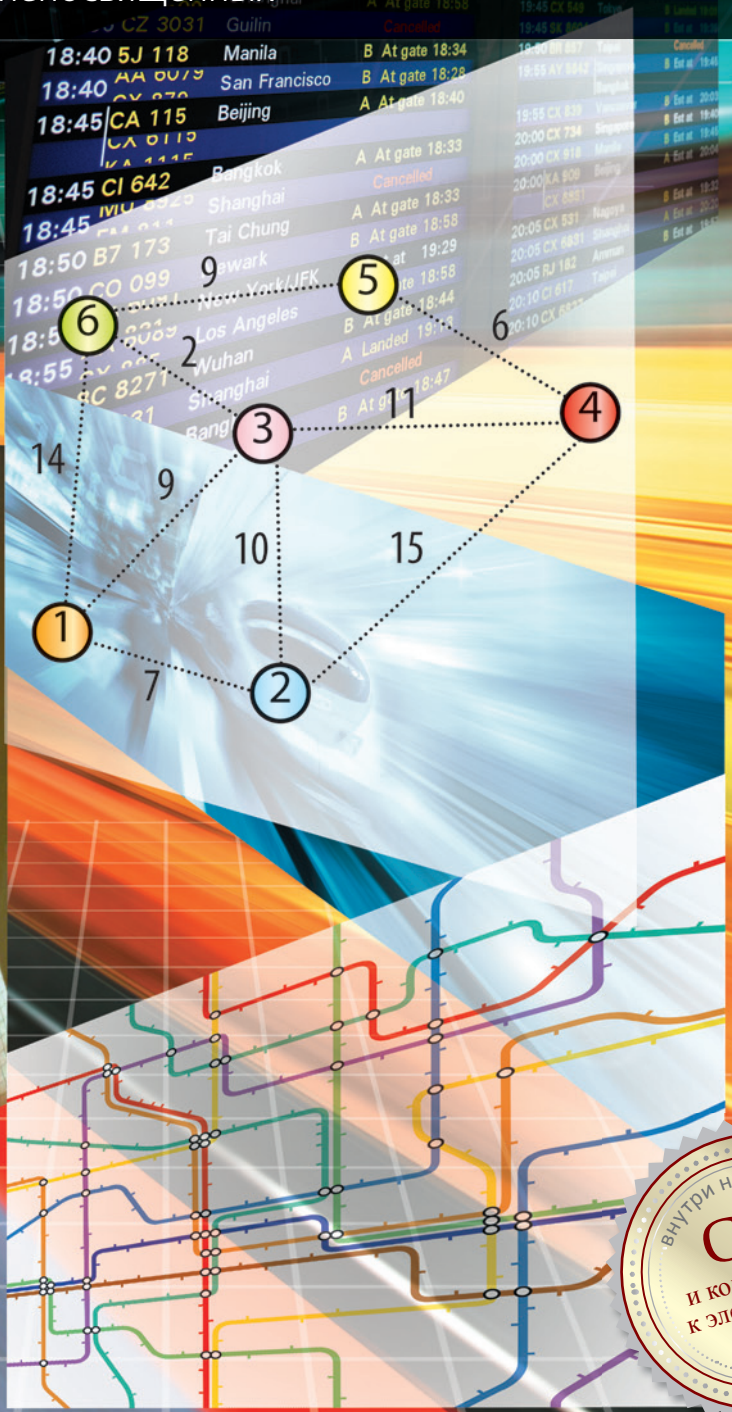
№4 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА



внутри номера CD и код доступа к электронной версии

издательский дом 1september.ru
Первое сентября
апрель 2012
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

Алгоритм Дейкстры — сам по себе интересный, важный, красивый, широко используемый (список можно продолжить) — среди прочих обладает интересным свойством. В определенном смысле он лежит на границе “обычной” (правда, профильной) и олимпиадной школьной информатики. Причем находится он именно в пересечении указанных областей. Для многих этот алгоритм — последняя ступенька общеобразовательного курса и первая на пути к олимпиадным вершинам.

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Дидактические, раздаточные и демонстрационные материалы к статьям
- Исходные коды программ
- Презентации для работы в классе

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ: по каталогу “Роспечати”: 32291 (бумажная версия), 19179 (электронная версия); “Почта России”: 79066 (бумажная версия), 12684 (электронная версия)

http://inf.1september.ru

Учебно-методический журнал для учителей информатики
Основан в 1995 г.
Выходит один раз в месяц

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 19 817 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, конференции и техническое обеспечение Издательского дома: Павел Кузнецов
Производство: Станислав Савельев
Административно-хозяйственное обеспечение: Андрей Ушков
Главный художник: Иван Лукьянов
Педагогический университет: Валерия Арсланьян (ректор)

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА
Английский язык – А.Громушкина
Библиотека в школе – О.Громова
Биология – Н.Иванова
География – О.Коротова
Дошкольное образование – Д.Тюттерин
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 – Е.Рачевский
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школьный психолог – И.Вачков

УЧРЕДИТЕЛЬ: ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011 в Министерстве РФ по делам печати
Подписано в печать: по графику 12.03.2012, фактически 12.03.2012
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

“Что такое, товарищи, дебют и что такое, товарищи, идея?”, или Читал пейджер, много думал

Классическим SMS все большее количество интернет-пользователей предпочитают Twitter. Дело зашло так далеко, что свой аккаунт в Twitter'е — @minobrnauki_ru — имеет и Министерство образования и науки

Если наши дети увлеченно и с бешеной скоростью барабанят по кнопкам мобильного телефона (или теперь уже непосредственно по экрану), не факт, что они “эсэмэсят”. Классическим SMS все большее количество интернет-пользователей предпочитают Twitter. Дело зашло так далеко, что свой аккаунт в Twitter'е — @minobrnauki_ru — имеет и Министерство образования и науки. Дело зашло еще дальше! ☺ Этот аккаунт реально используется, и в нем появляются вполне информативные сообщения. Так с чем же едят эту птичку?

