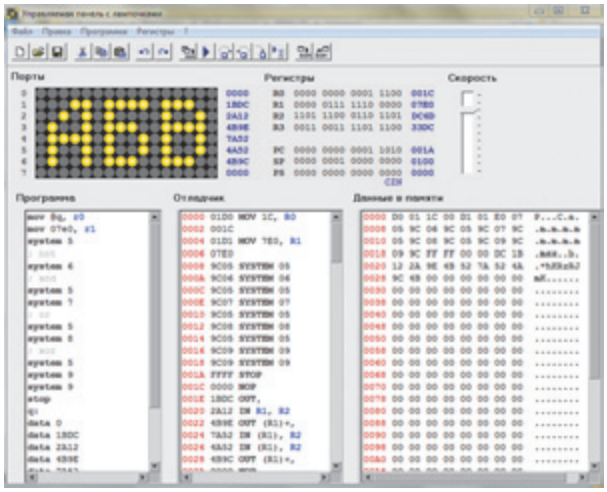


Индексы подписки | Почта России – 79006 (инд.); – 79574 (орг.) | Роспечать – 82291 (инд.); – 32591 (орг.)

# В номере



# На диске



К теме номера — авторская программа “Лампанель” с демонстрационными примерами.

К заметке про алгоритм LZW — реализация алгоритма и демонстрационная презентация.


ИНФОРМАТИКА

Учебно-методическая газета для учителей информатики
*Основана в 1995 г.*
Выходит два раза в месяц

РЕДАКЦИЯ:
гл. редактор С.Л. ОСТРОВСКИЙ
редакторы
Е.В. АНДРЕЕВА,
Д.М. ЗЛАТОПОЛЬСКИЙ
(редактор вкладки “В мир информатики”)
верстка Н.И. ПРОНСКАЯ
корректор Е.Л. ВОЛОДИНА
секретарь Н.П. МЕДВЕДЕВА
Фото: фотобанк Shutterstock
Газета распространяется по подписке
Цена свободная
Тираж 3000 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ “ПЕРВОЕ СЕНТЯБРЯ”
**Главный редактор:**
Артем Соловейчик
(Генеральный директор)
**Коммерческая деятельность:**
Константин Шмарковский
(Финансовый директор)
**Развитие, IT и координация проектов:**
Сергей Островский
(Исполнительный директор)
**Реклама и продвижение:**
Марк Сартан
**Мультимедиа, конференции и техническое обеспечение:**
Павел Кузнецов
**Производство:**
Станислав Савельев
**Административно-хозяйственное обеспечение:**
Андрей Ушков
**Дизайн:**
Иван Лукьянов, Андрей Балдин
**Педагогический университет:**
Валерия Арсланьян (ректор)

ГАЗЕТЫ ИЗДАТЕЛЬСКОГО ДОМА:
**Первое сентября** – Е.Бирюкова
**Английский язык** – А.Громушкина
**Библиотека в школе** – О.Громова
**Биология** – Н.Иванова
**География** – О.Коротова
**Дошкольное образование** – М.Аромштам
**Здоровье детей** – Н.Сёмина
**Информатика** – С.Островский
**Искусство** – М.Сартан
**История** – А.Савельев
**Классное руководство и воспитание школьников** – О.Леонтьева
**Литература** – С.Волков
**Математика** – Л.Рослова
**Начальная школа** – М.Соловейчик
**Немецкий язык** – М.Бузоева
**Русский язык** – Л.Гончар
**Спорт в школе** – О.Леонтьева
**Управление школой** – Я.Сартан
**Физика** – Н.Козлова
**Французский язык** – Г.Чесновицкая
**Химия** – О.Блохина
**Школьный психолог** – И.Вачков

УЧРЕДИТЕЛЬ:
ООО “ЧИСТЫЕ ПРУДЫ”

**Зарегистрировано ПИ № 77-72230 от 12.04.2001**
в Министерстве РФ по делам печати
Подписано в печать:
по графику 02.12.2010,
фактически 02.12.2010
Заказ №
Отпечатано в ОАО “Чеховский полиграфический комбинат”
ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300

АДРЕС ИЗДАТЕЛЯ:
ул. Киевская, д. 24,
Москва, 121165
**Тел./факс:** (499) 249-31-38

**Отдел рекламы:**
(499) 249-98-70
**http://1september.ru**

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:
**Телефон:** (499) 249-47-58
**E-mail: podpiska@1september.ru**

Документооборот Издательского дома “Первое сентября” защищен антивирусной программой Dr.Web

ПОДПИСНЫЕ ИНДЕКСЫ: по каталогу «Газеты. Журналы» агентства «Роспечать» – 32291 (инд); – 32591 (орг.); по каталогу «Почта России» – 79006 (инд); – 79574 (орг.)

# Компьютерная арифметика

**К.Ю. Поляков,**
**А.П. Шестаков,**
**Е.А. Еремин**

*Уважаемые коллеги! Мы представляем вашему вниманию очередную главу нового учебника "Информатика и ИКТ" для профильного уровня, подготовка которого будет завершена в ближайшее время (другие материалы учебника опубликованы в № 12 и 16 газеты "Информатика" за 2010 год, а также на сайте http://kpolyakov.narod.ru/school/probook.htm). Эта глава посвящена компьютерной арифметике — очень важной теме, в изложении которой допускается много неточностей. Мы благодарны Н.Д. Шумилиной за ценные замечания по поводу этого материала.*

## Особенности представления чисел в компьютере

Решая задачи на уроках математики, вы никогда не обсуждали, как хранятся числа. Математика — это теоретическая наука, для которой совершенно неважно, записаны они на маленьком или большом листе бумаги, зафиксированы с помощью счетных палочек, счетов или, используя новейшие технологии, внутри полупроводниковой схемы. Поэтому число в математике может состоять из любого количества цифр, которое требуется в решаемой задаче.

В то же время инженеры, разрабатывающие компьютер, должны спроектировать реальное устройство из вполне определенного количества деталей. Поэтому число разрядов, отведенных для хранения каждого числа, ограниченно, и точность вычислений тоже ограничена. Из-за этого при компьютерных расчетах могут возникать достаточно серьезные проблемы. Например, сумма двух положительных чисел может получиться отрицательной, а выражение **A + B** может совпадать с **A** при ненулевом **B**. Мы рассмотрим важные особенности компьютерной арифметики, которые нужно учитывать при обработке данных. В первую очередь они связаны с тем, как размещаются целые и вещественные числа в памяти компьютера.

## Предельные значения чисел

Как вы уже поняли, числа, хранящиеся в компьютере, не могут быть сколь угодно большими и имеют некоторые предельные значения. Представим себе некоторое вычислительное устройство, которое работает с четырехразрядными неотрицательными целыми десятичными числами. Для вывода чисел используется четырехразрядный индикатор, на котором можно отобразить числа от 0 (все разряды числа минимальны) до 9999 (все разряды максимальны):


Вывести на такой индикатор число 10 000 невозможно: не хватает технического устройства для пятого разряда. Такая "аварийная" ситуация называется *переполнением разрядной сетки*, или просто *переполнением* (англ. *overflow* — переполнение "сверху").

В нашем примере переполнение возникает при значениях, больших $9999 = 10^4 - 1$, где 4 — это количество разрядов. В общем случае, если в системе счисления с основанием $B$ для записи числа используется $K$ разрядов, максимально допустимое число $C_{max}$ вычисляется по аналогичной формуле[1]

$$C_{max} = B^K - 1 .$$

**Переполнение разрядной сетки** — это ситуация, когда число, которое требуется сохранить, не умещается в имеющемся количестве разрядов вычислительного устройства.

Подчеркнем, что переполнение никак не связано с системой счисления: оно вызвано *ограниченным количеством разрядов* и не зависит от количества возможных значений в каждом из этих разрядов.

Числа бывают не только положительными, но и отрицательными. Введение знака числа не меняет сделанных выше выводов, только вместо нулевого минимального значения появляется отрицательное, которое зависит от разрядности (оно равно –9999 в обсуждаемом выше примере, если мы добавим еще один "знаковый" разряд).

Рассмотрим теперь, что будет, если наше устройство работает не только с целыми, но и с дробными числами. Пусть, например, один из четырех разрядов относится к целой части числа, а остальные три — к дробной. Конечно, эффект переполнения сохранится и здесь: максимально допустимое число равно 9,999. Кроме того, дробная часть числа тоже ограниченна, поэтому любое число, имеющее более трех цифр после запятой, не может быть представлено точно: младшие цифры придется отбрасывать (или округлять).


Не все вещественные числа могут быть представлены в компьютере точно.

При ограниченном числе разрядов дробной части существует некоторое минимальное ненулевое значение $C_{min}$, которое можно записать на данном индикаторе (в нашем примере это 0,001). В общем случае, если число записано в системе счисления с основанием $B$ и для хранения дробной части числа используется $F$ разрядов, имеем


$$C_{min} = B^{-F} .$$

Любое значение, меньшее чем $C_{min}$, неотличимо от нуля. Такой эффект принято называть *антипереполнением* (англ. *underflow* — переполнение "снизу").

Кроме того, два дробных числа, отличающиеся менее чем на $C_{min}$, для компьютера неразличимы. Например, 1,3212 и 1,3214 на нашем индикаторе выглядят совершенно одинаково.


Дополнительная погрешность появляется при переводе дробных чисел из десятичной системы счисления в двоичную. При этом даже некоторые "круглые" числа (например, 0,2) в памяти компьютера представлены неточно, потому что в двоичной системе они записываются как бесконечные дроби и их приходится округлять до заданного числа разрядов.

Так как вещественные числа хранятся в памяти приближенно, сравнивать их (особенно если они являются результатами сложных расчетов) необходимо с большой осторожностью. Пусть при вычислениях на компьютере получили $X = 10^{-6}$ и $Y = 10^6$. Дробное значение $X$ будет неточным, и произведение $X \cdot Y$ может незначительно отличаться от 1. Поэтому при сравнении вещественных чисел в компьютере условие "равно" использовать нельзя. В таких случаях числа считаются равными, если их разность достаточно мала по модулю. В данном примере нужно проверять условие $|1 - X \cdot Y| < \varepsilon$, где $\varepsilon$ — малая величина, которая задает нужную точность вычислений. К счастью, для большинства практических задач достаточно взять $\varepsilon$ порядка $10^{-2}$... $10^{-4}$, а ошибка компьютерных расчетов обычно значительно меньше[2] (не более $10^{-7}$).

## Различие между вещественными и целыми числами

Существуют величины, которые по своей природе могут принимать только целые значения, например, счетчики повторений каких-то действий, количество людей и предметов, координаты пикселей на экране и т.п. (возможно, вы видели известный мультфильм "В стране невыученных уроков", где полтора землекопа искали породившего их двоечника). Кроме того, кодирование нечисловых видов данных (текста, изображений, звука) сводится именно к целым числам.

Чтобы сразу исключить все возможные проблемы, связанные с неточностью представления в памяти вещественных чисел, целочисленные данные кодируются в компьютерах особым образом.

Целые и вещественные числа в компьютере хранятся и обрабатываются по-разному.

[1] Докажите эту формулу самостоятельно, например, подсчитав количество всех возможных комбинаций значений цифр в $K$ разрядах.

[2] Тем не менее встречаются ситуации, когда вычислительные трудности все же возникают: классический пример — разность близких по значению десятичных дробей, отличающихся в последних значащих цифрах.

Операции с целыми числами, как правило, выполняются значительно быстрее, чем с вещественными. Не случайно в ядре современных процессоров реализованы только целочисленные арифметические действия, а для вещественной арифметики используется специализированный встроенный блок — *математический сопроцессор*.

Кроме того, использование целых типов данных позволяет экономить компьютерную память. Например, целые числа в интервале от 0 до 255 в языке Паскаль можно хранить в переменных типа *byte*, которые занимают всего один байт в памяти. В то же время самое "короткое" вещественное число (типа *single*) требует четырех байт памяти.

Наконец, только для целых чисел определены операции деления нацело и нахождения остатка. В некоторых задачах они удобнее, чем простое деление с получением дробного (к тому же не совсем точного) результата: например, без них не обойтись при вычислении наибольшего общего делителя двух чисел.

Таким образом, для всех величин, которые не могут иметь дробных значений, нужно использовать целочисленные типы данных.

### Дискретность представления чисел

Из курса информатики вы знаете, что существует непрерывное и дискретное представление информации. Их принципиальное различие состоит в том, что дискретная величина может принимать конечное количество различных значений в заданном интервале, а непрерывная имеет бесконечно много возможных значений. Для нашего обсуждения важно, что

- целые числа дискретны;
- вещественные (действительные, дробные) числа непрерывны;
- современный компьютер работает только с дискретными данными.

Таким образом, для хранения вещественных чисел в памяти компьютера нужно выполнить *дискретизацию* — записать непрерывную величину в дискретной форме. При этом может происходить искажение данных, поэтому большинство трудностей в компьютерной арифметике (антипереполнение, приближенность представления дробной части и др.) связано именно с кодированием дробных чисел.

### Программное повышение точности вычислений

Современные модели процессоров Intel "умеют" обрабатывать 8-, 16-, 32- и 64-разрядные двоичные целые числа, а также (в математическом сопроцессоре) 32-, 64- и 80-разрядные вещественные числа. Для большинства практических задач такой разрядности вполне достаточно. Если для каких-либо особо точных расчетов требуется повысить разрядность вычислений, это можно сделать программно. Например, можно считать, что 4 последовательно хранящихся целых 64-разрядных числа — это единое "длинное" число, и написать программу обработки таких "удлиненных" чисел. Очень удобно хранить числа в виде последовательности десятичных цифр[3], правда, программы, выполняющие обработку таких чисел, получаются сложными и медленными.

Использование этих и других программных методов позволяет увеличить разрядность обрабатываемых чисел по сравнению с аппаратной разрядностью компьютера. Однако ограничение разрядности (и связанный с ним эффект переполнения) все равно остается: в программу заложено *конкретное* число разрядов, да и объем памяти компьютера конечен.

## Контрольные вопросы

1. Чем отличается компьютерная арифметика от "обычной"? Почему?
2. Почему диапазон чисел в компьютере ограничен? Связано ли это с двоичностью компьютерной арифметики?
3. Что такое переполнение разрядной сетки?
4. Какие проблемы появляются при ограниченном числе разрядов в дробной части?
5. Что называется антипереполнением? Что, по-вашему, опаснее для вычислений — переполнение или антипереполнение?
6. *Может ли антипереполнение сделать невозможными дальнейшие вычисления? (Ответ: может, если далее нужно делить на полученный результат.)
7. Сколько бит информации несет знаковый разряд? (Поскольку он имеет два допустимых значения, то хранит 1 бит информации.)
8. Приведите примеры величин, которые по своему смыслу могут иметь только целые значения.
9. Какие преимущества дает разделение в компьютере целых и вещественных (дробных) чисел?
10. Какая математическая операция между двумя целыми числами может дать в результате нецелое число?
11. Чем отличается деление для целых и вещественных чисел?

[3] Такие задачи часто даются на школьных олимпиадах по информатике; для них даже придумано специальное название — *"длинная" арифметика*.

12. Вспомните определение дискретных и непрерывных величин. Какие множества чисел в математике дискретны, а какие — нет? Ответ обоснуйте.

13. Объясните, почему ограниченность разрядов дробной части приводит к нарушению свойства непрерывности.

14. Можно ли организовать вычисления с разрядностью, превышающей аппаратную разрядность компьютера? Попробуйте предложить способы решения этой задачи.

## Задачи

1. Вычислите максимальное целое значение для 8-разрядного двоичного числа и 3-разрядного десятичного (считать, что значения не бывают отрицательными). Какое из них оказалось больше?

2. Вычислите максимальное целое положительное значение для 16- и 32-битного двоичного числа. ($2^{16} - 1 = 65\,535$; $2^{32} - 1 = 4\,294\,967\,295$)

3. Пользуясь калькулятором, вычислите границу антипереполнения для чисел с 16 двоичными разрядами в дробной части. Напишите два близких дробных числа, которые для полученного значения окажутся неразличимыми. ($2^{-16} \approx 0{,}0000153$)

4*. Вычислите минимально возможное отрицательное значение для 16-разрядных двоичных чисел (учесть, что один из двоичных разрядов является знаковым). ($-2^{(16-1)} = -32\,768$)

5*. Придумайте простую вычислительную задачу, в которой для хранения результата не хватает 16 двоичных разрядов. (Классический пример — факториал числа 9, т.е. $1 \cdot 2 \cdot 3 \cdot \ldots \cdot 9 = 362\,880 > 2^{16}$.)