Формально говоря, Twitter является сервисом микроблогов. Каждый может завести себе аккаунт и начать вести такой микроблог, сообщения в которых называются “твитами”. Каждый может читать микроблоги других авторов. Тот, кто читает, — “фолловер”. От английского *follower* — последователь, но оттенок “сектанства”, который имеется в русском переводе, на самом деле лишний и случайный. Можно считать, что “фолловер” — подписчик, это более соответствует русской семантике. Как и в “нормальных” блогах, подписчики могут комментировать посты блоггера, отправляя свои сообщения в ответ.

Из различных социальных сервисов Twitter заимствовал наиболее востребованные возможности. Это и идея “перепоста” (здесь называется “ретвит”), и “облако тегов” (чаще всего состоящее из одного тега), которым можно снабжать свои твиты. Можно размещать в твите изображение, можно снабжать твит геолокационной меткой. Разумеется, имеется возможность отправить и прямое “личное” сообщение конкретному адресату. Правда, увидят такое сообщение и все ваши фолловеры.

Но самое главное, пожалуй, не это. Главная “фишка” Twitter’а в длине твита. Она не просто маленькая, она — крошечная! Трудно поверить — всего 140 символов! Длина SMS и то 160! (Правда, на английском — на русском меньше.) Но именно краткость и телеграфный стиль оказались той самой плодотворной идеей — востребованной нишей общения в Интернете, которая не была занята. Ограничение на длину твита создает массу неудобств, которые приходится мужественно преодолевать. Например, ссылки на интернет-ресурсы приходится сокращать посредством специальных сервисов. Иногда и обычные слова сокращать приходится. Если же просто “распирает” и хочется “твитнуть” больше, чем позволяют технические ограничения, нет другого выхода, как отправлять несколько последовательных твитов. Жутко неудобно! Но… что-то в этом есть ☺.

Аудитория Twitter увеличивается очень быстро, а российский сегмент сети растет быстрее других. Сравнительно недавно интерфейс web-клиента и системы регистрации Twitter’а был локализован, что тоже способствует притоку русскоязычных пользователей. Кстати, клиенты имеются для различных мобильных операционных систем, так что с доступом в Twitter, в том числе и с мобильных устройств — телефонов, планшетов, проблем нет.

Попробуйте! Писать и становиться блоггером вас никто не заставляет. Можно просто читать и всегда быть в курсе самых свежих новостей из самых первых рук.

О пользе владения формальными преобразованиями в алгебре логики

И.Н. Фалина,
Москва

Каждый из нас скорее всего задумывался над вопросом: “Что выносят наши ученики из информатики в “большую жизнь” после окончания школы?” По большому счету — знания о современном информационном мире, умения применять полученные знания и т.д. Все это верно. Но что конкретно им запомнится из школьного курса информатики? Что им, например, пригодится, когда их собственные дети будут учиться в школе? На такие вопросы нет однозначного ответа. Я рискну предложить свой взгляд на эту проблему, вернее, один из возможных ответов на поставленные вопросы.

Речь идет об умении выполнять формальные тождественные преобразования в алгебре логики. Чтобы убедить вас, мне понадобится выполнить несколько шагов.

Шаг первый. Решение задания А10 в ЕГЭ по информатике

Для задания А10 в спецификации ЕГЭ по информатике указано, что это задание проверяет “Знание основных понятий и законов математической логики”, относится оно к повышенному уровню сложности, примерное время его выполнения — 2 минуты.

Как правило, школьники решают это задание проверкой всех вариантов ответа, да именно так и учат школьников решать это задание в подавляющем большинстве школ.

Но решать такие задачи можно и по-другому, с применением формальных методов преобразования логических формул. Рассмотрим несколько таких заданий.

Пример 1. Для какого имени истинно высказывание:

(первая буква согласная → последняя буква согласная) ∧ (первая буква гласная → последняя буква гласная)?

1) Анна
2) Белла
3) Никита
4) Олег

Решение. Введем следующие обозначения:

1*С* — первая буква согласная;
1*Г* — первая буква гласная;
ПС — последняя буква согласная;
ПГ — последняя буква гласная.

Запишем наше высказывание в виде логической формулы, используя для обозначения конъюнкции знак умножения.

$$(1С \to ПС)\cdot(1Г \to ПГ) =$$
$$= (\overline{1С} \vee ПС)\cdot(\overline{1Г} \vee ПГ) =$$
$$= (1Г \vee ПС)\cdot(1С \vee ПГ) =$$
$$= (1Г\cdot 1С \vee 1Г\cdot ПГ \vee ПС\cdot 1С \vee ПС\cdot ПГ) =$$
$$= 1Г\cdot ПГ \vee 1С\cdot ПС.$$

При тождественных преобразованиях были использованы следующие факты:

1) $x \to y = \overline{x} \vee y$;

2) $\overline{1С} = 1Г$;

3) $1Г \cdot 1С = 0$ (буква не может быть одновременно и гласной, и согласной).

Таким образом, наше высказывание будет истинно в двух случаях: когда первая буква и последняя буква в слове — гласные или когда первая и последняя буквы — согласные. Конечно, мы должны теперь проверить все четыре варианта ответов. Но проверка такого условия существенно проще, чем проверка исходного условия.

Ответ: Анна.

Анализ решения. Можно утверждать, что если школьник уверенно владеет навыками тождественных преобразований логических выражений, то предлагаемый способ решения более надежен (т.е. менее вероятно им будет допущена ошибка).

Можно ли эту задачу предлагаемым способом решить за две минуты? За две минуты скорее всего нет, но за 3–4 минуты можно, это проверено на десятках школьников.

Сложно ли школьникам овладеть навыками тождественных преобразований так, чтобы их применить в стрессовой ситуации? Можно, если отработку этих навыков выполнять на различных специально подобранных задачах: при решении логических задач, при изучении алгоритмизации и программирования и баз данных. Но об этом позже.

Пример 2. Для какого имени ложно высказывание:

(первая буква гласная $\wedge$ последняя буква согласная) $\to$ $\neg$(третья буква согласная)?

1) Дмитрий
2) Антон
3) Екатерина
4) Анатолий

Решение. Используя интуитивно понятные обозначения (как и в примере 1), запишем наше высказывание в виде формулы:

$$(1Г \cdot ПС) \to \overline{3С} = \overline{(1Г \cdot ПС)} \vee 3Г =$$
$$= \overline{1Г} \vee \overline{ПС} \vee 3Г = 1С \vee ПГ \vee 3Г.$$

Далее, по условию задачи надо найти, для какого имени оно ложно. Или, что то же самое, для какого имени истинно отрицание этого высказывания:

$$\overline{1С \vee ПГ \vee 3Г} = \overline{1С} \cdot \overline{ПГ} \cdot \overline{3Г} = 1Г \cdot ПС \cdot 3С.$$

Выбрать имя, удовлетворяющее этому условию, уже не составляет никакого труда.

Ответ: Антон.

Анализ решения. Для решения таким способом школьник должен знать и уметь применять законы де Моргана. Но, как правило, ученики законы де Моргана запоминают легко (может, этому способствует имя закона? ☺).

Сделать это задание методом прямой проверки не очень сложно, но надо не запутаться и исключить те имена, которые удовлетворяют условию.

Пример 3. Для какого имени истинно высказывание:

$\neg$ (первая буква согласная $\to$ вторая буква согласная) $\wedge$ последняя буква согласная?

1) Ирина
2) Максим
3) Степан
4) Мария

Решение. Запишем высказывание в виде формулы:

$$\overline{(1С \to 2С)} \cdot ПС = \overline{(\overline{1С} \vee 2С)} \cdot ПС = (\overline{1Г \vee 2С}) \cdot ПС =$$
$$= \overline{1Г} \cdot \overline{2С} \cdot ПС = 1С \cdot 2Г \cdot ПС.$$

Ответ: Максим.

Пример 4 (*демоверсия ЕГЭ*-2012).

Для какого имени истинно высказывание:

(первая буква согласная $\to$ вторая буква согласная) $\wedge$ (предпоследняя буква гласная $\to$ последняя буква гласная)?

1) Максим
2) Степан
3) Кристина
4) Мария

Решение

$$(1С \to 2С) \cdot (ПрГ \to ПГ) = (\overline{1С} \vee 2С) \cdot (\overline{ПрГ} \vee ПГ) =$$
$$= 1Г \cdot ПрС \vee 1Г \cdot ПГ \vee 2С \cdot ПрС \vee 2С \cdot ПГ.$$

Первые два слагаемых *на множестве проверяемых имен* равны нулю (лжи), так как в предлагаемых ответах нет имен, начинающихся с гласных букв. Для рассмотрения остается только два имени: “Степан” и “Кристина”.

Оба имени удовлетворяют условию: вторая буква — согласная. Но имя “Степан” не удовлетворяет ни одному из условий: “предпоследняя буква согласная” или “последняя буква гласная”. А имя “Кристина” удовлетворяет даже обоим условиям (хотя хватило бы выполнения и одного какого-нибудь условия).

Ответ: Кристина.

Если же решать эту задачу методом прямой проверки, то школьник должен помнить, что при ложной посылке импликация всегда истинна (что бы ни следовало из лжи, результат всегда будет истинен). Проверив все четыре имени, убеждаемся, что первый член в конъюнкции истинен для всех имен. Следовательно, мы должны еще раз выполнить проверку всех четырех имен для второго члена конъюнкции и убедиться, что только на одном имени этот член даст истину, а на остальных трех — ложь.

Каким способом можно решить это задание быстрее и, главное, без ошибки — решать вам. Но в любом случае овладение способом тождественных преобразований полезно для решения других задач.

Шаг второй. Решение задач на массивы

Предположим, что ученикам 10-х или 11-х классов вашей школы, которые изучают программирование на каком-либо алгоритмическом языке, предложили решить следующую задачу.

Задача 1. В символьном массиве длины *n* через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква гласная $\wedge$ последняя буква согласная) $\rightarrow \neg$(третья буква согласная). Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Как вы думаете, что в решении этой задачи вашим школьникам покажется самым сложным: умение выделить очередное слово? организация проверки условия для выделенного слова?

Анализ стандартных рассуждений и решений школьников

Аналогичные задачи мы предлагаем 10-классникам после изучения темы “Алгебра логики” и после того, как ими были написаны программы для решения задач типа:

1) В символьном массиве длины *n* через пробел записаны слова. Подсчитать количество слов в массиве;

2) В символьном массиве длины *n* через пробел записаны слова. Найти самое длинное слово и вывести его на экран.

При решении задачи 1 ребята начинают рассуждать так. Будем выделять из массива очередное слово. Для каждого слова будем проверять, выполняется это условие или нет. Если условие выполняется, то слово выводим на экран. Эти действия выполняем до тех пор, пока не будут обработаны все слова массива.

Начинаем реализовывать этот алгоритм в виде программы. (В нашей школе подавляющее большинство школьников пишет на Паскале.)

Мы учим школьников разрабатывать программы по принципу “достраивания работающей программы”. Это значит, что школьник должен выполнить декомпозицию задачи. А далее программа сначала отлаживается только для первой подзадачи. Затем к ней дописывается (достраивается) второй блок. И программа отлаживается уже для первых двух блоков. И так далее. После завершения очередного этапа вы всегда получаете работающую программу. Школьники в программе выделили такие подзадачи:

1) ввод исходных данных;

2) выделение очередного слова;

3) проверка условия для выделенного слова;

4) вывод слова на экран.