## Хранение целых чисел в памяти компьютера

### Целые числа без знака

Беззнаковые (англ. *unsigned*) типы данных, то есть величины, не имеющие отрицательных значений, широко используются в вычислительной технике. Дело в том, что в задачах, решаемых на компьютерах, есть много таких значений: всевозможные счетчики (количество повторений циклов, число параметров в списке или символов в тексте), количество предметов и др.

Чтобы закодировать целое число без знака, достаточно перевести его в двоичную систему счисления и дополнить слева нулями до нужной разрядности. Например, число 28 записывается в 8-разрядную ячейку так:

0001 1100

Это же число в 16-разрядном представлении будет иметь слева еще 8 нулей. Восьмиразрядные коды некоторых характерных чисел приведены в таблице.

| $X_{10}$ | 0 | 1 | ... | 127 | 128 | 129 | ... | 255 |
|---|---|---|---|---|---|---|---|---|
| $X_{16}$ | 00 | 01 | ... | 7F | 80 | 81 | ... | FF |
| $X_2$ | 0000 0000 | 0000 0001 | ... | 0111 1111 | 1000 0000 | 1000 0001 | ... | 1111 1111 |

Минимальное значение для беззнаковых целых чисел всегда равно 0 (все разряды нулевые), а максимальное число $X_{max} = 2^K - 1$ состоит из всех единиц и определяется разрядностью (количеством бит) $K$.

| $K$ (бит) | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| $X_{max} = 2^K - 1$ | 255 | 65 535 | 4 294 967 295 | 18 446 744 073 709 551 615 |

Возникает вопрос: а что будет, если увеличить максимальное число в $K$-битной ячейке на единицу? Рассмотрим случай $K = 8$ и попытаемся прибавить единицу к числу $255_{10} = 1111\ 1111_2$. Добавляя дополнительный бит слева, получим:

```
  0 | 1111 1111
+ 0 | 0000 0001
  1 | 0000 0000
```

Отбросив несуществующий дополнительный разряд[4], получаем $255 + 1 = 0$. Как ни странно, именно это произойдет в реальном компьютере. Говорят, что при $K$ разрядах арифметика выполняется "по модулю $2^K$", т.е. при $K = 8$ имеем[5]:

$$(255+1)\ \mathbf{mod}\ 256 = 256\ \mathbf{mod}\ 256 = 0.$$

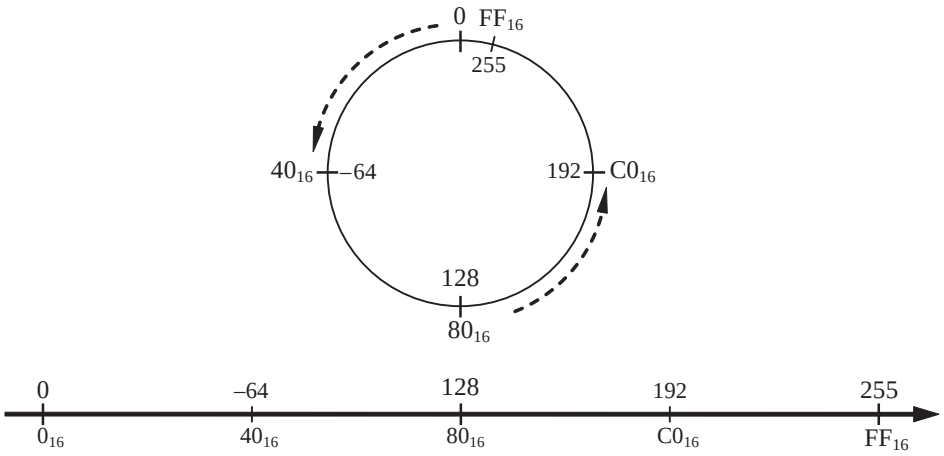
С другой стороны, вычитая единицу из минимального значения 0, к которому добавлен старший разряд за пределами 8-битной ячейки, получим $1111\ 1111_2 = 255_{10}$ (проверьте это самостоятельно).

---

[4] На самом деле для того чтобы обнаружить факт переполнения, этот разряд сохраняется в специальном управляющем бите процессора, который называется *битом переноса*.

[5] Здесь запись **a mod b** обозначает остаток от деления **a** на **b**.

Можно заметить, что при многократном увеличении числа на единицу мы доходим до максимального значения и *скачком возвращаемся к минимальному*. При вычитании единицы получается обратная картина — дойдя до минимума (нуля), мы сразу перескакиваем на максимум (255). Поэтому для изображения допустимого диапазона чисел лучше подходит не отрезок числовой оси (как в математике), а окружность:




Факт переполнения всегда фиксируется процессором, но выполнение программы не прерывается. Программе (точнее, программисту) предоставляется возможность как-то реагировать на переполнение или "не заметить" его.

## Целые числа со знаком

Теперь рассмотрим числа со знаком (англ. *signed*). Для того чтобы различать положительные и отрицательные числа, в двоичном коде выделяется один бит для хранения знака числа — *знаковый разряд*. По традиции для этого используют самый старший бит, причем нулевое значение в нем соответствует знаку "плюс", а единичное — "минусу". Ноль формально является положительным числом, так как все его разряды, включая знаковый, нулевые.

Поскольку один бит выделяется для хранения информации о знаке, ровно половина из всех $2^K$ чисел будут отрицательными. Учитывая, что одно значение — нулевое, положительных чисел будет на единицу меньше, т.е. допустимый диапазон значений оказывается несимметричным.

Положительные числа записываются в знаковой форме так же, как и в беззнаковой, но для значения остается на один разряд меньше. А как поступить с отрицательными числами? Первое, что приходит в голову, это кодировать отрицательные значения точно так же, как и положительные, только записывать в старший бит единицу. Такой способ кодирования называется *прямым кодом*. Несмотря на свою простоту и наглядность, он не применяется в компьютерах для представления целых чисел[6]. Это неудобно, потому что действия над числами, записанными в прямом коде, выполняются по-разному для разных сочетаний знаков. Поэтому в современных компьютерах отрицательные числа кодируются с помощью другого метода, который менее нагляден, но позволяет выполнять арифметические действия с положительными и отрицательными числами по одному и тому же алгоритму.

Как же представить целые числа, чтобы арифметика выглядела максимально просто? Попробуем, например, вычислить код, соответствующий числу –1. Для этого просто вычтем из нуля единицу:

```
  1 0000 0000
- 0 0000 0001
-------------
  0 1111 1111
```

Чтобы вычитание "состоялось", придется занять из несуществующего старшего бита единицу, что не очень естественно, но зато быстро приводит к правильному результату[7]. Заметим, что фактически мы вычитали не из 0, а из 256. В общем случае вычисление происходит по формуле $2^K - X$, где для данного примера $K = 8$, а $X = 1$.

Однако предложенный способ перевода не слишком хорош, поскольку мы использовали дополнительный "несуществующий" разряд. Вместо этого можно использовать равносильный алгоритм:

$$256 - X = (255 - X) + 1 = \texttt{not}\ X + 1$$

Здесь "**not**" обозначает логическую операцию "НЕ" (инверсию), применяемую к каждому биту числа отдельно (все нули заменяются на единицы и наоборот).

Итак, для получения кода целого отрицательного числа ($-X$) нужно:

**Алгоритм А0**

1) Выполнить инверсию каждого разряда двоичного представления положительного числа $X$ (такой код называется *обратным*).

2) К полученному результату прибавить единицу.

[6] Тем не менее прямой код используется в представлении вещественных чисел (см. ниже).

[7] Для проверки можно прибавить к полученному коду единицу, в результате должен получиться ноль.

В результате получается *дополнительный код* — он *дополняет* число до $2^K$.

Алгоритм A0 приводится в большинстве учебников, но его можно немного изменить так, чтобы облегчить человеку "ручные" вычисления:

**Алгоритм A1**

1) Вычислить число $X - 1$ и перевести его в двоичную систему.
2) Выполнить инверсию каждого разряда результата.

Оба алгоритма дают одинаковые результаты, но A1 для человека существенно проще, потому что ему легче отнять единицу в "родной" десятичной системе, чем прибавлять ее в двоичной (при использовании A0).

Наконец, оба пункта алгоритма A0 можно объединить, получив еще один вариант:

**Алгоритм A2**

Выполнить инверсию всех старших битов числа, кроме последней (младшей) единицы и тех нулей, которые стоят после нее.

Например, определим дополнительный код числа "–16", которое хранится в 8-разрядной ячейке. Здесь $X = 16$. Используя алгоритмы A0 и A1, получаем:

| | **A0** |
|---|---|
| $X_2$ | $0001\ 0000_2$ |
| **not** | $1110\ 1111_2$ |
| +1 | $1111\ 0000_2$ |

| | **A1** |
|---|---|
| $X - 1$ | 15 |
| $(X - 1)_2$ | $0000\ 1111_2$ |
| **not** | $1111\ 0000_2$ |

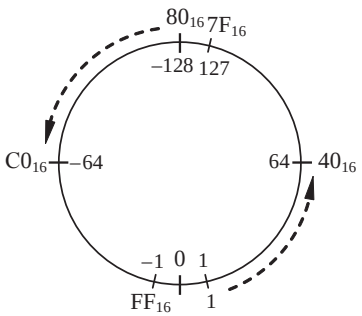
Применение алгоритма A2 к числу $16 = 00010000_2$ сводится к замене первых трех нулей единицами: $1111\ 0000_2$.

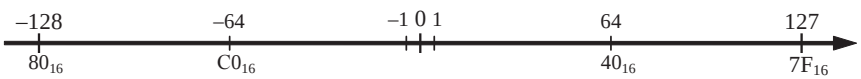
Для проверки можно сложить полученный результат с исходным числом и убедиться, что сумма обратится в ноль (перенос из старшего разряда не учитываем). Повторное применение любого из алгоритмов A0–A2 всегда приводит к восстановлению первоначального числа (убедитесь в этом самостоятельно). Это свойство также удобно использовать для проверки.

В следующей таблице показаны шестнадцатеричные и двоичные коды некоторых характерных 8-разрядных чисел.

| $X_{10}$ | –128 | –127 | ... | –1 | 0 | 1 | ... | 127 |
|---|---|---|---|---|---|---|---|---|
| $X_{16}$ | 80 | 81 | ... | FF | 00 | 01 | ... | 7F |
| $X_2$ | 1000 0000 | 1000 0001 | ... | 1111 1111 | 0000 0000 | 0000 0001 | ... | 0111 1111 |

Обратите внимание на скачок при переходе от –1 к 0 и на два граничных значения: 127 и –128. "Кольцо" для чисел со знаком выглядит так:

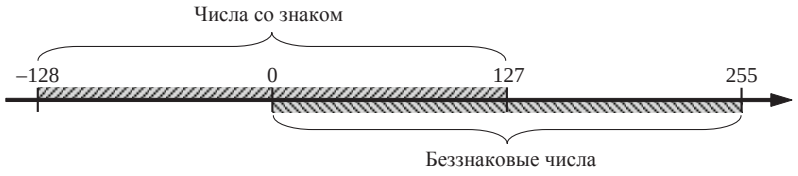



Чтобы сравнить коды целых чисел без знака и со знаком, объединим обе таблицы:

| Код | 0 | 1 | 2 | ... | 7F | 80 | 81 | ... | FE | FF |
|---|---|---|---|---|---|---|---|---|---|---|
| Без знака | 0 | 1 | 2 | ... | 127 | 128 | 129 | ... | 254 | 255 |
| Со знаком | 0 | 1 | 2 | ... | 127 | –128 | –127 | ... | –2 | –1 |

Общее количество значений со знаком и без него одинаково, но их диапазоны сдвинуты друг относительно друга на числовой оси:



В наших рассуждениях использовались 8-разрядные числа, но все выводы справедливы для чисел любой разрядности. От числа разрядов $K$ зависят только граничные значения $X_{max}$ и $X_{min}$, приведенные в таблице:

| $K$ | 8 | 16 | 32 | 64 |
|---|---|---|---|---|
| $X_{max}$ | 127 | 32 767 | 2 147 483 647 | 9 223 372 036 854 775 807 |
| $X_{min}$ | –128 | –32 768 | –2 147 483 648 | –9 223 372 036 854 775 808 |

Хотя дополнительный код гораздо менее нагляден, чем прямой, он значительно упрощает выполнение арифметических операций в компьютере. Например, вместо вычитания используется сложение с дополнительным кодом вычитаемого, поэтому не нужно проектировать специальное устройство для вычитания чисел.

## Контрольные вопросы

1. Чем отличается представление в компьютере целых чисел со знаком и без знака?
2. Приведите примеры величин, которые всегда имеют целые неотрицательные значения.
3. Как представлены в компьютере целые числа без знака?
4. Как изменится диапазон представления чисел, если увеличить количество разрядов на 1, на 2, на $n$?
5. Какое максимальное целое беззнаковое число можно записать с помощью $K$ двоичных разрядов? Что произойдет, если прибавить единицу к этому максимальному значению?
6. Как действует процессор при переполнении?
7. Почему максимальное положительное и минимальное отрицательное значения у целых двоичных чисел со знаком имеют разные абсолютные значения?
8. Верно ли, что положительные числа кодируются одинаково в знаковом и беззнаковом форматах?
9. Сформулируйте различные алгоритмы получения дополнительного кода для отрицательного числа.
10*. Докажите, что алгоритмы А0, А1 и А2 всегда дают один и тот же результат.
11. Что получится, если правила перевода в дополнительный код применить к отрицательному числу?
12. Как можно проверить правильность перевода в дополнительный код?
13. Какое минимальное отрицательное значение можно записать с помощью $K$ двоичных разрядов?
14*. Может ли быть переполнение при сложении двух отрицательных чисел? Какой знак будет у результата?
15. В чем главное преимущество дополнительного кода при кодировании отрицательных чисел?
16. Почему компьютер может обойтись без вычитания?

## Задачи

1. Цвет пикселя изображения кодируется как целое беззнаковое число. Найдите максимальное количество цветов при двух- и трехбайтовом кодировании.
2. Используя арифметику 8-разрядных чисел без знака, выполните действия: 250 + 10 и 8 — 10. Объясните полученные результаты. (Ответы: 4 и 254.)
3. Выполните сложение десятичных чисел 65 530 + 9 в 16-битной арифметике без знака. (Ответ: 3.)
4. Выполните сложение десятичных чисел 32 760 + 9 в 16-битной арифметике со знаком. (Ответ: –32 767.)
5. Переведите в дополнительный код отрицательные числа “–1”, “–10”, “–100” и запишите их с помощью 8 двоичных разрядов.
6. Как выглядит прямой код для отрицательных чисел “–1”, “–10”, “–100”, записанных с помощью 8 двоичных разрядов? (Ответ: 1000 0001, 1000 1010, 1110 0100.)
7. Рассматриваются 8-разрядные числа со знаком. Какие из приведенных шестнадцатеричных чисел отрицательные: 1, 8, F, 10, 18, 20, 30, 3F, 40, 70, 7F, 80, 90, A1, CC, F0, FF? Как это можно быстро определять?
8. Отвечая на вопрос учителя: как вычислить максимальное положительное и минимальное отрицательное значения у целых $K$-разрядных двоичных чисел со знаком, ученик ответил кратко: $2^K$. В чем он ошибся, а в чем нет? Вычислите правильные значения для $K = 12$.
9. Каков будет результат операции 127 + 3 в 8-разрядной арифметике со знаком? Объясните полученный результат. (Ответ: –126.)
10*. Факториалом называется произведение последовательных целых чисел, например, $3! = 1 \cdot 2 \cdot 3 = 6$. Вычисления выполняются в 16-разрядной целочисленной арифметике со знаком. Для какого максимального значения $n$ удастся вычислить $n!$ и что получится при вычислении $(n + 1)!$? (Ответ: 7! = 5040; 8! = –25 216, что говорит о переполнении — см. предыдущую задачу.)

# Операции с целыми числами

## Сложение и вычитание

Сложение и вычитание требуются не только для расчетов по формулам, но и для организации вычислений. Например, для того чтобы повторить какое-то действие $R$ раз, используют переменную-счетчик, к которой после каждого выполнения этого действия прибавляют единицу, а затем результат сравнивают с $R$. Вместо этого можно сразу записать в счетчик значение $R$ и после каждого повторения вычитать из него единицу, пока не получится ноль[8].

Благодаря тому, что отрицательные числа кодируются в дополнительном коде, при сложении можно не обращать внимания на знаки слагаемых, то есть *со знаковым разрядом обращаются точно так же, как и со всеми остальными.*

Например, сложим числа $5_{10}$ ($0000\ 0101_2$) и $-9_{10}$ ($1111\ 0111_2$), используя 8-разрядную двоичную арифметику. Применим сложение столбиком, не задумываясь о знаках чисел:

```
  0000 0101
+ 1111 0111
-----------
  1111 1100
```

Для расшифровки получившегося отрицательного числа применим к нему схему получения дополнительного кода: $1111\ 1100 \rightarrow 0000\ 0100_2 = 4_{10}$. Таким образом, результат равен $-4_{10}$, что совпадает с правилами "обычной" арифметики.

При сложении двух чисел с одинаковыми знаками может случиться переполнение — сумма будет содержать слишком большое количество разрядов. Покажем, как это выглядит для положительных и отрицательных чисел.

Сложим десятичные числа 96 и 33. Их сумма 129 выходит за восьмибитную сетку. Для того чтобы обнаружить переполнение, добавим к обоим слагаемым еще один старший бит, совпадающий со знаковым:

```
  0 | 0 1 1 0 0 0 0 0
+ 0 | 0 0 1 0 0 0 0 1
-----------------------
  0 | 1 0 0 0 0 0 0 1
  C   S
```

Знаковый разряд $S$ результата равен 1, т.е. сумма получилась отрицательной, хотя оба слагаемых положительны! Старший бит результата, не поместившийся в разрядную сетку, попадает в специальную ячейку процессора, которая называется *битом переноса 0* (от англ. *carry* — перенос)[9]. Если бит переноса не совпадает со знаковым битом результата, произошло переполнение и результат неверный.

То же самое получается, если сложить два достаточно больших по модулю отрицательных числа, например, "–96" и "–33". Добавим к кодам обоих чисел один старший разряд, равный знаковому разряду:

```
  1 | 1 0 1 0 0 0 0 0
+ 1 | 1 1 0 1 1 1 1 1
-----------------------
  1 | 0 1 1 1 1 1 1 1
  C   S
```

Получается, что в результате бит $S$ равен 0, хотя ответ должен быть отрицательным. Биты $C$ и $S$ не совпадают, это говорит о том, что произошло переполнение. Несложно проверить (сделайте это самостоятельно), что если переполнения нет, значения битов $S$ и $C$ всегда одинаковы независимо от знаков слагаемых.

Сложение многоразрядных двоичных чисел в компьютере выполняет специальное устройство — *сумматор* (см. параграф 3.7). Как мы уже говорили, вычитание сводится к сложению с дополнительным кодом вычитаемого, поэтому отдельного "блока вычитания" в компьютере нет.

## Умножение и деление

Умножение и деление выполнять труднее, чем сложение и вычитание. Вспомните, например, что в математике умножение часто заменяют многократным сложением, а деление — многократным вычитанием.

К двоичным числам можно применять обычную схему умножения "столбиком". Перемножим, например, числа $9_{10}$ ($0000\ 1001_2$) и $5_{10}$ ($0000\ 0101_2$).

```
    00001001
  × 00000101
-------------
    00001001
+ 00000000
  00001001
-------------
  0000101101
```

Легко проверить, что это число равно $45_{10}$.

[8] Второй вариант более эффективен, потому что процессор автоматически сравнивает результат очередного действия с нулем (см. 5.4.1).

[9] При программировании на языках высокого уровня бит переноса недоступен.

В сравнении с десятичной системой здесь есть серьезное упрощение: первый сомножитель умножается на единицу (в этом случае результат равен ему самому) или на ноль (результат — 0). Поэтому компьютерное умножение целых чисел состоит из следующих элементарных действий:

- вычисление очередного произведения в зависимости от младшего бита второго сомножителя: оно равно нулю (если этот бит нулевой) или первому множителю (если бит равен единице);
- сдвиг содержимого сумматора, накапливающего сумму, влево на 1 разряд;
- сложение содержимого сумматора с очередным произведением;
- сдвиг второго сомножителя вправо на 1 разряд (при этом следующий бит попадет в младший разряд).

Таким образом, удается построить схему умножения без использования таблицы умножения. Заметим, что умножение — это довольно трудоемкая операция, и для ее ускорения конструкторы используют различные "хитрые" приемы. Поэтому в реальных компьютерах все может выглядеть значительно сложнее, чем в учебном примере.

Умножение, как и сложение, выполняется одинаково для положительных и отрицательных чисел (в дополнительном коде). Если в нашем примере вместо числа 9 подставить "–9", то получится:

```
      11110111
  ×   00000101
  ----------
      11110111
+   00000000
    11110111
  ----------
  1001101001 1
```

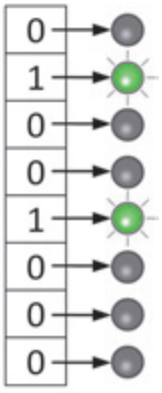
Оставив только 8 младших битов, можно убедиться (применяя алгоритмы А0–А2), что результат — это дополнительный код числа "–45".

Теория деления нацело оказывается намного сложнее, чем для умножения, поэтому мы ее обсуждать не будем.

## Сравнение

В отличие от арифметических действий операция сравнения *по-разному* выполняется для чисел со знаком и без него. Еще раз внимательно посмотрим на таблицы кодов 8-битных чисел, приведенные в 4.2.2. Если сравниваемые коды не превышают $7F_{16}$, то оба числа положительны и сравнение однозначно. Если это не так, то сравнение чисел с учетом и без учета знака дает разные результаты. Например, для беззнаковых чисел $81_{16}$ ($129_{10}$) больше, чем $7F_{16}$ ($127_{10}$). Для чисел со знаком, наоборот, отрицательное значение $81_{16}$ ($-127_{10}$) будет меньше, чем $7F_{16}$ ($127_{10}$). Поэтому современные процессоры имеют разные команды для сравнения чисел со знаком и без знака. Чтобы не путаться, в первом случае обычно используют термины "больше/меньше", а во втором — "выше/ниже".

## Поразрядные логические операции




Как известно из математической логики, обработка истинности и ложности высказываний может быть представлена как набор операций с двоичными кодами. Оказывается, что логические операции, введенные первоначально для обработки логических данных, можно формально применить отдельно к битам двоичного числа, и такой подход широко используется в современных компьютерах.

Рассмотрим электронное устройство для управления гирляндой лампочек. Состояние каждой из них будет задаваться отдельным битом в некотором управляющем регистре: если бит равен нулю, лампочка выключена, если единице — включена. Для получения различных световых эффектов (типа "бегущих огней") требуется зажигать или гасить отдельные лампочки, менять их состояние на противоположное и т.д. Точно так же биты регистров используются для управления внешними устройствами.

Будем применять логические операции *к каждому* биту числа, как обычно считая, что 1 соответствует значению "истинно", а 0 — "ложно". Эти операции часто называют *поразрядными*, или *битовыми*, поскольку действия совершаются над каждым разрядом в отдельности, независимо друг от друга[10].

Введем несколько терминов, которые используются в литературе по вычислительной технике. *Сброс* — это запись в бит нулевого значения, а *установка* — запись единицы. Таким образом, если бит в результате какой-то операции становится равным нулю, то говорят, что он сбрасывается. Аналогично, когда в него записана единица, говорят, что бит установлен.

*Маска* — это константа, которая определяет область применения логической операции к битам многоразрядного числа. С помощью маски можно скрывать (защищать) или открывать для выполнения операции отдельные биты[11].

[10] Для сравнения, сложение (как и другие арифметические действия) не является поразрядной операцией, поскольку возможен перенос из младшего разряда в более старший.

[11] Использование маски аналогично выделению области рисунка в графическом редакторе — для выделенных пикселей маска равна 1, для остальных — нулю.

Основные логические операции в современных процессорах — это “И” (**and**), “ИЛИ” (**or**), “исключающее ИЛИ” (**xor**) и “НЕ” (**not**).

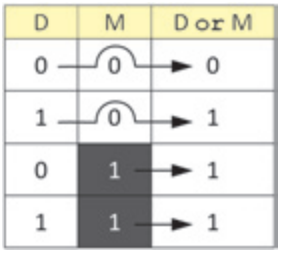
**Логическое “И”** (**and**). Если обозначить через *D* содержимое некоторого бита данных (англ. *data* — данные), а через *M* — значение соответствующего ему бита маски (англ. *mask* — маска), то получим следующую таблицу.


Из таблицы видно, что при выполнении логического “И” нулевой бит в маске всегда сбрасывает (делает равным нулю) соответствующий бит результата, а единичный бит позволяет сохранить значение *D* (как бы пропускает его, открывая “окошко”).

С помощью логической операции “И” можно *сбросить* отдельные биты числа (те, для которых маска нулевая), не меняя значения остальных битов (для которых в маске стоят единицы).

Например, операция *X* **and** 1 сбросит у любого числа *X* все биты, кроме самого младшего. С помощью этого приема легко узнать, является ли число четным: остаток от деления на 2 равен последнему биту!

**Логическое “ИЛИ”**. Вспомнив таблицу истинности логической операции “ИЛИ” (**or**), можно обнаружить, что ноль в маске сохраняет бит (*X* **or** 0 = *X*), а единица — устанавливает соответствующий бит результата (*X* **or** 1 = 1).




С помощью логической операции “ИЛИ” можно *установить* отдельные биты числа (те, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

Например, операция *X* **or** $80_{16}$ установит старший бит восьмиразрядного числа *X*, формально сделав его тем самым отрицательным.

Таким образом, используя операции “И” и “ИЛИ”, можно сбрасывать и устанавливать любые биты числа, то есть строить любой нужный двоичный код. Где это может пригодиться? Рассмотрим примеры решения конкретных задач.

**Пример 1.** На клавиатуре набраны 3 цифры, образующие значение целого числа без знака. Определить, какое число было введено.

При нажатии клавиши на клавиатуре в компьютер поступает код нажатой клавиши. Выпишем десятичные и шестнадцатеричные коды всех символов, обозначающих цифры:

| Символ | '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' |
|---|---|---|---|---|---|---|---|---|---|---|
| $X_{10}$ | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 |
| $X_{16}$ | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |

Будем пользоваться шестнадцатеричными кодами: как видно из таблицы, их связь с цифрами числа гораздо проще. Чтобы получить числовое значение цифры из кода символа *X*, достаточно сбросить его старшие 4 бита, не изменяя значение 4 младших. Для этого нужно использовать операцию *X* **and** 0F.

Пусть $S_1$ — код первого введенного символа, $S_2$ — второго, $S_3$ — третьего, а *N* обозначает искомое число. Тогда алгоритм перевода кодов символов в число выглядит так:

1. $N = 0$.
2. $W = S_1$ **and** 0F (выделяем первую цифру).
3. $N = 10 \cdot N + W$ (добавляем ее к числу).
4. $W = S_2$ **and** 0F (выделяем вторую цифру).
5. $N = 10 \cdot N + W$ (добавляем ее к числу).
6. $W = S_3$ **and** 0F (выделяем третью цифру).
7. $N = 10 \cdot N + W$ (добавляем ее к числу).

Пусть, например, набраны символы ‘1’, ‘2’ и ‘3’. Тогда по таблице находим, что $S_1 = 31$, $S_2 = 32$ и $S_3 = 33$. Значение $W$ на втором шаге вычисляется так:

```
and   0011 0001
      0000 1111
      ---------
      0000 0001
```

Так как $W = 1$, на третьем шаге получаем $N = 1$. Следующая пара шагов — четвертый и пятый, дают результаты $W = 2$ и $N = 12$ соответственно. Наконец, результат завершающих шагов — $W = 3$ и $N = 123$.

Такая процедура используется в каждом компьютере: именно так коды цифровых символов, которые вы набрали, преобразуются в числа, с которыми компьютер выполняет арифметические действия.

**Пример 2.** Создадим структуру данных $S$, которая отражает, есть или нет в некотором числе каждая из цифр от 0 до 9. В математике такая структура называется *множеством*. Для хранения $S$ будем использовать 16-разрядное целое число.

| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

Договоримся, что младший бит числа имеет номер 0 и хранит информацию о том, есть ли во множестве цифра 0 (если этот бит равен 0, такой цифры нет, если равен 1, то есть). Аналогично первый бит (второй по счету справа) показывает, есть ли во множестве цифра 1, и т.д. Старшие биты 10–15 при этом не используются. Например, во множестве, изображенном на рисунке, есть только цифры 0, 3, 6 и 9.

Для записи элементов во множество и проверки их наличия удобно использовать логические операции. Рассмотрим для примера пятый элемент множества (бит 5). Маска, которая потребуется для обращения к нему, — это единица в пятом разряде и нули во всех остальных, т.е. $M = 00\,20_{16}$. С ее помощью можно добавить элемент ко множеству с помощью операции “ИЛИ”: $S = S$ **or** $M$. А узнать, есть ли во множестве интересующая нас цифра, можно, выделив соответствующий бит с помощью логического “И” ($P = S$ **and** $M$) и проверив результат на равенство нулю.

**“Исключающее ИЛИ”**. Как видно из таблицы истинности, операция “исключающее ИЛИ” (**xor**) не изменяет биты, когда маска нулевая, и меняет на противоположные при единичной маске.

| D | M | D xor M |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

Например, команда $Y = Y$ **xor** FF делает инверсию всех битов 8-разрядного целого числа. Напомним, что это один из этапов получения дополнительного кода отрицательных чисел.

С помощью логической операции “исключающее ИЛИ” можно выполнить *инверсию* отдельных битов числа (тех, для которых маска единичная), не меняя значения остальных битов (для которых в маске стоят нули).

**Пример 3.** Пусть $X$ — это результат выполнения некоторого вычислительного теста, а $Y$ — то, что ожидалось получить (“правильное” значение). Нужно определить, в каких разрядах различаются эти числа (для инженера это очень полезная подсказка, где искать неисправность).

Предположим, что $X = 7$ и $Y = 3$. В результате операции $X$ **xor** $Y$ устанавливаются в единицу только те разряды, которые в этих числах не совпали, а остальные сбрасываются[12]. В данном случае находим, что числа отличаются только одним битом:

```
xor   0000 0111
      0000 0011
      ---------
      0000 0100
```

**Пример 4.** Используя логическую операцию “исключающее ИЛИ”, можно шифровать любые данные. Покажем это на примере простого текста “**2*2=4**”.

Выберем любую маску, например, $23_{10} = 0001\,0111_2$. Эта маска представляет собой *ключ шифра* — зная ключ, можно расшифровать сообщение. Возьмем первый символ — цифру “2”, которая имеет код $50_{10} = 0011\,0010_2$, и применим “исключающее ИЛИ” с выбранной маской:

```
xor   0011 0010
      0001 0111
      ---------
      0010 0101
```

[12] Профессиональные программисты часто используют операцию **xor** для обнуления переменной: команда $R := R$ **xor** $R$ запишет в переменную $R$ ноль, независимо от ее начального значения.

Полученное значение 0010 $0101_2 = 37_{10}$ — это код символа “%”. Для расшифровки применим к этому коду “исключающее ИЛИ” с той же маской:

```
      0010 0101
xor   0001 0111
-------------
      0011 0010
```

В результате получили число 50 — код исходной цифры “2”.

Повторное применение операции “исключающее ИЛИ” с той же маской восстанавливает исходное значение, т.е. эта логическая операция *обратима*.

Если применить такую процедуру шифрования ко всем символам текста “**2*2=4**”, то получится зашифрованный текст “%**=**%**7*7#**”.

Обратимость операции “исключающее ИЛИ” часто используется в компьютерной графике для временного наложения одного изображения на другое. Это может потребоваться, например, для выделения области с помощью инвертирования ее цвета.

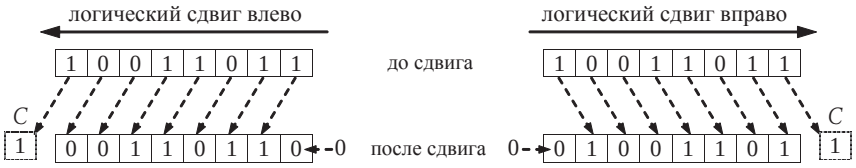
**Логическое “НЕ”** (инвертирование, инверсия, **not**) активно применяется, например, при получении дополнительного кода отрицательных чисел (см. алгоритм А0 выше).

В отличие от всех рассмотренных выше операций “НЕ” — это *унарная* операция, т.е. она действует на все биты *одного* числа, маски здесь не используются.

## Сдвиги

Об операции сдвига вспоминают гораздо реже, чем она того заслуживает. Сдвиги незаменимы тогда, когда требуется проделать ту или иную обработку *каждого* бита, входящего в число. Наконец, сдвиги двоичного числа позволяют быстро умножить или разделить число на степени двойки: 2, 4, 8 и т.д. Поэтому программисты очень ценят и широко применяют всевозможные разновидности сдвигов.