В соответствии с приведенными подзадачами школьники сначала пишут программу, которая выполняет только ввод исходных данных.

```
const n = 20;
var a: array [1..n] of char;
    i: integer;
begin
  for i := 1 to n do read(a[i]);
  for i := 1 to n do write(a[i]);
  writeln;
end.
```

Предлагаю школьникам воспользоваться типизированным описанием массива: всегда видим, что находится в массиве, и заполнять массив не надо каждый раз.

```
const n = 23;
 a: array [1..n] of char =
    'anna bella Nikita Artur';
```

Замечание 1. На этом массиве правильно работающая программа должна вывести на экран

```
anna
bella
Nikita
```

Имена и использование регистров (имена `anna` и `bella` записаны с маленькой буквы) подобраны не случайно. На этих именах можно выловить все ошибки, которые, как правило, допускают школьники при написании этой программы.

Далее школьники пишут блок выделения очередного слова. На этом этапе возникает вопрос: а куда записывать выделенное слово? Очевидно, что надо делать дополнительный массив, но какой длины? После обсуждения приходим к выводу, что длина дополнительного массива, в который будем записывать очередное имя, должна быть на единицу больше самого длинного известного имени (на всякий случай ☺). Решаем, что самое длинное известное имя — Александра (10 букв).

В приведенной ниже программе очередное слово записывается в символьный массив `slovo`. Если все делать аккуратно, то можно не обнулять массив `slovo` в цикле перед записью очередного слова.

```
const n = 23;
      m = 11;
 a: array [1..n] of char =
    'anna bella Nikita Artur';
var i, j, k: integer;
    slovo: array [1..m] of char;
begin
   j := 1;
   for i := 1 to n do
     if a[i] <> ' ' then
     begin
     {запись очередного слова}
       slovo[j] := a[i];
       j := j + 1;
     end
     else
     begin
       for k := 1 to j do write(slovo[k]);
       writeln;
       j := 1
     end;
  end.
```

Но в результате запуска написанной программы *последнее слово не выводится на экран*. В чем дело? Оказывается, что в программе признаком окончания очередного имени является пробел. А после последнего имени в массиве нет пробела! Выход — увеличить размерность исходного массива и после последнего имени тоже поставить пробел. Это известный *метод барьерного элемента*. Дописывание после последнего имени пробела (барьера!) позволяет все слова массива обрабатывать по одному и тому же алгоритму.

Затем ребята приступают к написанию блока определения истинности условия для выделенного слова. (Обсудили, что блок проверки нужно вставить в опцию `else` условного оператора.) У школьников возникает чисто технический вопрос: как понять, что буква является гласной или согласной? В результате общения между собой (ученикам разрешается обсуждать возникающие проблемы друг с другом) школьники стали использовать оператор `case` и для каждого простого высказывания из условия задачи завели булевскую переменную. Например:

*G*1 — булевская переменная, которая будет равна `true`, если первая буква гласная, и `false` в противном случае.

```
case slovo[1] of
'a','e','i','o','y','u': g1 := true
else g1 := false
end;
```

Я вижу, что текст программы у ребят разрастается, программа становится "вязкой". Тогда я предлагаю в блоке проверки использовать множество и операцию вхождения в множество `in`. Договорились, что в блоке `const` опишем переменную `Let: set of char = ['a', 'e', 'i', 'o', 'u', 'y']`. (Школьники до этого момента не были знакомы с типом "множество", но объяснение использования таких переменных не заняло много времени.) Но при программировании собственно условия ребята (практически все!) допускают логическую ошибку. Школьники реализуют условие, в котором есть импликация, следующим образом:

```
p := j - 1;
if (slovo[1] in let)
    and not (slovo[j] in let)
then
  if (slovo[3] in let)
  then
    for k := 1 to p do write(slovo[k])
```

То есть они реализуют правильно только две строчки из таблицы истинности операции импликации: $f(1, 1) = 1$ и $f(1, 0) = 0$. Про то, что $f(0, 1) = 1$ и $f(0, 0) = 0$, школьники забыли! В результате на данном значении массива программа не печатает ни одного имени.

Надо искать ошибку. Школьники понимают, что ошибка где-то в операторе IF. Как только ни пытаются они переделать этот оператор! Но все же приходится возвращаться к основам алгебры логики.

После выполнения тождественных преобразований логическое условие задачи имеет вид:

$$(1Г \cdot ПС) \to \overline{3С} = \overline{(1Г \cdot ПС)} \vee 3Г =$$
$$= \overline{1Г} \vee \overline{ПС} \vee 3Г = 1С \vee ПГ \vee 3Г.$$

На данном массиве программа должна вывести три первых имени.

Тогда школьники с облегчением (отмучились!) исправляют программу, которая принимает примерно такой вид:

```
const n = 24;
      m = 11;
      a: array [1..n] of char =
         'anna bella Nikita Artur ';
      let: set of char=
          ['a','e','i','y','o','u'];
var i, j, k, p: integer;
    slovo: array [1..m] of char;
begin
   j := 1;
   for i := 1 to n do
     if a[i] <> ' ' then
     begin
       slovo[j] := a[i];
       j := j + 1;
     end
     else
     begin
       p := j - 1;
       if not (slovo[1] in let)
          or (slovo[p] in let)
          or (slovo[3] in let)
       then for k := 1 to p do
              write(slovo[k]);
       writeln;
       j := 1
     end;
end.
```

Но программа выводит все четыре имени вместо трех. То есть она опять работает неверно! В результате обсуждения вспоминаем, что строчные и заглавные буквы в кодировочных таблицах имеют разные коды. Решено использовать функцию Upcase, а все буквы в множестве Let сделать заглавными.