Идея операции сдвига довольно проста: все биты кода одновременно сдвигаются в соседние разряды[13] влево или вправо.




Отдельно надо поговорить о двух крайних битах, у которых “нет соседей”. Для определенности обсудим сдвиг влево. Для самого младшего бита (на рисунке он крайний справа) данные взять неоткуда, поэтому в него просто заносится ноль. Самый старший (крайний слева) бит должен потеряться, так как его некуда сохранить. Чтобы данные не пропали, содержимое этого разряда копируется в специальный бит переноса *C*, с которым может работать процессор.

Рассмотренный тип сдвига обычно называется **логическим сдвигом**. Его можно использовать для быстрого умножения и деления. Рассмотрим, например, 8-разрядный двоичный код 0000 1100, который представляет число $12_{10}$. Выполнив логический сдвиг влево, получим 0001 1000, т.е. число $24_{10}$, которое вдвое больше! Это не случайность: вспомните, что происходит, если к десятичному числу справа приписать дополнительный ноль, например, 34 → 340.

При сдвиге вправо любое четное число уменьшается ровно в 2 раза. В случае нечетного значения происходит деление нацело, при котором остаток отбрасывается. Например, из 0001 0001 = $17_{10}$ при сдвиге вправо получается 0000 1000 = $8_{10}$.

Логический сдвиг влево на 1 разряд увеличивает целое положительное двоичное число вдвое, а сдвиг вправо — делит на 2 нацело.

**Пример**. Для умножения числа, находящегося в ячейке *Z*, на 10 можно использовать такой алгоритм:

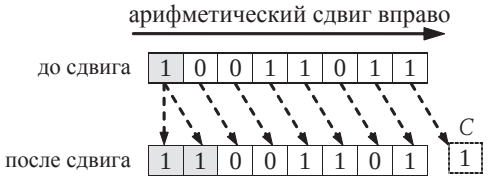
1. Сдвиг влево *Z* (получаем $2Z_0$, где $Z_0$ — исходное число).
2. $X = Z$ (сохраним $2Z_0$).
3. Сдвиг на 2 бита влево *X* (вычислили $8Z_0$).
4. $X = X + Z$ ($X = 8Z_0 + 2Z_0 = 10Z_0$).

Для некоторых компьютеров такая последовательность выполняется быстрее, чем стандартная операция умножения.

Посмотрим, что получится для отрицательных чисел. Сдвинем влево код 1111 1000 (8-разрядное представление числа “–8”): получится 1111 0000. Легко проверить, что это дополнительный код числа “–16”, то есть значе-

[13] Аппаратно сдвиг реализуется необычайно просто и изящно: регистр, содержащий число, сбрасывается в ноль, при этом из тех разрядов, где исчезла единица, электрический импульс проходит в соседние и устанавливает их в единицу. Важно, что все разряды обрабатываются одновременно.

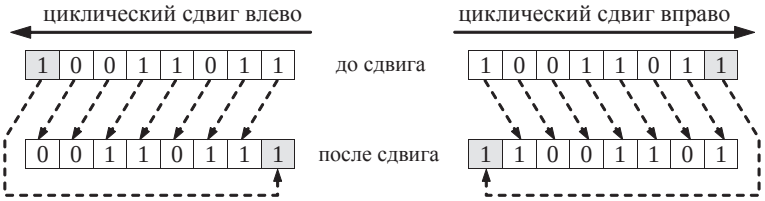
ние удвоилось! Но со сдвигом вправо ничего не получается: из 1111 1000 получаем 0111 1100 — это код положительного числа! Дело в том, что при сдвиге вправо отрицательных чисел, в отличие от положительных, старший разряд надо заполнять не нулем, а единицей! Чтобы исправить положение, вводится еще одна разновидность сдвига — **арифметический сдвиг**. Его единственное отличие от логического состоит в том, что старший (знаковый) бит не меняется, то есть знак числа остается прежним.




Если применить арифметический сдвиг к коду 1111 1000, получается 1111 1100 — дополнительный код числа "–4", то есть произошло деление на 2. В качестве упражнения проверьте, как ведет себя отрицательное нечетное число при арифметическом сдвиге вправо.

Арифметический сдвиг влево не требуется, поскольку он ничем не отличается от обычного логического сдвига.

То, что в результате логических сдвигов содержимое крайних разрядов теряется, не всегда удобно. Поэтому в компьютере предусмотрен **циклический сдвиг**, при котором бит из одного крайнего разряда переносится в другой ("по циклу"):




Циклический сдвиг позволяет "просмотреть" все биты и вернуться к исходному значению. Если сделать последовательно 8 циклических сдвигов 8-битного числа, каждый его бит на каком-то шаге окажется на месте младшего разряда, где его можно выделить с помощью логической операции "И" с маской 1. Так можно "просматривать" не только младший, но и любой другой разряд (например, для выделения старшего разряда нужно использовать маску $80_{16}$).

## Контрольные вопросы

1. Покажите на примере, как складываются два положительных целых числа, записанные в 8-разрядные ячейки. Что изменится, если числа будут отрицательными?
2. Что такое дополнительный код? Сформулируйте правила получения дополнительного кода числа.
3. При каких комбинациях знаков слагаемых в результате сложения может возникнуть переполнение? (Ответ: если знаки одинаковы.)
4. Какое устройство выполняет в компьютере сложение? Вспомните, что вы о нем знаете.
5. Почему не нужно разрабатывать специальное устройство для вычитания целых чисел?
6. Перемножьте столбиком два положительных целых числа в двоичной системе счисления. Изменится ли алгоритм выполнения операции, если у одного из сомножителей поменять знак?
7. Почему коды чисел со знаком и без знака нужно сравнивать по-разному?
8. Что такое поразрядные операции? Приведите примеры.
9. Почему арифметические операции нельзя отнести к поразрядным?
10. Что такое маска?
11. Как, используя маску, сбросить определенный бит в ноль?
12. Напишите значение маски для того, чтобы сбросить в 16-разрядном числе 2 младших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
13. Как, используя маску, установить определенный бит в единицу?
14. Напишите значение маски для того, чтобы установить в 16-разрядном числе 2 старших бита, не изменяя все остальные. Какую логическую операцию нужно для этого использовать?
15. Как, используя логические операции, определить, делится ли число на 4? На 8?
16. В каких практических задачах можно применять установку или сброс битов двоичного кода?
17. Каковы возможности операции "исключающее ИЛИ"?
18*. Попробуйте придумать алгоритм шифрования кода с помощью операции "исключающее ИЛИ". Постарайтесь предложить простой алгоритм изменения маски, а не просто использовать константу.
19. Прочитайте еще раз материал, связанный с переполнением при сложении, описанный в 4.3.1. Какой логической операцией можно определить, совпадают или нет биты *C* и *S*? (Ответ: *C* **xor** *S*.)
20. Какую роль играет операция "НЕ" при получении отрицательных чисел?
21. Как выполнить инверсию всех битов, не используя логическую операцию "НЕ"?

22. Что такое сдвиг? Какие вы знаете виды сдвига?

23. Как обрабатываются самый старший и самый младший биты при различных типах сдвига?

24. Покажите на примерах, что сдвиг влево двоичного кода удваивает число, а сдвиг вправо — уменьшает вдвое.

25. Почему логический сдвиг не годится для уменьшения в два раза отрицательных чисел? Как работает арифметический сдвиг?

26. Почему не требуется арифметический сдвиг влево?

27*. Выведите правило вычисления результата арифметического сдвига отрицательного нечетного числа на один разряд вправо. Проверьте, применимо ли это правило к положительным нечетным числам. Как упрощается формула для четных исходных значений? (Результат указанного сдвига для некоторого числа $X$ вычисляется по формуле $\mathrm{int}(X/2)$, где функция int — это *целая часть* числа, определяемая как максимальное целое значение, не превосходящее $X$. Например, если $X = -3$, то после сдвига имеем $\mathrm{int}(-1{,}5) = -2$. Для положительных нечетных чисел формула также справедлива. Заметим, что для любых четных чисел деление дает в результате целое число, так что брать целую часть не потребуется.)

28. Где могут применяться сдвиги?

## Задачи

1. Переведите в 8-разрядный двоичный код десятичные числа 31 и 19 и сложите их. Для проверки переведите полученную сумму в десятичную систему счисления.

2. Повторите вычисления предыдущей задачи, заменив первое слагаемое на "–31". Подумайте, что изменится, если код сделать 16-разрядным.

3. Выберите произвольные значения двух целых чисел $A$ и $B$ и запишите их в виде 8-разрядных двоичных кодов. Проверьте путем непосредственных вычислений справедливость тождества $A - B = A + (-B)$.

4. Сложение ведется в 8-разрядной арифметике со знаком. Какое максимальное число можно прибавить к двоичной константе $100000_2$, чтобы не возникло переполнения? Как изменится результат, если число будет беззнаковым?

5. Переведите в двоичный код десятичные числа 12 и 7 и перемножьте их. Для проверки переведите результат в десятичную систему счисления.

6. Повторите вычисления предыдущей задачи, заменив первый сомножитель на "–12". Считайте, что числа представлены в 8-разрядном коде.

7. Братья Петя и Коля часто спорят по поводу решения задач по информатике. Главная причина состоит в том, что Петя всегда решает задачу, как показал учитель, а Коля вечно придумывает что-то свое, причем не всегда удачно. Сегодня, например, они осваивали двоичную арифметику, умножая $1000_2$ на $11011_2$. Петя добросовестно умножал столбиком, а Коля, не говоря ни слова, взял второй сомножитель и, приписав к нему три нуля, получил такой же ответ. После объяснений Петя был вынужден признать правоту брата. Как объяснил свое решение Коля? (Ответ: он просто догадался применить переместительный закон для умножения, переставив сомножители.)

8. Какое из двух беззнаковых чисел больше: 0111 0111 или 1000 1000? Изменится ли ваш ответ, если вам скажут, что исходные коды — это 8-разрядные числа со знаком? Переведите оба значения для случаев чисел со знаком и без него в десятичную систему счисления. (119 и 136, 119 и –120)

9. Код строчной латинской буквы 'a' равен $61_{16}$, а заглавной 'A' — $41_{16}$. Используя логическую операцию "И", преобразуйте код строчной буквы в заглавную. Проверьте, работает ли предложенный вами метод для других букв.

10. Используя логическую операцию "ИЛИ", преобразуйте код заглавной буквы 'A' в строчную 'a'. Проверьте, работает ли предложенный вами метод для других букв.

11. Петя и Коля решают домашнюю задачу: "Известны коды двух введенных цифр $C_1$ и $C_2$. Найти сумму этих цифр". Петя, как обычно, глядя на решение задач в классе, пишет: "1) $N_1 = C_1$ **and** $30_{16}$; 2) $N_2 = C_2$ **and** $30_{16}$; 3) $S = N_1 + N_2$". Коля предлагает более короткое решение: "1) $S = C_1 + C_2$; 2) $S = S$ **and** $0F_{16}$". Петя, ссылаясь на образцы задач в учебнике, критикует маску $0F_{16}$. Но Коля показывает на двух примерах ('2' и '3'; '5' и '7'), что его алгоритм дает правильные результаты. Что скажет учитель по поводу Колиного решения? (Ответ: что оно приводит к неправильным ответам в случаях, когда сумма цифр больше $15_{10}$.)

12. Выполните битовую операцию $X$ **and** 3 для следующих десятичных значений $X$: 4, 5, 8, 15, 16. Для каких из них получился нулевой ответ? Что общего у этих чисел? (Ответ: все они делятся нацело на 4.)

13. Разработайте аналогичные способы определения делимости на 2, 8 и 16.

14. Цвет точки в формате RGB хранится как 4-байтовое целое число, которое в шестнадцатеричном виде выглядит так: 00 **RR GG BB** (т.е. старший байт не используется, а в каждом из последующих байт хранится одна из трех цветовых компонент[14]). Напишите последовательность операций, позволяющих выделить из 32-битного числа каждую из трех цветовых компонент. Какая из них потребует большего числа операций?

15. Петя и Коля решают задачу: "Цвет точки в формате RGB хранится как 4-байтовое целое число $N$, которое в шестнадцатеричном виде выглядит так: 00 **RR GG BB**. Напишите последовательность операций, позволяющих выделить из 32-битного числа красную компоненту". Петино решение: "1) $N = N$ **and** $FF0000_{16}$; 2) логический сдвиг $N$ вправо на 16 разрядов". Колино решение содержит только вторую из этих операций. Чье решение правильное? (Ответ: правильны оба, но у Коли рациональнее.)

16. Используя только сдвиги, очистите (установите в ноль) 4 старших разряда 8-битного значения. Как с помощью сдвигов очистить 4 младших разряда?

17. Каков результат логического сдвига влево на 4 разряда шестнадцатеричного целого числа $FEDC_{16}$?

[14] Несмотря на то что старший байт кажется лишним, этот способ хранения не лишен смысла. Дело в том, что процессор не приспособлен к обработке 3-байтовых данных, тогда как с 4-байтовыми работает очень быстро. Описанный формат, в частности, применяется при хранении таблиц цветов в графическом формате BMP. В других форматах (например, в PNG) старший байт используется для хранения степени прозрачности пикселя (альфа-канала).

Сравните его с результатом циклического сдвига. (Ответы: $EDC0_{16}$ и $EDCF_{16}$.)

18. Заданы два шестнадцатеричных целых числа: $1234_{16}$ и $FEDC_{16}$. К каждому из них применяются логический, циклический и арифметический сдвиги вправо на 4 разряда (каждый раз сдвигается первоначальное значение, а не результат предыдущего сдвига!). Напишите и объясните результаты для каждой операции. (Ответы: $0123_{16}$, $4123_{16}$ и $0123_{16}$; $0FED_{16}$, $CFED_{16}$, $FFED_{16}$.)

19. Запишите число "–18" в 8-разрядном двоичном коде. Что получится, если применить к нему логический сдвиг вправо? арифметический сдвиг вправо? Сравните полученные результаты и объясните их.

20. Переведите число "–1" в дополнительный двоичный код и дважды примените к нему арифметический сдвиг вправо. Какой будет результат? (–1; объяснение этого факта вытекает из формулы, которая обсуждается в вопросе 27)

21. Выполните приведенный в тексте параграфа алгоритм умножения на 10 для $Z = 1100_2$. Для проверки переведите результат в десятичную систему счисления.

## Хранение вещественных чисел в памяти компьютера

В начале главы мы отмечали принципиальное различие между вещественными и целыми числами: целые числа дискретны, а вещественные, напротив, непрерывны, а значит, не могут быть полностью корректно перенесены в дискретную по своей природе вычислительную машину. Как же все-таки кодируются в компьютерах вещественные числа?

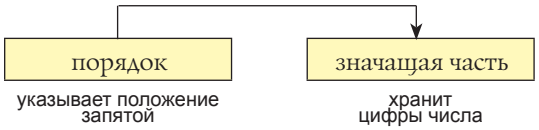
В первых ЭВМ использовалось кодирование *с фиксированной запятой*. Это значит, что положение запятой, отделяющей целую часть от дробной, было жестко закреплено в разрядной сетке конкретной ЭВМ — раз и навсегда для всех чисел и для всех технических устройств этой машины. Все вычислительные алгоритмы были заранее "настроены" на это фиксированное размещение. Но в задачах, которые решаются на компьютерах, встречаются самые разнообразные по величине числа, от размера атома до астрономических расстояний. Чтобы согласовать их с таким жестким представлением, программист, подготавливая задачу к решению на ЭВМ, выполнял большую предварительную работу по *масштабированию* данных: маленькие числа умножались на определенные коэффициенты, а большие, напротив, делились. Масштабы подбирались так, чтобы результаты всех операций, включая промежуточные, не выходили за пределы разрядной сетки и, в то же время, обеспечивалась максимально возможная точность (все разряды данных по возможности находились в пределах сетки). Эта работа требовала много времени и часто являлась источником ошибок.

Тем не менее работа с фиксированным размещением запятой не только показала недостатки метода, но и наметила путь их устранения. В самом деле, если наиболее сложным и трудоемким местом является масштабирование данных, надо его автоматизировать. Иными словами, надо научить машину самостоятельно размещать запятую так, чтобы числа при счете не выходили за разрядную сетку и по возможности сохранялись с максимальной точностью. Конечно, для этого нужно разрешить компьютеру "перемещать" запятую, а значит, дополнительно как-то сохранять в двоичном коде числа информацию о ее текущем положении. В этом и заключается главная идея представления чисел *с плавающей запятой*[15].

Удобное представление вещественных чисел не пришлось специально придумывать. В математике уже существовал подходящий способ записи, основанный на том, что любое число $A$ в системе счисления с основанием $B$ можно записать в виде

$$A = \pm Z \cdot B^P,$$

где $Z$ называют *значащей частью*, а показатель степени $P$ — *порядком* числа. Для десятичной системы это выглядит привычно, например, заряд электрона равен $-1{,}6 \cdot 10^{-19}$ кулона, а скорость света в вакууме составляет $3 \cdot 10^8$ м/сек.




Однако представление числа с плавающей запятой — не единственно. Например, число 23,4 можно записать следующими способами:

$2340 \cdot 10^{-2} = 234 \cdot 10^{-1} = 23{,}4 \cdot 10^0 = 2{,}34 \cdot 10^1 =$
$= 0{,}234 \cdot 10^2 = 0{,}0234 \cdot 10^3 = \ldots$

На первый взгляд выбор очень большой, однако большинство вариантов обладают серьезными недостатками. В частности, все представления, в которых значащая часть содержит нули непосредственно после запятой (0,0234, 0,00234 и т.п.) или перед ней (2340, 23 400 и т.п.), не подходят, поскольку, сохраняя эти незначащие нули, мы напрасно увеличиваем разрядность чисел. Согласно математической теории, для обеспечения максимальной точности при сохранении цифр числа в фиксированном количестве разрядов надо выбирать такой метод, при котором значащие цифры числа следует поместить как можно ближе к запятой. С этой точки зрения оптимальным будет вариант, когда целая часть равна нулю, а первая ненулевая цифра находится сразу после запятой (в нашем примере 0,234). При этом вместо двух частей (целой и дробной) остается только дробная, что фактически делает ненужной "разделительную" запятую.

Но взгляните на рисунок ниже, изображающий такое число на индикаторе: первый разряд всегда равен нулю, что делает его практически бесполезным. Поэтому с точки зрения экономии разрядов лучше взять другой вариант, в котором значащая часть равна 2,34 (см. второй рисунок). Именно такой выбор закреплен в стандарте IEEE 754[16], на котором основана арифметика вещественных чисел в современных компьютерах.

Итак, существует два приблизительно равноценных способа представления чисел с плавающей запятой:


- оптимальный с теоретической точки зрения, в котором целая часть нулевая, а первая цифра дробной части ненулевая ($0{,}234 \cdot 10^2$);

[15] В англоязычных странах используется термин *floating point* — плавающая точка, поскольку в этих странах целая часть от дробной отделяется не запятой, как у нас, а точкой.

[16] Последняя версия стандарта называется IEEE 754-2008.

• более удобный с практической точки зрения, в котором целая часть состоит из единственной ненулевой цифры ($2{,}34 \cdot 10^1$).

К сожалению, эта "двойственность" порождает некоторую путаницу. В теоретической литературе, как правило, используется первый способ[17]. Все описания конкретных компьютерных систем, напротив, базируются на втором. Причем в обоих случаях обычно используется один и тот же русский термин — *мантисса*. Зато в англоязычной компьютерной литературе приняты два разных термина: в первом случае значащая часть называется *mantissa* (слово *мантисса* для математиков однозначно связано с дробной частью числа), а во втором — *significand* (значащая часть).

Далее мы будем использовать второй вариант, поскольку именно он дает возможность решать задачи, связанные с практическим кодированием вещественных чисел. В связи с этим мы будем использовать термин "значащая часть", а не "мантисса".

В компьютере используется такое представление вещественных чисел с плавающей запятой, при котором значащая часть $Z$ удовлетворяет условию $1 \le Z < B$, где $B$ — основание системы счисления. Такое представление называется *нормализованным*.

Нормализованное представление числа единственно — в нашем примере это $2{,}34 \cdot 10^1$. Любое число может быть легко нормализовано. Единственное, но важное исключение из правила составляет ноль — для него невозможно получить $Z \ge 1$. Ради такого важного случая было введено дополнительное соглашение: число 0, в котором все биты нулевые, в качестве исключения считается нормализованным.

Все сказанное выше можно применить и к двоичной системе:

$$A = \pm Z \cdot 2^P, \text{ причем } 1 \le Z < 2.$$

Например: $-7_{10} = -111 \cdot 2^0 = -1{,}11 \cdot 2^{10}$ (не забывайте, что значащая часть и порядок записаны в двоичной системе!); отсюда $Z = 1{,}11$ и $P = 10$. Двоичная значащая часть **всегда** (исключая, разумеется, ноль!) начинается с единицы, т.к. $1 \le Z < 2$. Поэтому во многих компьютерах (в том числе в компьютерах на базе процессоров Intel) эта так называемая *скрытая единица* не хранится в ОЗУ, что позволяет сэкономить еще один дополнительный разряд значащей части[18].

Идея "скрытой единицы" раньше действительно давала заметное увеличение точности представления чисел, поскольку количество разрядов в устройствах ЭВМ того времени было невелико и поэтому усложнение метода кодирования было оправданно. Сейчас, когда процессоры работают с 64-битными данными, это скорее дань традиции, чем практически полезная мера[19].

Таким образом, при кодировании вещественного числа с плавающей запятой фактически хранится две величины: его значащая часть (*significand*) и порядок. От разрядности значащей части зависит точность вычислений, а от разрядности порядка — диапазон представления чисел. В таблице приведены характеристики стандартных вещественных типов данных, используемых в математическом сопроцессоре Intel:

| Тип | Диапазон | Число десятичных значащих цифр | Размер (байт) |
|---|---|---|---|
| *single* | $1{,}4 \cdot 10^{-45} - 3{,}4 \cdot 10^{38}$ | 7–8 | 4 |
| *double* | $4{,}9 \cdot 10^{-324} - 1{,}8 \cdot 10^{308}$ | 15–16 | 8 |
| *extended* | $3{,}6 \cdot 10^{-4951} - 1{,}2 \cdot 10^{4932}$ | 18–19 | 10 |

Рассмотрим, как "распланированы" 4 байта, отводимые под простейший тип *single*. Тип *double* устроен совершенно аналогично, а тип *extended*, который является основным форматом для вычислений в сопроцессоре, отличается только тем, что в нем единица в целой части не "скрывается".

В числах типа *single* 23 младших бита (с номерами от 0 до 22) хранят значащую часть числа, следующие 8 (с 23-го по 30-й) — порядок, а старший (31-й) бит отведен под знак числа.

Правила двоичного кодирования вещественных чисел во многом отличаются от правил кодирования целых чисел. Для того чтобы в них разобраться, рассмотрим конкретный пример — закодируем число "–17,25" в формате *single*. Прежде всего переведем модуль числа в двоичную систему, отдельно целую и дробную части (см. главу 2):

$$17{,}25 = 10001{,}01_2$$

Для нормализации нужно передвинуть запятую на $4_{10} = 100_2$ разряда влево:

$$10001{,}01 \cdot 2^0 = 1{,}000101 \cdot 2^{100}$$

Построим значащую часть, "скрыв" единицу в целой части:

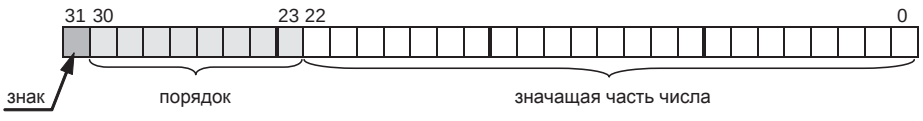
$$M = Z - 1 = 0{,}\mathbf{0001010...0}$$

Так как число отрицательное, знаковый разряд нужно установить в 1, т.е. $S = 1$.

В отличие от целых чисел значащая часть вещественных чисел хранится в прямом коде.

Таким образом, значащие части положительного и равного по модулю отрицательного числа одинаковы, а отличаются они только старшим (знаковым) битом.

Теперь остается закодировать двоичный порядок 100. Порядок — это целое число со знаком, для него используется *кодирование со смещением*: чтобы




[17] К этой группе относится большинство отечественных книг по основам вычислительной техники.

[18] В результате то, что *осталось* после "скрытия" единичной целой части, можно вполне обоснованно называть мантиссой.

[19] Оценим величину добавки для математического сопроцессора Intel. Значащая часть чисел двойной точности вместо "скрытой единицы" приобретает дополнительный 53-й(!) бит, что прибавляет к значению числа поправку $2^{-53} \approx 1{,}1*10^{-16}$, влияющую на 16-й десятичный знак; согласно IEEE 754-2008 в 128-битных числах эта поправка будет и того меньше: $2^{-113} \approx 9{,}6*10^{-35}$.

вообще избавиться от знака порядка, к нему добавляют некоторое положительное смещение $d$:

$$P_d = P + d.$$

Величина смещения подбирается так, чтобы число $P_d$ было всегда положительным. В этом случае оказывается легче сконструировать сопроцессор для обработки вещественных чисел.

Для кодирования порядка в числах типа *single* используют смещение $d = 127_{10} = 7F_{16}$. Таким образом, для нашего примера

$$P_d = 100 + 111\ 1111 = \mathbf{1000\ 0011}$$

Собирая теперь $S$, $P_d$ и $M$ в единое 32-разрядное число, получаем:

1 **10000011** 00010100000000000000000,

или более компактно в шестнадцатеричной системе — **C1 8A 00 00**. Этот код и будет записан в память[20].

Не все двоичные комбинации для вещественных чисел соответствуют "правильным" числам: некоторые из них кодируют бесконечные значения, а некоторые — нечисловые данные (англ. *NaN = not a number*, "не число"). Они отличаются от остальных чисел тем, что имеют максимально возможный порядок[21] (например, для типа *single* это смещенный порядок $P_d = 255$, а для типа для *double* — 2047). Подобные "неправильные" данные возникают только в результате ошибок в вычислениях.

Если сравнить между собой представление целых и вещественных чисел, то станет видно, что они сильно отличаются друг от друга. Поэтому не удивительно, что свойства значений, скажем, "3" и "3,0" в компьютерной арифметике совершенно разные.

## Контрольные вопросы

1. Чем вызваны трудности, возникающие при представлении вещественных чисел в компьютере? Как они связаны с непрерывностью вещественных чисел в математике?
2. Объясните, как хранятся вещественные числа с фиксированной запятой. Почему этот метод не используется в современных компьютерах?
3. Что такое плавающая запятая? Из каких частей состоит число при кодировании с плавающей запятой?
4. Приведите примеры физических величин, которые обычно записывают в форме с плавающей запятой.
5. Почему метод представления чисел с плавающей запятой неоднозначен? Как изменится порядок, если запятую сместить на один разряд влево (вправо)?
6. Что такое нормализованная форма записи числа?
7. Как требования нормализации связаны с точностью представления вещественных чисел?
8. Единственно ли нормализованное представление числа? Все ли числа имеют нормализованное представление?
9. Почему старший бит значащей части нормализованного двоичного числа всегда равен единице? Как этот факт используется на практике?
10. Какие числа сохраняются в памяти с нулевой значащей частью? (С учетом эффекта "скрытой единицы" нулевую значащую часть имеет любое число, в точности равное степени двойки, а также, в виде исключения, машинный ноль.)
11. На что влияют разрядность значащей части и разрядность порядка?
12. Почему задание разрядности для целых чисел однозначно определяет их свойства, а для вещественных — нет? (Ответ: для вещественных чисел дополнительно требуется задать распределение битов между значащей частью и порядком; только тогда можно сделать выводы о диапазоне и точности представления чисел.)
13. Что вы знаете о типах *single*, *double* и *extended*?
14. Как хранится порядок во всех рассмотренных форматах вещественных чисел? Почему не хранится знак порядка?
15. Сравните методы хранения отрицательных целых и вещественных чисел.
16. Как по двоичному представлению вещественного числа определить, положительно оно или отрицательно? Подходит ли этот метод для целых чисел?
17. В каком из вещественных форматов не используется "скрытая единица" и почему? (Ответ: в формате *extended*, поскольку этот формат предназначен для проведения вычислений, а типы *single* и *double* — только для хранения чисел в памяти.)
18. Какие логические операции и с какой маской надо применить, чтобы в переменной типа *single*: а) выделить значащую часть, сбросив порядок и знаковый бит; б) восстановить в полученной знаковой части "скрытую единицу"?
19. Как можно выделить смещенный порядок из числа типа *single*? Как получить истинное значение порядка? (Ответ: сдвиг на 23 разряда вправо и логическое "И" для сброса знакового бита; для вычисления несмещенного порядка из полученного значения надо вычесть 127.)
20. С помощью какой маски можно выделить знаковый бит числа, хранящегося в формате *single*?
21. Что такое NaN?
22. Чем отличается представление в памяти целого числа и равного ему вещественного с нулевой дробной частью (например, 12 и "12,0")?

## Задачи

1. Рассмотрим вымышленный 32-разрядный компьютер, в котором вещественные числа кодируются с фиксированной запятой, причем к целой части относится один байт, а к дробной — три. Рассчитайте для такой машины максимальное и минимальное допустимые числа и сравните с аналогичными значениями для типа *single*; объясните разницу. Вычислите также "порог" антипереполнения, т.е. минимальное число, отличное от нуля.
2. Запишите в нормализованном виде следующие десятичные вещественные числа: $43 \cdot 10^{21}$; 1040; 1,5; 0,32; 0,0005; $0{,}34 \cdot 10^{-12}$.
3. Запишите в нормализованном виде следующие двоичные вещественные числа (значащая часть и порядок даны в двоичной системе счисления): $11 \cdot 2^{10100}$; 10110; 1,1; 0,101; 0,0001; $11{,}001 \cdot 2^{-1000}$. Обратите внимание на значение первого бита значащей части.
4. Сравните диапазон чисел, который представляется в 32-битной форме *single*, с диапазоном целых 32-разрядных чисел со знаком.

[20] На самом деле в IBM-совместимых персональных компьютерах байты будут сохранены в память в обратном порядке: 00 00 8A C1.

[21] Это вполне логично, поскольку для вещественных чисел переполнение (получение "бесконечного" значения) наступает именно при больших порядках.

5. Рассчитайте величину порядка со смещением для типа *single*, если числа имеют двоичные порядки $-11_2$, 0 и $11_2$. (Ответ: $7B_{16}$, $7E_{16}$ и $81_{16}$.)

6. Определите, как хранятся в памяти в формате single следующие вещественные десятичные числа: 1; 100; 0,1. Ответ запишите в шестнадцатеричной системе счисления. (Ответ: 3F 80 00 $00_{16}$, 42 C8 00 $00_{16}$, 3D CC CC $CD_{16}$ — перестановка байт в памяти для компьютеров IBM PC здесь не учитывается.)

7. Используя результаты предыдущей задачи, получите соответствующие коды для вещественных чисел "–1", "–100" и "–0,1".

8. Определите, какому десятичному значению соответствуют коды (тип *single*): 3F C0 00 $00_{16}$, BF C0 00 $00_{16}$, 3F4 40 00 $00_{16}$. (Ответ: 1,5; –1,5; 0,75.)

9*. Оцените максимальное количество десятичных знаков в числе *single*. (*Возможное решение*. Точность вычислений зависит от разрядности значащей части; с учетом "скрытой единицы" для *single* она равна 24 бита. Максимальное число с нулевой дробной частью, которое "поместится" в нее, равно $2^{24} - 1 = 16\,777\,215$. В этом числе 8 десятичных знаков, что хорошо согласуется с приведенным в таблице значением.)

10*. Как преобразовать некоторое небольшое положительное вещественное число с нулевой дробной частью, например $9_{10} = 1{,}001 \cdot 2^{11}$, в форму беззнакового целого? (Ответ: сбросить разряды, содержащие знак и порядок, с помощью операции логического "И", восстановить "скрытую единицу" с помощью "ИЛИ" и сдвинуть оставшийся код значащей части на $N_Z - P$ разрядов вправо, где $N_Z$ — число разрядов в значащей части, а $P$ — порядок данного числа. В большом числе могло бы оказаться, что $P > N_Z$, тогда пришлось бы сдвигать уже влево!)

## Операции с вещественными числами

### Сложение и вычитание

Рассмотрим принципы вещественной компьютерной арифметики на простых примерах. Сложим $7{,}25_{10} = 111{,}01$ и $1{,}75_{10} = 1{,}11$ (здесь и далее будем явно указывать систему счисления только для десятичных чисел). Представим эти числа в нормализованном виде: $111{,}01 \cdot 2^0 = 1{,}1101 \cdot 2^{10}$ и $1{,}11 \cdot 2^0$ (еще раз подчеркнем, что значащие части и порядки чисел указываются в двоичной системе!). Не будем сейчас использовать "скрытую" единицу: это нужно только при сохранении чисел в памяти, а при изучении арифметических действий удобнее иметь "развернутые" значения.