Ура! Программа написана. Работает верно. На написание этой программы ушло практически два урока. В программе используются следующие переменные:

n — размерность исходного массива, увеличенная на 1 (для реализации метода барьерных элементов);

m — размерность массива для хранения выделенного имени;

a — исходный массив слов;

slovo — массив для хранения и обработки очередного имени;

let — множество, состоящее из гласных букв английского алфавита;

i, *j*, *k* — переменные цикла;

p — количество букв в очередном выделенном слове.

```
const n = 24;
      m = 11;
      a: array [1..n] of char =
        'anna bella Nikita Artur ';
      let: set of char =
          ['A','E','I','Y','O','U'];
var i, j, k, p: integer;
    slovo: array [1..m] of char;
begin
   j := 1;
   for i := 1 to n do
     if a[i] <> ' ' then
     begin
       slovo[j] := a[i];
       j := j + 1;
     end
     else
     begin
      p := j - 1;
      if not (Upcase(slovo[1]) in let)
         or (Upcase(slovo[p]) in let)
         or (Upcase(slovo[3]) in let)
      then for k := 1 to p do
           write(slovo[k]);
      writeln;
      j := 1
     end;
end.
```

Замечание 2. Изменяя условие истинности (или ложности), можно сформулировать несколько вариантов различных условий таких задач. Примеры условий задач с исходными значениями массивов даны в файле "Задачи" на диске. Более сильным школьникам можно дать задачу с требованием не использовать дополнительный массив.

Замечание 3. Можно изменить условие задачи следующим образом:

Задача 2. В символьном массиве длины *n* записаны имена латинскими буквами. Между словами, а также в начале массива и в конце его может быть *несколько* пробелов. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква гласная $\wedge$ последняя буква согласная) $\rightarrow$ $\neg$(третья буква согласная). Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание 4. Разобранная задача позволяет сделать акцент и на технических вещах (например, ввод массива, кодировочные таблицы), и на методических (использование метода барьерного элемента), и на алгоритмических (правильная запись сложного логического условия на алгоритмическом языке).

Замечание 5. Если вы не давали своим ученикам задание "Написать таблицу истинности операции импликация $x \rightarrow y$", то найдите возможность дать такое задание.

Школьники, как правило, пытаются сделать это задание с использованием оператора IF, но делают ошибку, о которой говорилось выше.

Примеры верно написанных программ, которые следует обсудить с учениками.

С использованием оператора IF

```
var x, y, f: boolean;
begin
  for x := false to true do
    for y := false to true do
    begin
      if not x then f := true
      else
      begin
        if y then f := true
        else f := false
      end;
      writeln(x:6, y:6, f:6)
    end
end.
```

С использованием оператора IF и более компактной записью условного оператора в опции else

```
var x, y, f: boolean;
begin
  for x := false to true do
    for y := false to true do
    begin
      if not x then f := true
      else f := y;
     writeln(x:6, y:6, f:6)
  end
end.
```

Без использования оператора IF

```
var x, y, f: boolean;
begin
  for x := false to true do
    for y := false to true do
    begin
      f := not x or y;
      writeln(x:6, y:6, f:6)
    end
end.
```

Заметим, что операцию импликации можно запрограммировать и с помощью одной операции сравнения, не прибегая к преобразованию ее в дизъюнкцию. Тогда оператор присваивания в последнем примере будет выглядеть так:

```
f := x <= y;
```

Это следует из таблицы истинности операции импликация.

Шаг третий. Тема "Процедуры и функции"

При изучении процедур и функций также полезно использовать задачи такого вида.

Задача 3. В символьном массиве длины *n* через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква гласная ∧ последняя буква согласная) → ¬(третья буква согласная). Гласными буквами в английском алфавите являются {a, e, i, o, u, y}. В программе должна быть логическая функция, которая определяет, удовлетворяет ли конкретное имя условию задачи.

Замечание 5. Очевидно, что здесь, как и в Шаге 2, можно сформулировать много однотипных условий задач и каждому ученику класса дать свой вариант задания.

Возможный текст программы:

```
const n = 24;
      m = 11;
      a: array [1..n] of
       char = 'anna bella Nikita Artur ';
type Tslovo = array [1..m] of char;
var i, j, k: integer;
    slovo: Tslovo;
function Uslov(b: Tslovo;
               dl: integer): boolean;
 const let: set of
       char = ['A','E','I','Y','O','U'];
 begin
  Uslov := not (Upcase(b[1]) in let)
           or (Upcase(b[dl]) in let)
           or (Upcase(b[3]) in let)
 end;
{-----------------}
begin
   j := 1;
   for i := 1 to n do
     if a[i] <> ' ' then
     begin
       slovo[j] := a[i];
       j := j + 1;
     end
     else
     begin
       if uslov(slovo, j - 1)
       then
         for k := 1 to j - 1 do
           write(slovo[k]);
       writeln;
       j := 1;
     end;
   readln
end.
```

Шаг четвертый. Решение задач на обработку файлов

Этот же тип задач можно использовать и при изучении строк и текстовых файлов.

Задача 4. В текстовом файле Viktorina.txt построчно записаны фамилии и имена людей, принявших участие в sms-викторине. Так как устроители викторины не ожидали такого количества правильных ответов, они решили наградить тех участников, чьи имена удовлетворяют условию: (Первая буква гласная ∧ Последняя буква согласная) → ¬(Третья буква согласная).

Требуется сформировать текстовый файл conquer.txt, в который надо построчно поместить фамилии и имена награжденных.

Замечание 6. Здесь, так же как и в Шагах 2 и 3, можно сформулировать много однотипных условий задач и каждому ученику класса дать свой вариант задания.

Возможный текст программы:

```
var inf, outf: text;
    s, slovo: string;
    i: integer;
function Uslov(b: string):boolean;
const let: set of
      char = ['A','E','I','Y','O','U'];
begin
  Uslov := not (upcase(b[1]) in let)
         or (upcase(b[length(b)]) in let)
         or (upcase(b[3]) in let)
end;
{-----------------}
begin
  assign (inf, 'victorina.txt');
  reset(inf);
  assign (outf, 'conquer.txt');
  rewrite(outf);
  while not eof(inf) do
  begin
    readln(inf, s);
    i := 1;
    slovo := '';
   while (i <= length(s)) and (s[i] = ' ')
    do i := i + 1;
    {пропуск ведущих пробелов}
    while (i <= length(s))
    and (s[i] <> ' ') do i := i + 1;
    {пропуск фамилии}
    while (i <= length(s))
    and (s[i] = ' ') do i := i + 1;
    {пропуск межсловных пробелов}
    while (i <= length(s))
    and (s[i] <> ' ') do
    begin
      slovo := slovo + s[i];
      i := i + 1
    end;
    if uslov(slovo) then writeln(outf, s)
  end;
  close(inf);
  close(outf)
end.
```

Шаг пятый. Выводы

Рассмотренные задачи, с одной стороны, являются не слишком простыми для школьников. Они требуют умения выполнять декомпозицию задачи, требуют владения определенными тех-

ническими навыками программирования. С другой стороны, на этих задачах можно достаточно просто показать, что знания основных законов алгебры логики помогут, во-первых, в принципе решить такую задачу правильно, а во-вторых, помогут написать более простую и короткую программу. А, как известно, чем проще программа, тем легче ее отладить. Кстати, аналогичные задачи можно давать при изучении БД (построение сложных условий поиска в БД).

Предлагаемая система задач удовлетворяет таким требованиям развивающего обучения, как (1) принцип дидактической спирали, (2) ведущая роль теоретических знаний, (3) от простого к сложному. В результате использования такой системы задач у школьников подсознательно закрепляется навык использования теоретических знаний, причем не только на уроках информатики. Это как раз тот "багаж", который школьник уносит во "взрослую" жизнь. Несколько лет тому назад моя дочь — чистый гуманитарий — сказала: «Мое "домашнее математическое образование" позволяет мне понять, что когда несколько молодых людей говорят о матрице, это не значит, что они обсуждают фильм "Матрица"».

А если говорить конкретно о законах алгебры логики, умении выполнять формальные тождественные преобразования, то, конечно же, если эти навыки не тренировать, они со временем теряются. Но не теряются знания, их в нужный момент можно "достать". И если взрослому человеку, изучавшему когда-то давно в школе алгебру логики, придется помогать своему сыну в решении задачи про принцесс и тигров, то скорее всего он хотя бы будет понимать, о чем идет речь. А это уже немало! Не исключено также, что именно алгебра логики позволит разобраться в сложных условиях кредитного договора или банковского вклада. А с этим столкнуться придется уже большинству из нынешних учеников.

Задачи на обработку массивов, строк и файлов, требующие умения выполнять тождественные логические преобразования

1. Задачи на обработку массивов

В приведенных условиях задач формулировки логических высказываний позаимствованы с сайта Константина Юрьевича Полякова http://kpolyakov.narod.ru/. Полный список задач к статье содержится на диске к этому номеру

Задача 1. В символьном массиве длины n через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: ¬(первая буква согласная → последняя буква согласная) ∧ вторая буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
char ='Irina Stepan Ksenia Maria';
```

Задача 2. В символьном массиве длины n через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква согласная → вторая буква согласная) ∧ последняя буква гласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
char ='Maksim Stepan Ksenia
       Maria';
```

Задача 3. В символьном массиве длины n через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: ¬ (последняя буква гласная → первая буква согласная) ∧ вторая буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
         char ='Irina Stepan
         Artem Maria';
```

Задача 4. В символьном массиве длины n через пробел записаны слова латинскими буквами. Вывести на экран только те слова, которые удовлетворяют следующему условию: ¬ (первая буква согласная → (вторая буква согласная ∨ последняя буква гласная)). Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
     char ='Gore privet
     kreslo zakon';
```

2. Задачи на обработку массивов повышенной трудности

Задача 5. В символьном массиве длины *n* записаны имена латинскими буквами. Между словами, а также в начале массива и в конце его может быть *несколько* пробелов. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква согласная → вторая буква гласная) ∧ последняя буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char = 'Alisa Maksim Stepan Elena';
```

Задача 6. В символьном массиве длины *n* записаны имена латинскими буквами. Между словами, а также в начале массива и в конце его может быть *несколько* пробелов. Вывести на экран только те имена, которые удовлетворяют следующему условию: (вторая буква гласная → первая буква гласная) ∧ последняя буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Alisa Maksim Stepan Elena';
```

Задача 7. В символьном массиве длины *n* записаны слова латинскими буквами. Между словами, а также в начале массива и в конце его может быть *несколько* пробелов. Вывести на экран только те слова, которые не удовлетворяют следующему условию: (вторая буква гласная → предпоследняя буква согласная) ∧ первая буква стоит в алфавите раньше третьей. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Dunay Moskwa Dvina Wolga';
```

Задача 8. В символьном массиве длины *n* записаны слова латинскими буквами. Между словами, а также в начале массива и в конце его может быть *несколько* пробелов. Вывести на экран слова, которые удовлетворяют следующему условию: (вторая буква гласная → предпоследняя буква согласная) ∧ первая буква стоит в алфавите раньше третьей. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Dunay Moskwa Dvina Wolga';
```

Задача 9. В символьном массиве длины *n* через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: ¬(первая буква согласная → последняя буква согласная) ∧ вторая буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}. Дополнительный массив не использовать.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Irina Stepan Ksenia Maria';
```

Задача 10. В символьном массиве длины *n* через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: (первая буква согласная → вторая буква согласная) ∧ последняя буква гласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}. Дополнительный массив не использовать.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of char ='Maksim
Stepan Ksenia Maria';
```

Задача 11. В символьном массиве длины *n* через пробел записаны имена латинскими буквами. Вывести на экран только те имена, которые удовлетворяют следующему условию: ¬ (Последняя буква гласная → Первая буква согласная) ∧ Вторая буква согласная. Гласными буквами в английском алфавите являются {a, e, i, o, u, y}. Дополнительный массив не использовать.