Числа, записанные в форме с плавающей запятой, нельзя непосредственно сложить. Дело в том, что когда числа имеют разный порядок, их значащие части оказываются сдвинутыми друг относительно друга. Поэтому первое, что делает процессор перед сложением вещественных чисел, — *выравнивает их порядки до большего*. Число, имеющее меньший порядок $p_2$ (и значащую часть $z_2$), "подгоняется" к числу с большим порядком $p_1$ следующим образом.

1. Если $p_2 = p_1$, то порядки выровнены и преобразования закончены.
2. $p_2 = p_2 + 1$.
3. Сдвинуть значащую часть $z_2$ на один разряд вправо.
4. Перейти к пункту 1.

Для нашего примера разность порядков равна $10 - 0 = 10 = 2_{10}$, так что для выравнивания порядков значащую часть придется сдвинуть дважды (порядок при этом увеличится на 2): $1{,}11 \cdot 2^0 = 0{,}0111 \cdot 2^{10}$. Подчеркнем, что ради проведения сложения нормализацию пришлось временно нарушить.

Теперь числа имеют одинаковый порядок и их значащие части можно складывать:

```
   1,1101
+  0,0111
---------
  10,0100
```

Полный результат (с учетом порядка) равен $10{,}01 \cdot 2^{10}$ (убедитесь, что получившееся число равно $9_{10}$). Но значащая часть результата больше 2, поэтому для записи числа в память его необходимо нормализовать: $10{,}01 \cdot 2^{10} = 1{,}001 \cdot 2^{11}$.

В этом примере мы нигде не учитывали ограниченность разрядной сетки, и для простоты специально взяли короткие числа. Как же обстоит дело в реальных вычислениях? При выравнивании порядков происходит сдвиг значащей части меньшего из чисел вправо, при этом ее младшие (правые) разряды могут выйти за пределы разрядной сетки и будут отброшены. При сложении чисел с большой разностью порядков в результате таких сдвигов меньшее число может стать равно нулю. Например, представьте себе, что при 24-битной значащей части у слагаемых $A$ и $B$ разность порядков составляет, например, $26_{10}$. В этом случае при выравнивании порядков произойдет 26 сдвигов значащей части вправо, так что абсолютно все(!) ее разряды исчезнут. В результате сложения окажется, что $A + B = A$, хотя $B \neq 0$ — это очередной (но далеко не единственный) пример погрешности компьютерных вычислений.

## Умножение и деление

Числа, представленные в форме с плавающей запятой, "хорошо приспособлены" для выполнения умножения и деления. При перемножении достаточно (в полном соответствии с правилами математики) перемножить их значащие части, а порядки сложить. При делении значащие части делятся, а порядки вычитаются. Конечно, результат может оказаться ненормализованным, но это легко устраняется стандартной процедурой.

Рассмотрим, как выполняется умножение чисел $1{,}25_{10} = 1{,}01_2$ и $4{,}0_{10} = 100{,}0_2$. В нормализованном виде они запишутся как $1{,}01 \cdot 2^0$ и $100 \cdot 2^0 = 1{,}0 \cdot 2^{10}$. В этом примере значащие части можно перемножить устно: $1{,}01 \cdot 1{,}0 = 1{,}01$. Теперь сложим порядки: $0 + 10 = 10$. Таким образом, результат равен $1{,}01 \cdot 2^{10}$. Он уже удовлетворяет требованиям нормализации, поэтому никаких дополнительных действий не требуется. Легко показать, что $1{,}01 \cdot 2^{10} = 101 \cdot 2^0 = 5_{10}$.

Знакомство с вещественной арифметикой убедительно показывает важную роль битовых операций, изучен-

ных в 4.3. Для нормализации постоянно используются сдвиги; для выделения значащей части или порядка из общего кода числа обязательно потребуется логическая операция "И", а для получения единого кода из порядка и значащей части можно использовать логическое "ИЛИ". В обеих последних задачах также необходимы сдвиги.

## Контрольные вопросы

1. Почему перед сложением или вычитанием вещественных чисел требуется выравнивать порядки?

2. Какое число — большее или меньшее — подвергается сдвигу при выравнивании порядков? Почему? (Ответ: большее число придется сдвигать влево, в результате старшие разряды могут выйти за разрядную сетку, что недопустимо; у меньшего числа будут теряться правые, т.е. младшие разряды, что плохо, но допустимо.)

3. Верно ли, что при выравнивании порядков значащая часть всегда сдвигается вправо?

4. Как вычислить количество сдвигов, которое потребуется произвести для выравнивания порядков? (Ответ: оно равно разности порядков слагаемых.)

5. Может ли оказаться так, что при выполнении операции сложения значащие части придется не складывать, а вычитать? (Ответ: да, все зависит от комбинации знаков слагаемых.)

6. Как изменится двоичный код вещественного числа, если это число умножить на 2? Сравните с тем, как меняется код целого числа при удвоении.

7. Почему в компьютерной арифметике возможны случаи, когда $A + B = A$ при $B \neq 0$? При каких условиях может так получиться?

8*. Может ли при вычитании быть переполнение? Антипереполнение?

9. Сформулируйте правила умножения и деления вещественных чисел.

10*. Верно ли, что когда для размещения результата умножения в значащей части не хватает разрядов — это переполнение? Когда возникает переполнение? (Ответ: нет, при нехватке разрядов в значащей части отбрасываются младшие биты, что несколько снижает точность, но к переполнению не приводит; переполнение происходит, когда не хватает знаков для размещения порядка.)

11. Может ли в результате арифметической операции нарушиться нормализация? Как в таких случаях нужно поступать?

## Задачи

1. Суммируются два двоичных числа: $1{,}0 \cdot 2^{100}$ и $1{,}11 \cdot 2^{11}$. Какое из них будет сдвигаться при выравнивании порядков? На сколько разрядов?

2. Выполните сложение двух десятичных чисел 2,5 и 0,125, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Результат представьте в формат *single* и запишите в шестнадцатеричном коде. (Ответ: число 2,625 в формате *single* запишется как $40\ 28\ 00\ 00_{16}$.)

3. Выполните вычитание десятичных чисел 0,125 – 2,5, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Вычтите из большего числа (2,5) меньшее (0,125), а знак добавьте в конце. Результат представьте в формат *single* и запишите в шестнадцатеричном коде. (Ответ: число –2,375 в формате *single*: $C0\ 18\ 00\ 00_{16}$.)

4. *Выполните сложение двух десятичных чисел 0,1 и 0,2, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Предположите, что на значащую часть выделяется 8 разрядов, "скрытую единицу" не используйте. (*Решение*. $0{,}1_{10} = 0{,}0(0011)_2$, т.е. в заданной разрядности с округлением последнего бита $1{,}1001101 \cdot 2^{-100}$; 0,2 вдвое больше, поэтому достаточно у предыдущего числа к порядку прибавить единицу: $1{,}1001101 \cdot 2^{-11}$. С учетом округления сумма будет равна $1{,}0011010 \cdot 2^{-10}$.)

5. Докажите, что если двоичное число $1{,}01 \cdot 2^{10}$ сложить с самим собой, то его значащая часть не изменится, а порядок возрастет на 1.

6*. Заданы 5 вещественных чисел, причем одно из них $A = 1{,}0 \cdot 2^{111}$, а все остальные равны между собой: $B = C = D = E = 1{,}0 \cdot 2^{-10}$ (значения значащих частей и порядков записаны в двоичной системе счисления). Убедитесь, что в случае, когда значащая часть представлена 8 битами ("скрытая единица" не используется), результат сложения всех чисел оказывается зависящим от порядка сложения, в частности, $A + B + C + D + E \neq E + D + C + B + A$. Объясните результат. (*Решение*. Разница порядков у $A$ с любым из оставшихся чисел равна

$$111 - (-10) = 1001_2 = 9_{10}.$$

Это означает, что при выравнивании порядков значащая часть чисел $B$, $C$, $D$ или $E$ заведомо "покинет" 8-разрядную сетку. Следовательно, $A + B + C + D + E = A$. Теперь просуммируем в обратном порядке. Сумму четырех одинаковых слагаемых легко найти, если вспомнить, что каждое умножение на 2 — это увеличение порядка на 1. В итоге $E + D + C + B = 4 \cdot B = 1{,}0 \cdot 2^{-10+10} = 1{,}0 \cdot 2^0$. Поскольку теперь разность порядков уже меньше разрядности значащей части, то получим другой ответ: $1{,}0000001 \cdot 2^{111}$. Эффект можно объяснить так. Каждое из чисел $B$, $C$, $D$ или $E$ мало по сравнению с $A$ и поэтому при сложении с ним просто теряется. Тем не менее, если все их предварительно сложить, то сумма уже попадет на разрядную сетку и даст небольшую добавку к значению $A$.)

7. Перемножьте два десятичных числа 0,75 и 1,25, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Для проверки переведите результат в десятичную систему счисления. (Ответ: $1{,}111 \cdot 2^{-1}$.)

8. Двоичное число $1{,}1 \cdot 2^{1000}$ записано в формате, в котором под порядок вещественного числа отводится 4 бита. Произойдет ли переполнение, если его возвести в квадрат?

9. Выполните деление десятичных чисел 0,75 и 0,25, предварительно преобразовав их к двоичной форме и выделив значащую часть и порядок. Для проверки переведите результат в десятичную систему счисления. (Ответ: $1{,}1 \cdot 2^1$.)

10. Двоичное число $1{,}0 \cdot 2^{-1000}$ записано в формате, в котором под порядок вещественного числа отводится 4 бита. Что произойдет, если его поделить на $1{,}0 \cdot 2^{1001}$?

**В МИР ИНФОРМАТИКИ // № 155**

**Газета для пытливых учеников и их талантливых учителей**
vmi@1september.ru

## ШКОЛА ПРОГРАММИРОВАНИЯ

# Обмены, обмены...

**Продолжение. Начало см. “В мир информатики” № 152 (“Информатика” № 22/2010)**

Мы продолжаем рассматривать особенности программ, связанных с обменом значениями величин, элементов массива и т.п.

### 5. Задача “Обмен местами двух элементов двумерного массива с заданными номерами”

Задача может быть решена двумя способами:

1) с использованием вспомогательной переменной:

```
вспомогательная := a[m1, m2]
a[m1, m2] := a[m3, m4]
a[m3, m4] := вспомогательная
```

2) без использования вспомогательной переменной (см. задачу 1 в первой части статьи):

```
a[m1, m2] := a[m1, m2] + a[m3, m4]
a[m3, m4] := a[m1, m2] – a[m3, m4]
a[m1, m2] := a[m1, m2] – a[m3, m4]
```

— где `m1` и `m2` — индексы первого из обмениваемых элементов, `m3` и `m4` — второго.

### 6. Задача “Обмен местами двух строк двумерного массива с заданными номерами”

Эта задача также может быть решена двумя способами:

1) с использованием вспомогательного одномерного массива `вспомогательный_массив` для временного хранения одной из обмениваемых строк;

2) с использованием вспомогательной величины `вспомогательная` для временного хранения значения отдельного элемента массива.

Если номера обмениваемых строк — `s1` и `s2`, то в первом случае фрагмент программы выглядит так:

```
|Запоминаем все элементы строки с номером s1
|в массиве вспомогательный_массив
нц для номер_столбца от 1 до m
▪ вспомогательный_массив[номер_столбца] :=
  a[s1, номер_столбца]
кц
|Переписываем все элементы строки с номером s2
|в соответствующие элементы строки
|с номером s1
нц для номер_столбца от 1 до m
▪ a[s1, номер_столбца] := a[s2, номер_столбца]
кц
|Заполняем строку с номером s2
|"старыми" элементами строки с номером s1
нц для номер_столбца от 1 до m
▪ a[s2, номер_столбца] :=
      вспомогательный_массив[номер_столбца]
кц
```

а во втором:

```
|Каждый элемент строки с номером s1
|меняем местами с соответствующим элементом
|строки с номером s2
нц для номер_столбца от 1 до m
▪ вспомогательная := a[s1, номер_столбца]
▪ a[s1, номер_столбца] := a[s2, номер_столбца]
▪ a[s2, номер_столбца] := вспомогательная
кц
```

Самостоятельно разработайте вариант программы, в котором величина `вспомогательная` не используется.

### 7. Задача “В заданном двумерном массиве из *n* строк и *m* столбцов получить его «зеркальное относительно строк отображение», т.е. первую строку поменять местами с последней, вторую — с предпоследней и т.д.”

Схема фрагмента программы, решающего данную задачу, такая:

```
нц для i от 1 до div(n, 2)
▪ Меняем местами i-ю и (n – i + 1)-ю строки
кц
```

Обмен же пар строк можно провести, как и в задаче 6, двумя способами:

1) с использованием вспомогательного одномерного массива `вспомогательный_массив` для временного хранения одной из обмениваемых строк;

2) с использованием вспомогательной величины `вспомогательная` для временного хранения значения отдельного элемента массива.

### 8. Задача "В заданном двумерном массиве из *n* строк и *m* столбцов поменять местами его верхнюю и нижнюю половины (последовательность строк в каждой половине измениться не должна)"

Как в задаче 4, целесообразно использовать величину разности между номерами обмениваемых строк:

```
разность = div(n + 1, 2)
```

Тогда схема фрагмента программы, решающего эту задачу, будет иметь вид:

```
нц для i от 1 до div(n, 2)
▪ Меняем местами i-ю и (i + разность)-ю строки
кц
```

### Задания для самостоятельной работы

1. В двумерном массиве хранится информация о зарплате каждого из 12 сотрудников отдела за 6 месяцев (первого сотрудника — в первом столбце, второго — во втором и т.д.). В связи с изменением фамилий сотрудниц необходимо поменять местами данные в 4-м и 10-м столбцах. Разработайте программу, решающую эту задачу.

2. Составьте программу решения задачи: "Из заданного двумерного массива из *n* строк и *m* столбцов получить его «зеркальное относительно столбцов отображение»".

3. Разработайте программу решения задачи: "В заданном двумерном массиве из *n* строк и *m* столбцов поменять местами его правую и левую половины (последовательность строк в каждой половине измениться не должна)".

4. Даны два двумерных массива одинаковых размеров, заполненные данными одного и того же типа. Составьте программу, с помощью которой можно обменять значениями эти два массива.

## ЗАДАЧНИК

## Ответы, решения, разъяснения

**к заданиям, опубликованным в газете "В мир информатики" № 148 ("Информатика" № 18/2010)**

### 1. Числовой ребус "ИКС в квадрате"

Напомним, что необходимо было решить числовой ребус:

| | | | | | |
|---|---|---|---|---|---|
| | | × | И | К | С |
| | | | И | К | С |
| | | | Р | И | С |
| | Д | А | Р | | |
| И | К | С | | | |
| * | * | * | * | * | |

*Решение*

Видно, что **И** = 1.

Есть четыре цифры, которые, будучи умноженными сами на себя, дают результат, оканчивающийся на эту же цифру, — 0, 1, 5 и 6. Цифра 1 уже использована. Значение **С** = 0 не подходит, так как при нем все три частных произведения оканчиваются на одну и ту же цифру 0.

Исследуем значения **С** = 5 и **С** = 6, проанализировав произведение числа 1**К**С на **С**.

При **С** = 5 имеем:

| | | | | |
|---|---|---|---|---|
| | × | 1 | К | 5 |
| | | И | К | 5 |
| | | Р | 1 | 5 |

Так как из крайнего справа разряда в разряд десятков "в уме" переходит 2, а произведение **К** на 5 оканчивается на 0 (при четном **К**) или на 5 (при нечетном **К**), то 1 в произведении не может быть получено. Значит, **С** = 6:

| | | | | |
|---|---|---|---|---|
| | × | 1 | К | 6 |
| | | И | К | 6 |
| | | Р | 1 | 6 |

Так как из крайнего справа разряда в разряд десятков "в уме" переходит 3, то произведение **К** на 6 должно оканчиваться на 8. Это может быть при **К** = 3 или при **К** = 8. Второй вариант не подходит — при нем произведение 186 на 6 — четырехзначное число. А при **ИКС** = 136 имеем: 136 × 136 = 18 496.

Итак, **ИКС** = 136.

*Правильные ответы прислали:*

— Акбарова Снежанна, Габдуллин Руслан, Гатина Гузель, Зайнакаев Вадим, Коробейникова Полина, Матиенко Анна и Швецова Анна, Республика Башкортостан, Краснокамский р-н, село Николо-Берёзовка, школа № 1, учитель **Ситдикова А.Г.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Арчеков Роман и Орозбаев Алексей, г. Волгоград, лицей № 9, учитель **Широкова Л.В.**;

— Биякова Екатерина, Биякова Ирина и Курманова Алина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Богатырев Максим, Васьков Алексей, Горбачева Дарья, Добрынина Людмила, Елисеева Кристина, Ермолаев Александр, Рожкова Вера, Романычев Павел, Табакова Кристина, Фёклина Юлия, Фуфыгин Алексей, Чапаев Иван и Чукарёва Юлия, средняя школа села Кипцы, Саратовская обл., Екатериновский р-н, учитель **Омельченко С.Ю.**;

— Васкан Александра, Желиазко Николь и Рачков Денис, Москва, гимназия № 1540, учитель **Пряхина Л.С.**;

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Грибанов Владлен, Дукач Светлана, Дюбарова Анастасия, Клименко Надежда, Романова Надежда и Сагитова Зильда, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Кальянова Анастасия, Крайнева Александра и Тимофеева Юлия, г. Стерлитамак, Республика Башкортостан, школа № 1, учитель **Орлова Е.В.**;

— Каюмов Ансель, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Мещерякова Анастасия и Никонова Валентина, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

— Мустаева Айсылу и Хисматуллин Ильмир, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**;

— Николаев Владимир, средняя школа села Янтиково, Чувашская Республика, учитель **Николаева В.П.**;

— Николаев Даниил, Республика Башкортостан, г. Стерлитамак, гимназия № 5, учитель **Пучкина С.А.**;

— Порфирьева Анастасия, средняя школа села Янтиково, Чувашская Республика, учитель **Порфирьева И.В.**

## 2. Задача “Цепочки символов”

Напомним, что необходимо было определить, какая цифра стоит в десятой строке на 1013-м месте, если строки создаются по следующему правилу:

1) первая строка состоит из одного символа — цифры “1”;

2) каждая из последующих цепочек символов создается такими действиями: в очередную строку дважды записывается цепочка цифр из предыдущей строки (одна за другой, подряд), а в конец приписывается еще одно число — номер строки по порядку (на $i$-м шаге дописывается число “$i$”).

*Решение*

Проанализировав правило построения строк или первые несколько строк, строящихся по описанному правилу:

(1) 1
(2) 112
(3) 1121123
(4) 112112311211234

— можно установить, что:

1) количество символов в $i$-й строке равно $2^i - 1$;

2) последние $i$ символов-цифр — это цифры 1, 2, ..., $i$, перед которыми стоит 1.

Но это справедливо только для однозначных $i$. Так как 10 — двузначное число, то количество символов в 10-й строке будет равно $2^{10} - 1 + 1 = 1024$. Перед цифрами числа 10 будут стоять цифры 1, 2, ..., 9. Их — 9, и они будут занимать места с 1014-го по 1022-е. Значит, на 1013-м месте будет стоять цифра 1.

*Ответы представили:*

— Акбарова Снежанна, Габдуллин Руслан и Мулюков Руслан, Республика Башкортостан, Краснокамский р-н, село Николо-Берёзовка, школа № 1, учитель **Ситдикова А.Г.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Бариев Айрат, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Биякова Ирина и Курманова Алина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Грибанов Владлен, Дукач Светлана, Дюбарова Анастасия, Клименко Надежда, Романова Надежда и Сагитова Зильда, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Мурзин Алмаз, Республика Татарстан, Актанышский р-н, село Актаныш, средняя школа № 2, учитель **Гилязова Г.М.**;

— Потопяк Наталья, Амурская обл., г. Благовещенск, школа № 11, учитель **Астахова Н.В.**;

— Умутбаев Вячеслав, Свердловская обл., Красноуфимский р-н, Тавринская средняя школа, учитель **Ярцев В.А.**;

— Хотеев Сергей, Москва, гимназия № 1530, учитель **Шамшев М.В.**;

— Шишкина Анастасия, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**

## 3. Задача “Сколько стоило мороженое?”

Напомним, что необходимо было определить “старинную” стоимость мороженого, если когда-то Павлику не хватало на мороженое 19-ти копеек, а Вове — одной, а когда они сложили свои деньги вместе, то на покупку все равно не хватило.

*Решение*

Обозначим сумму денег, имевшихся у Павлика, — $П$, у Вовы — $В$, а стоимость мороженого — $С$.

Тогда можем записать, что:

1) $С = П + 19$; (1)

2) $С = В + 1$. (2)

Так как $П + В$ меньше $С$, то, подставив вместо $С$ сумму из (1), получим:

$П + В < П + 19$, откуда $В < 19$. Но это нам ничего не дает ☹.

А если использовать вместо $С$ сумму из (2), то:

$П + В < В + 1$, или $П < 1$, то есть $П = 0$.

Иными словами, у Павлика денег совсем не было. А поскольку ему не хватало на мороженое 19 копеек, то это значит, что стоимость этого “деликатеса” и была равна 19 копеек.

Можно также решить задачу методом рассуждений.

Прежде всего ясно, что мороженое не могло стоить менее 19 копеек (у Павлика не могла быть отрицательная сумма денег).

Допустим, что стоимость мороженого — 20 копеек. В этом случае у Павлика была 1 копейка, а у Вовы — 19. Но тогда общая сумма была бы достаточной для покупки. Аналогично и при стоимости в 21 копейку и более. Значит, истинная стоимость — 19 копеек. (Вот бы сейчас так!☺)

*Правильные ответы представили:*

— Акбарова Снежанна, Габдуллин Руслан, Зайнакаев Вадим, Коробейникова Полина, Матиенко Анна, Швецова Анна и Уразова Вероника, Республика Башкортостан, Краснокамский р-н, школа № 1, село Николо-Берёзовка, учитель **Ситдикова А.Г.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Биякова Екатерина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Боголюбова Анна и Василевская Вероника, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Булатов Рустам, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Грибанов Владлен, Дукач Светлана, Дюбарова Анастасия, Клименко Надежда, Романова Надежда и Сагитова Зильда, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дудин Сергей и Щепотинник Даниил, Москва, гимназия № 1530, учитель **Козырева О.В.**;

— Есипова Анна, Лобов Андрей и Федоров Максим, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Курников Вадим, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Моисеева Оксана, Республика Татарстан, Актанышский р-н, село Актаныш, средняя школа № 2, учитель **Гилязова Г.М.**;

— Ященко Алексей, г. Ковров Владимирской обл., школа № 43, учитель **Федосеев С.А.**

## 4. Задание “В некий день”

Напомним, что необходимо было ответить на вопрос: “Суть какого приема, который используется в программировании, иллюстрирует следующее стихотворение?”:

В некий день
один поэт с мозгами набекрень
поэму сел писать,
начавши:
В некий день
один поэт с мозгами набекрень
поэму сел писать,
начавши:
В некий день
один поэт с мозгами набекрень
поэму сел писать,
начавши:
...

*Ответы прислали:*

— Акбарова Снежанна, Габдуллин Руслан и Мулюков Руслан, Республика Башкортостан, Краснокамский р-н, село Николо-Берёзовка, школа № 1, учитель **Ситдикова А.Г.**;

— Аксенов Василий, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Арчеков Роман и Орозбаев Алексей, г. Волгоград, лицей № 9, учитель **Широкова Л.В.**;

— Биякова Екатерина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Васкан Александра, Желиазко Николь и Рачков Денис, Москва, гимназия № 1540, учитель **Пряхина Л.С.**;

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Грибанов Владлен, Дукач Светлана, Дюбарова Анастасия, Клименко Надежда, Романова Надежда и Сагитова Зильда, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Деминцев Борис, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Крайнева Александра, г. Стерлитамак, Республика Башкортостан, школа № 1, учитель **Орлова Е.В.**;

— Мурзин Алмаз, Республика Татарстан, Актанышский р-н, село Актаныш, средняя школа № 2, учитель **Гилязова Г.М.**;

— Мустаева Айсылу и Хисматуллин Ильмир, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**;

— Николаев Владимир, средняя школа села Янтиково, Чувашская Республика, учитель **Николаева В.П.**;

— Порфирьева Анастасия, средняя школа села Янтиково, Чувашская Республика, учитель **Порфирьева И.В.**;

— Хасаншин Айдар, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Хотеев Сергей, Москва, гимназия № 1530, учитель **Шамшев М.В.**;

*Ответ*

Особенностью построения отдельных фрагментов стихотворения является не просто их повторение, а тот факт, что каждое следующее четверостишие связано с последним словом предыдущего фрагмента (вызывается этим словом и рассказывает о том, как начал писать поэму поэт). Получается, что каждое четверостишие как бы вызывает самого себя.

В программировании ситуацию в программе, когда процедура или функция вызывает в качестве вспомогательной саму себя, называют *рекурсией* (от лат. *recursio* — возвращение). Суть этого приема и иллюстрирует приведенное стихотворение. Об этом написали в своих ответах:

— Арчеков Роман и Орозбаев Алексей (лицей № 9 г. Волгограда);

— Васкан Александра, Желиазко Николь и Рачков Денис (гимназия № 1540 г. Москвы);

— Хотеев Сергей (гимназия № 1530 г. Москвы).

К сожалению, большинство читателей, приславших ответы, посчитали, что стихотворение иллюстрирует суть оператора цикла, который, как известно, обеспечивает повторение одной и той же последовательности действий (эти последовательности не "вызывают" друг друга). Заметим также, что при "простом" повторении фрагменты из четырех строк не были бы смещены друг относительно друга. Кроме того, в задании шла речь о "приеме, используемом в программировании", а не об алгоритмических (программных) конструкциях "цикл с параметром", "цикл с условием" и т.п.

## 5. Задача "Чему равны БИТ и БАЙТ?"

Напомним, что были предложены два задания, в которых фигурировали слова-числа **БИТ** и **БАЙТ**.

*Решение*

Запишем ребус в виде:

| × | **Б** | **И** | **Т** |
|---|---|---|---|
| | | | * |
| **Б** | **А** | **Й** | **Т** |

— где символ "*" может соответствовать любому однозначному числу.

Далее выпишем возможные сочетания значений буквы **Т** и символа "*", при которых 4-значное произведение **Т** на "*" оканчивается на **Т**:

| **Т** | * |
|---|---|
| 2 | 6 |
| 4 | 6 |
| 5 | 7 или 9 |
| 6 | 6 |
| 8 | 6 |

**Примечание**. При **Т** = 5 значения "*" (нечетные), меньшие 7, не подходят, так как в этих случаях даже число **Б**95, будучи умноженным на 5, не может дать четырехзначное произведение, начинающееся цифрой **Б** (убедитесь в этом!).

Для решения первого задания следует в первую очередь проанализировать значение символа "*", равное 6.

В этом случае единственно возможное значение **Б** равно 1 (также убедитесь в этом!). Результаты анализа возможных вариантов сочетания значений **Т** и **И** приведены в таблице:

| **Т** | **И** | **Возможен ли вариант?** | **Обоснование** |
|---|---|---|---|
| 2 | 7 | Да | 172 × 6 = 1032 |
| | 8 | Да | 182 × 6 = 1092 |
| | 9 | Нет | 192 × 6 = 1152 |
| 4 | 7 | Нет | 174 × 6 = 1044 |
| | 8 | Нет | 184 × 6 = 1104 |
| | 9 | Нет | 194 × 6 = 1164 |
| 6 | 7 | Да | 176 × 6 = 1056 |
| | 8 | Нет | 186 × 6 = 1116 |
| | 9 | Нет | 196 × 6 = 1176 |
| 8 | 6 | Нет | 168 × 6 = 1008 |
| | 7 | Да | 178 × 6 = 1068 |
| | 9 | Нет | 198 × 6 = 1188 |

Итак, при "*" = 6 возможные варианты имеются (их несколько), поэтому бóльшие значения символа "*" можно не анализировать.

*Ответ на первое задание*: 6.

При решении второго задания значение **И**, равное 9, можно не рассматривать. Но зато в этом случае следует исследовать и значения **Б**, равные 2, 3 и 4. Кроме того, сразу видно, что из четырех допустимых значений в последней таблице подходит только вариант **БИТ** = 182, **БАЙТ** = 1092 (учитывая, что коды букв "И" и "Й" в системе кодировки ASCII отличаются на 1). Анализ показывает, что кроме него есть еще одно решение ребуса — **БИТ** = 185, **БАЙТ** = 1295.

*Ответ на второе задание*:

1) **БИТ** = 182, **БАЙТ** = 1092;

2) **БИТ** = 185, **БАЙТ** = 1295.

**Примечание**. Если допустить, что символ "*" может соответствовать не только однозначному числу, то возможны и другие решения, одно из которых нашел Артем Хайретдинов: **БИТ** = 521, **БАЙТ** = 5731.

*Ответы прислали*:

— Акбарова Снежанна, Габдуллин Руслан, Зайнакаев Вадим, Коробейникова Полина, Матиенко Анна, Мулюков Руслан и Швецова Анна, Республика Башкортостан, Краснокамский р-н, село Николо-Берёзовка, школа № 1, учитель **Ситдикова А.Г.**;

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Базылев Юрий и Галушкова Карина, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Биякова Екатерина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Григоренко Василий, Григоренко Дмитрий, Есипова Мария, Круглякова Мария и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Деминцев Борис, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Мнацаканян Ашот, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Мурзин Алмаз, Республика Татарстан, Актанышский р-н, село Актаныш, средняя школа № 2, учитель **Гилязова Г.М.**;

— Хайретдинов Артем, Москва, Центр образования № 548 "Царицыно", учитель **Хайретдинова М.А.**

Лучшим признан ответ Карины Галушковой, которая правильно выполнила оба задания. Редакция решила наградить Карину дипломом. Поздравляем!

Заметим также, что в ряде ответов не учитывался тот факт, что одна и та же цифра не может быть зашифрована разными буквами.

## 6. Статья "Монеты на столе"

Напомним, что необходимо было определить, кто выиграет (начинающий ее или делающий ход вторым) в игру по следующим правилам: "Двое по очереди кладут одинаковые монеты на круглый стол, причем так,

чтобы они не накладывались друг на друга. Имеется неограниченное число монет. Проигрывает тот, кто не сможет сделать ход". Спрашивалось также, достаточно ли информации в условии, или для ответа необходимо знать размеры стола и монет?

*Решение*

Информации для ответа в условии достаточно. В этой игре выигрывает тот, кто начинает игру независимо от размеров стола! Первым ходом он должен положить монету так, чтобы центры монеты и стола совпали. После этого на каждый ход второго игрока начинающий должен класть монету симметрично относительно центра стола. При такой стратегии после каждого хода первого игрока позиция будет симметрична. Поэтому если возможен очередной ход второго игрока, то возможен и симметричный ему ответный ход первого. Когда-то делающий ход вторым разместить монету на столе не сможет. Следовательно, побеждает первый игрок.

**Примечание.** Конечно, мы не допускаем, что размеры стола и монет таковы, что начинающий игру не сможет разместить первую монету ☺.

*Ответы представили*:

— Аксенов Василий, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Васкан Александра, Желиазко Николь и Рачков Денис, Москва, гимназия № 1540, учитель **Пряхина Л.С.**;

— Деминцев Борис, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

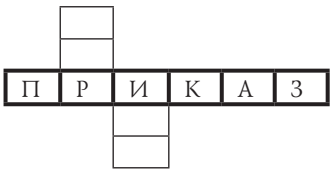
— Кальянова Анастасия и Тимофеева Юлия, г. Стерлитамак, Республика Башкортостан, школа № 1, учитель **Орлова Е.В.**;

— Фокеева Нина, Республика Башкортостан, г. Уфа, гимназия № 3 им. А.М. Горького, учитель **Болдырева С.В.**;

— Ященко Алексей, г. Ковров Владимирской обл., школа № 43, учитель **Федосеев С.А.**

## 7. Головоломка "Получить из «приказа» «каприз»"

Напомним, что необходимо было, перекладывая карточки с буквами на соседнюю свободную клетку листа, за минимальное количество перекладываний получить слово КАПРИЗ из слова ПРИКАЗ:




Минимальное число перекладываний, за которые можно решить задачу, — 26 (есть несколько вариантов решения).

*Ответы прислали*:

— Андрющенко Александр и Свистунов Николай, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Арчеков Роман и Орозбаев Алексей, г. Волгоград, лицей № 9, учитель **Широкова Л.В.**;

— Биякова Екатерина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**;

— Баженов Михаил, Кузин Михаил и Протасов Максим, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Васкан Александра, Желиазко Николь и Рачков Денис, Москва, гимназия № 1540, учитель **Пряхина Л.С.**;

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Гриценко Любовь и Толмачев Владимир, г. Рубцовск Алтайского края, школа № 1, учитель **Толмачева Н.П.**;

— Евсеева Надежда, Малов Дмитрий и Чуркина Анастасия, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Желтышева Елизавета, средняя школа поселка Суксун, Пермский край, учитель **Желтышева Л.А.**;

— Кальянова Анастасия и Тимофеева Юлия, г. Стерлитамак, Республика Башкортостан, школа № 1, учитель **Орлова Е.В.**;

— Карасева Анастасия и Пучкина Елена, Республика Башкортостан, г. Стерлитамак, гимназия № 5, учитель **Пучкина С.А.**;

— Каюмов Ансель, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Мустаева Айсылу, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**;

— Мякишева Ирина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Николаев Владимир, средняя школа села Янтиково, Чувашская Республика, учитель **Николаева В.П.**;

— Порфирьева Анастасия, средняя школа села Янтиково, Чувашская Республика, учитель **Порфирьева И.В.**;

— Хайретдинов Артем, Москва, Центр образования № 548 "Царицыно", учитель **Хайретдинова М.А.**

## 8. Восемь вопросов (рубрика "Поиск информации")

*Ответы на заданные вопросы*

1. По разным источникам, самой распространенной русской фамилией является *Иванов* или *Смирнов*.

2. Из четырех приведенных понятий, связанных с одеждой, *не* по фамилии генерала назван "макинтош" (по имени изобретателя ткани — шотландского химика Ч.Макинтоша).

3. "Советником грации" парижанки-модницы середины XVII века называли зеркало.

4. Великой детской писательнице Астрид Линдгрен в день ее 91-летия шведы преподнесли такой подарок — соорудили волшебную деревню, в которой "живут" герои ее книг.

5. Здесь в вопросе многие читатели не обратили внимания на слово "приправа" и на фразу "в разные времена". Еще в древности средством для улучшения умственной деятельности и афродизиаком считалась

горчица. Как написали в своих ответах Владимир Николаев и Анастасия Порфирьева из средней школы села Янтиково, Чувашская Республика, "...в античные времена лекари заметили, что эта пряность оказывает сильное тонизирующее действие на организм человека. Существовало мнение, что если пить горчицу натощак, это обостряет сообразительность".

6. "Голубой дом" — резиденция президента в Южной Корее.

7. Народные артисты Александр Калягин, Леонид Куравлев, Николай Губенко, Олег Басилашвили и Станислав Любшин пробовались на главную роль в фильме "Осенний марафон".

8. Впервые в мировой истории персидский царь Кир II Великий приговорил к "смертной казни" реку Диалу, правый приток Тигра, в которой у него трагически погиб любимый белый конь.

*Ответы представили:*

— Аржанов Дмитрий, Ашихина Полина, Григорьев Кирилл и Юхтенко Илья, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Бергер Юлия, Сухорученко Сергей, Холмогоров Виктор и Шишкина Анастасия, Свердловская обл., г. Нижняя Салда, школа № 7, учитель **Зорихина Н.Ю.**;

— Борисова Светлана, Гаврилова Дарья и Стафеева Елизавета, г. Новоуральск Свердловской обл., школа № 58, учитель **Стафеева Н.А.**;

— Васкан Александра, Желиазко Николь и Рачков Денис, Москва, гимназия № 1540, учитель **Пряхина Л.С.**;

— Великородная Светлана и Хомякова Валентина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Вылка Татьяна, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Желтышева Елизавета, средняя школа поселка Суксун, Пермский край, учитель **Желтышева Л.А.**;

— Кальянова Анастасия и Тимофеева Юлия, г. Стерлитамак, Республика Башкортостан, школа № 1, учитель **Орлова Е.В.**;

— Линьков Вячеслав и Протасов Максим, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Карасева Анастасия и Пучкина Елена, Республика Башкортостан, г. Стерлитамак, гимназия № 5, учитель **Пучкина С.А.**;

— Кирченкова Надежда, Москва, Центр образования № 1406 (школа для детей с нарушениями слуха), учитель **Миронова А.А.**;

— Николаев Владимир, средняя школа села Янтиково, Чувашская Республика, учитель **Николаева В.П.**;

— Порфирьева Анастасия, средняя школа села Янтиково, Чувашская Республика, учитель **Порфирьева И.В.**;

— Сорокин Вадим, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Тимофеева Ирина и Федоров Максим, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Фокеева Нина, Республика Башкортостан, г. Уфа, гимназия № 3 им. А.М. Горького, учитель **Болдырева С.В.**;

— Чугунков Евгений, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Ященко Алексей, г. Ковров Владимирской обл., школа № 43, учитель **Федосеев С.А.**

Программы, с помощью которых можно получить изображения пятиконечной звезды, предложенные в статье "Рисуем пятиконечную звезду", прислали:

— Базылев Юрий, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Богданова Л.М.**;

— Хотеев Сергей, Москва, гимназия № 1530, учитель **Шамшев М.В.**

Благодаря Сергея и Юрия, заметим, что в разработанных ими программах требуемые изображения получаются путем вырисовывания каждой линии с помощью отдельного оператора (операторы цикла, как в статье, не используются).

Правильные решения задач "В небольшом городке" и "Иностранные языки" представили также Богатырев Максим, Васьков Алексей, Горбачева Дарья, Добрынина Людмила, Елисеева Кристина, Ермолаев Александр, Рожкова Вера, Романычев Павел, Табакова Кристина, Фёклина Юлия, Фуфыгин Алексей, Чапаев Иван и Чукарёва Юлия, средняя школа села Кипцы, Саратовская обл., Екатериновский р-н, учитель **Омельченко С.Ю.**, а правильное решение числового ребуса "Сколько будет ОГОГО + УГУГУ?" — Игнатьева Сахая и Игнатьева Ульяна, средняя школа им. С.П. Барашкова села Качикатцы Хангаласского улуса, Республика Саха (Якутия), учитель **Яковлева М.Д.**

Участником конкурса № 79 "Снежинки" являлись также:

— Бариев Айрат, Республика Татарстан, г. Бавлы, гимназия № 4, учитель **Шафиков Н.Р.**;

— Биякова Екатерина, Биякова Ирина и Курманова Алина, средняя школа села Ложниково, Омская обл., Тарский р-н, учитель **Коровин Д.В.**

## Спорт и учеба

В классе 25 учеников. Из них 17 умеют играть в баскетбол, 13 умеют плавать, а 8 — ходить на лыжах. Ни один из учеников не владеет теми тремя видами спорта, но как баскетболисты, так и пловцы и лыжники имеют хорошие или удовлетворительные оценки по математике, что тем более знаменательно, так как шесть учеников имеют неудовлетворительные оценки по этому предмету.

Сколько учеников имеют отличные оценки по математике? Сколько пловцов умеют ходить на лыжах?

## Кусок мыла

После семи стирок длина, ширина и высота куска мыла уменьшилась вдвое. На сколько стирок хватит оставшегося куска?

## Кто куда поехал?

Сотрудники одного из отделов Харатьян, Ульянов, Зотов, Иванов и Василенко должны были поехать в разные города — Астрахань, Бийск, Вологду, Грозный, Смоленск и Екатеринбург. При этом Харатьян мог ехать только в Астрахань, Бийск или Смоленск; Ульянов — только в Бийск и Грозный, Зотов мог ехать только один и только в Вологду, Иванов не мог ехать вместе с Ульяновым, Василенко мог ехать только с Харатьяном или Зотовым, но не в Смоленск. В какой город поехал каждый из них, чтобы при этом вдвоем они не были ни в одном городе, а каждый побывал только в одном городе?

## Гривенники и пятиалтынные

В статье [1] были описаны некоторые задачи, решение которых целесообразно проводить, используя программу Microsoft Excel. С помощью указанной программы решите одну старинную задачу: "Сколько существует способов составить сумму в 6 рублей из одних только гривенников (10 коп.) и пятиалтынных (15 коп.)?"

**Указания по выполнению**

Если обозначить число гривенников в некой комбинации, соответствующей условию задачи, буквой $x$, а количество пятиалтынных — буквой $y$, то можем записать:

$$10x + 15y = 600,$$

откуда $y = (600 - 10x)/15$.

Использовав полученную зависимость, можем получить на листе следующие данные:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | *x* | *y* | | |
| 2 | 0 | 40 | 600 | |
| 3 | 1 | 39,33333333 | 600 | |
| … | | | | |

**Примечания**

1. Максимально возможное значение $x$ можно получить, рассмотрев случай $y = 0$.
2. Рассчитанные в столбце C значения приведены для контроля.

Условие, по которому можно определить, подходит тот или иной способ, установите самостоятельно. Количество подходящих способов можно подсчитать "вручную" или использовав стандартные функции Excel.

Ответы присылайте в редакцию.

## Литература

1. Информатика помогает математике. / "В мир информатики" № 152, 154 ("Информатика" № 22, 24/2010).

# "ЛОМАЕМ" ГОЛОВУ

## Помогите археологу[1]

Археолог обнаружил обрывок древней рукописи, на котором было изображение, аналогичное показанному на *рис.* 1. Он обращается ко всем: "Помогите мне понять, что это означает".

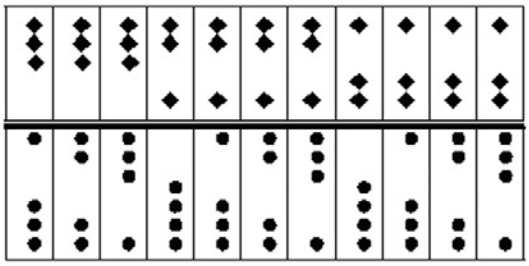


*Рис.* 1. Прорисовка изображения из древней рукописи

Если у вас есть какие-то соображения, помогающие археологу разобраться в этом вопросе, пожалуйста, присылайте их в редакцию.

## Ребусы по информатике

Мы завершаем[2] публикацию ребусов, разработанных учениками школы № 2 им. Н.П. Массонова из г. Свислочь Гродненской обл., Республика Беларусь (учитель **Синица А.А.**). Решите, пожалуйста, эти ребусы. Определите также, с какой темой они связаны. Дайте определение каждому из найденных терминов.

**Ребус № 1**


**Ребус № 2**


[1] Задачу представил М.А. Цайгер.

[2] См. газету-вкладку "В мир информатики" № 149, 151, 153 ("Информатика" № 19, 21, 23/2010).

**Ребус № 3**


**Ребус № 4**


## Числовой ребус “СЁСТРЫ”

Решите, пожалуйста, числовой ребус:

**НАТАША + ТОНЯ = СЁСТРЫ**

Как принято в таких головоломках, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры.

# GAMES.EXE

## Доминошки на клетчатой доске

Двое играют в такую игру на клетчатой доске размером 10 × 10 клеток. За ход разрешается накрыть любые две соседние клетки доминошкой (прямоугольником 1 × 2 клетки) так, чтобы доминошки не перекрывались. Проигрывает тот, кто не сможет сделать ход. Кто выиграет — начинающий ее или делающий ход вторым?

# ЗАДАЧНИК

## Странный муж

Некий мужчина отправляется на работу, которая находится на расстоянии 1 км от дома. Дойдя до места работы, он вдруг вспоминает, что перед уходом забыл поцеловать жену, и поворачивает назад. Пройдя полпути, он меняет решение, посчитав, что правильнее вернуться на работу. Пройдя 1/3 км по направлению к работе, он вдруг осознает, что будет настоящим подлецом, если так и не поцелует жену. На этот раз, прежде чем изменить мнение, он проходит 1/4 км. Так он продолжает метаться, и после $N$-го этапа, пройдя $1/N$ км, снова меняет решение.

**1.** Разработав программу (на языке программирования, который вы изучаете), определите:

1) на каком расстоянии от дома будет находиться мужчина после 100-го этапа (если допустить, что такое возможно);

2) какой общий путь он при этом пройдет;

3) на каком расстоянии от дома будет находиться мужчина после того, как на очередном этапе он пройдет меньше, чем 1 м (если также допустить, что это возможно);

4) на каком этапе это произойдет.

**2.** Получите ответы на заданные в пп. 1 и 2 вопросы, оформив лист электронной таблицы Microsoft Excel.

Ответы присылайте в редакцию (можно выполнять не все задания).

# ВНИМАНИЕ! КОНКУРС

## Конкурс № 82

В качестве заданий этого конкурса предлагаем выполнить задания для самостоятельной работы, предложенные в статье “Обмены, обмены...” в этом выпуске.

## Конкурс № 83

В качестве заданий этого конкурса предлагаем решить задачу “Кто куда поехал?” (см. выше).

## Конкурс № 84

Заданием этого конкурса является решение задачи “Гривенники и пятиалтынные” (см. статью с аналогичным названием в этом выпуске) с использованием стандартных функций программы Microsoft Excel. Ответ получите в ячейке C65:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | X | Y | | |
| 2 | 0 | 40 | 600 | |
| 3 | 1 | 39,33333333 | 600 | |
| ... | | | | |
| 65 | | Искомое число способов: | | |

Программы или/и ответ или/и лист Excel отправьте в редакцию до 1 февраля по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*. Пожалуйста, четко укажите в ответе свои фамилию и имя, населенный пункт, номер и адрес школы, фамилию, имя и отчество учителя информатики.

*LZW — один из самых известных и до сих пор широко используемых алгоритмов сжатия без потерь. Из актуальных на сегодняшний день применений этого алгоритма в первую очередь можно упомянуть компрессию PDF-файлов, TIFF-файлов (опционально) и, конечно, формат GIF, который в свое время стал настоящим форматом раздора в компьютерном мире.*


Якоб Зив

# Алгоритм раздора

Сначала о том, что такое сжатие (компрессия) без потерь. При таком способе сжатия информация, восстановленная из сжатого состояния, в точности равна оригиналу. Кажется, что это единственно возможный вариант — а как же иначе? Разве можно сжимать так, чтобы нельзя было в точности восстановить исходную информацию? Оказывается — можно. Например, в звуковых файлах могут присутствовать фрагменты, которые мы либо вообще не слышим, либо для данной конкретной цели прослушивания они не существенны — обычную речь мы легко разбираем даже в весьма посредственном качестве. Сжатие без возможности восстановления таких "избыточных" фрагментов и называется сжатием с потерями.

Разумеется, сжимать с потерями можно далеко не всегда. Типичный пример — сжатие исполняемого кода программ. Заманчиво, конечно, выкинуть сотню-другую "лишних" битов и запустить программу — авось заработает ☺. Но лучше не экспериментировать. В подобных случаях требуется сжимать без потерь, с возможностью точного восстановления. Одной из первых широко распространенных программ для сжатия без потерь была программа compress для операционной системы Unix. Эта программа появилась в 1986 г. (ну, или около этого времени, точную дату установить сейчас трудно). В ней как раз и был реализован алгоритм LZW.

Принципиально сжатие без потерь, как правило, основано на одной из двух ключевых идей. Идея первая — повторяющиеся последовательности одинаковых фрагментов заменяются на блок, состоящий из указанного фрагмента и счетчика — количества повторений. Подобный алгоритм замечательно сожмет, например, рисунок с большим количеством участков одного цвета. На этой идее основан, например, алгоритм RLE (***Run-length encoding***) и его модификации. Этот алгоритм использовался в графическом формате PCX. Помните такой ☺?

Вторая идея такова: если в сжимаемых данных встречаются повторяющиеся фрагменты, их можно заменить некими кодами, более короткими, чем сами эти фрагменты. Например, в тексте "Вася встретил Васятку" можно выделить повторяющийся фрагмент "Вася", закодировать его, например 1, и преобразовать текст в "1 встретил 1тку" со словарем "1 Вася". Алгоритм LZW основан именно на такой идее.

Название алгоритма LZW в отличие от RLE образовано как аббревиатура от первых букв фамилий его разработчиков — Абрахама Лемпеля, Якоба Зива и Терри Велча. Сам алгоритм был опубликован именно Велчем в 1984 г., как улучшенная реализация LZ78, разработанного в 1978 г. Лемпелем и Зивом. Интересно, что патент на алгоритм при этом принадлежал одному Зиву и компании Unisys. "Принадлежал" потому, что в 2003-м и 2004 годах истекли сроки действия всех патентов на алгоритм в США и Европе, и более на использование алгоритма нет патентных ограничений.

Именно эти ограничения в свое время стали причиной появления формата PNG, который был разработан в качестве бесплатной альтернативы формату GIF, в основе которого лежал LZW. И, соответственно, все производители программного обеспечения, способного сохранять файлы в формате GIF, должны были выплачивать патентные отчисления.

*Дополнительная информация об алгоритме LZW и компьютерная презентация имеются на диске, который будет вложен в следующий номер "Информатики" (№ 2/2011).*