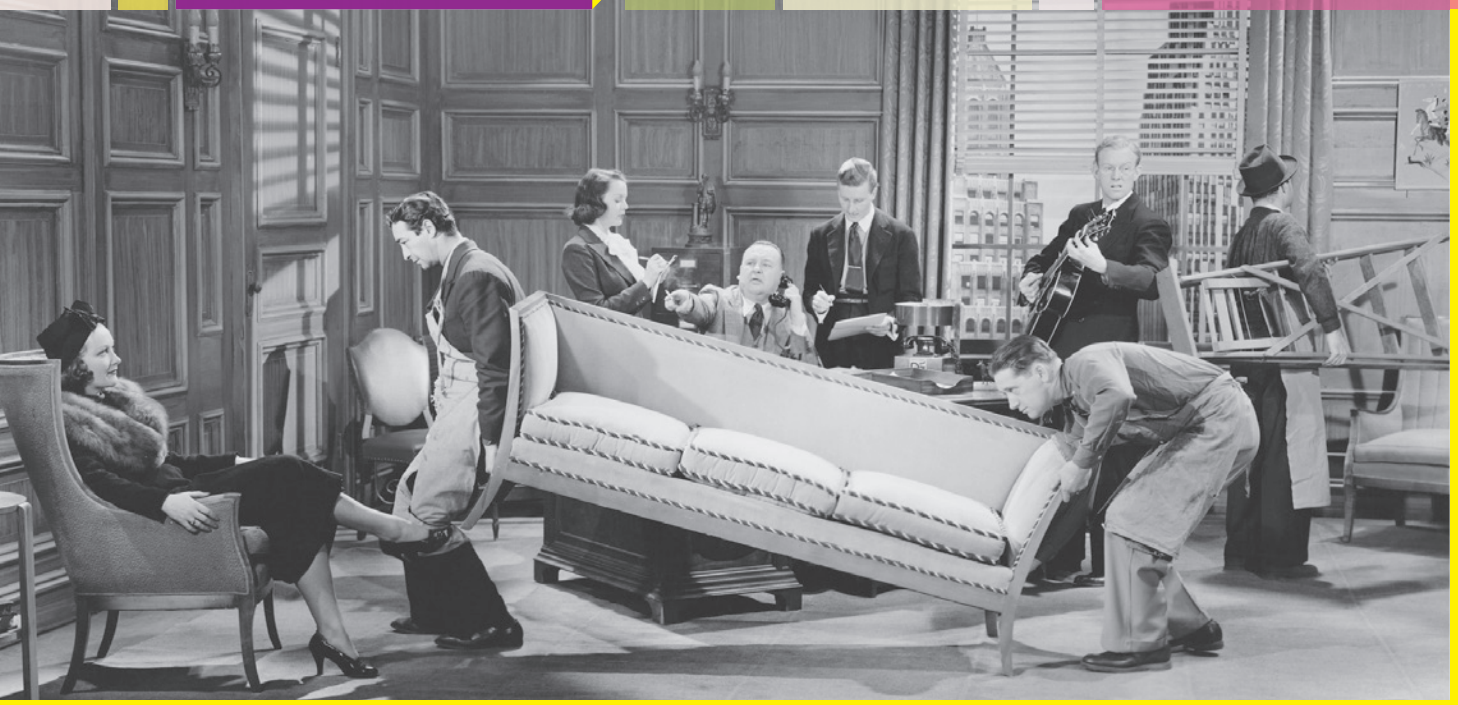
Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Irina Stepan Artem Maria';
```

Задача 12. В символьном массиве длины *n* через пробел записаны слова латинскими буквами. Вывести на экран только те слова, которые удовлетворяют следующему условию: ¬ (первая буква согласная → (вторая буква согласная ∨ последняя буква гласная)). Гласными буквами в английском алфавите являются {a, e, i, o, u, y}. Дополнительный массив не использовать.

Замечание. При отладке программы предложите школьникам такое содержание массива:

```
Const a: array [1..n] of
      char ='Gore privet kreslo zakon';
```



Нормализация баз данных

Ю.В. Пашковская,
учитель информатики
гимназии № 1,
г. Жуковский,
Московская обл.

Процедура нормализации считается самым сложным этапом в создании базы данных. Отчасти потому, что многое в ней совершается интуитивно. Существует множество нормальных форм, но проектировщику базы данных важно суметь найти компромисс между степенью нормализации и производительностью СУБД.

Эта статья родилась из стремления изложить достаточно сложный материал наиболее понятным языком, чтобы не только дать учащимся возможность освоить навыки пользователей конкретной СУБД, но и приблизить их к творчеству создателей баз данных.

Существует множество классификаций баз данных — по модели данных, по степени распределенности, по содержимому, по технологии хранения и т.д. Мы же рассмотрим классификацию баз данных по организации связей между ее объектами.

Виды структур данных

1. **Линейная** организация, или **список**. Она объединяет **не связанные** между собой однотипные элементы, собранные по какому-либо признаку. Пример — список учеников 10-го класса (см. *рис.* 1).

Однотипные элементы здесь — ученики. А признак, по которому они объединены, — принадлежность одному классу. Несвязанность их между собой означает, что удаление одного ученика из списка класса не влечет за собой удаление других.

10-й класс
Классный руководитель Антонова Наталья Васильевна

№ п/п	Ф.И.О. учащегося	Национальность	Год рождения
1	Григорян Ольга	армянка	21.05.1993
2	Эдильбаев Арслан	ногаец	11.03.1993
3	Магомедов Артур	лакец	19.04.1993
4	Ибиев Альберт	даргинец	12.09.1993

Рис. 1

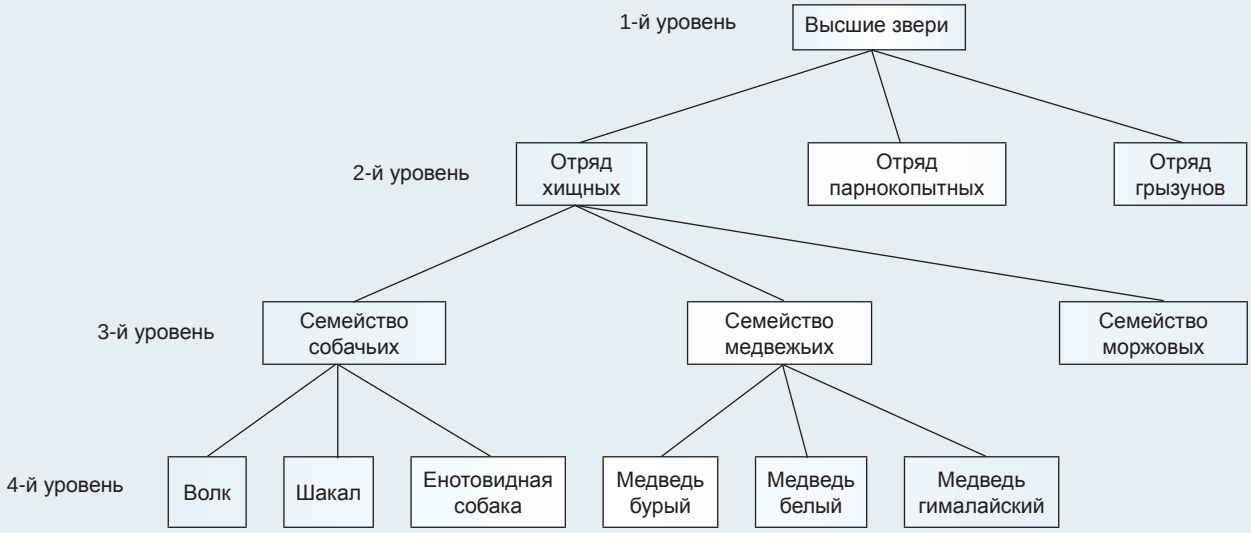


Рис. 2

Линейная организация является простейшей структурой базы данных, тем ее кирпичиком, из совокупности которых складываются более сложные структуры. (Освоить приемы работы со списком можно в Excel или Calc (OpenOffice).)

Следующие две структуры относятся к сложным.

2. Иерархическая организация, или многоуровневый список, или дерево. В ней каждый элемент нижнего уровня **связан только с одним** элементом верхнего. Таким образом, можно говорить о вложенности, подчиненности, наследовании элементов. Примером иерархической организации может служить классификация животных (см. *рис.* 2).

В иерархической организации количество уровней подчиненности соответствует количеству участвующих в ней классов объектов. На *рис.* 2 показаны связи между тремя классами объектов: отряды, семейства, роды.

Внутри иерархической организации можно выделить два вида списков. Списки первого вида назовем **горизонтальными**, поскольку образованы они из объектов, принадлежащих одному классу, и могут рассматриваться независимо друг от друга. В нашем примере к ним относятся списки *Отряды, Семейства, Роды*.

Отряды

№	Название
1	Хищные
2	Парнокопытные
3	Грызуны

Семейства

№	Название
1	Собаки
2	Медведи
3	Моржи

Роды

№	Название
1	Волк
2	Шакал
3	Енотовидная собака
4	Медведь бурый
5	Медведь белый
6	Медведь гималайский

Рис. 3

Список второго вида может быть назван **вертикальным**, поскольку представляет собой **совокупность связей**, каждая из которых соединяет один элемент нижнего уровня с корнем дерева через промежуточные элементы. Количество элементов вертикального списка равно числу элементов нижнего уровня дерева.

Связи

№	Род	Семейство	Отряд
1	Волк	Собаки	Хищные
2	Шакал	Собаки	Хищные
3	Енотовидная собака	Собаки	Хищные
4	Медведь бурый	Медведи	Хищные
5	Медведь белый	Медведи	Хищные
6	Медведь гималайский	Медведи	Хищные

Рис. 4

Очевидно, что в иерархической базе данных вертикальный список объектов является расширением горизонтального списка для самого нижнего уровня.

3. **Сетевая** организация — набор объектов двух разных классов, в котором **каждый** объект одного класса **может быть связан с каждым** объектом другого класса. Примером может служить граф, иллюстрирующий денежные вклады граждан в разные банки:

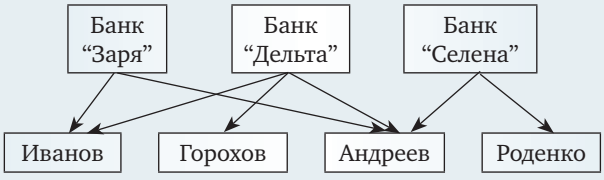


Рис. 5

Здесь можно выделить два класса объектов — *Банки* и *Вкладчики*, каждый из которых является по своей организации **горизонтальным** списком.

Вкладчики

№	Фамилия
1	Иванов
2	Горохов
3	Андреев
4	Роденко

Банки

№	Банк
1	Заря
2	Дельта
3	Селена

Рис. 6

Так же, как в иерархической структуре, в сетевой есть и **вертикальный** список, отражающий **связи** между объектами разных классов. В нашем примере это список вкладов всех вкладчиков. Если граф — взвешенный, то вес каждого ребра — еще одна характеристика связи между объектами разных классов, и она, так же как и другие характеристики связи, может быть отражена в вертикальном списке:

Вклады

№	Вкладчик	Банк	Сумма, руб.
1	Иванов	Заря	100 000
2	Иванов	Дельта	200 000
3	Горохов	Дельта	250 000
4	Андреев	Дельта	153 000
5	Андреев	Заря	200 000
6	Андреев	Селена	600 000
7	Роденко	Селена	450 000

Рис. 7

Элементы вертикального списка, как и полагается элементам линейной структуры, являются независимыми друг от друга. Это значит, что отзыв, к примеру, Ивановым своего вклада из банка “Заря” никак не повлияет на остальные вклады из списка.

В то же время, удаление любого элемента из горизонтального списка повлечет за собой удаление всех связей, в которые он вовлечен. Если обанкротится банк “Заря”, то потребуют удаления 1-й и 5-й элементы из списка *Вклады* (см. *рис.* 7). То же случится, если будет удален какой-либо элемент из списка *Вкладчики*.

Вообще говоря, вертикального списка достаточно, чтобы полностью описать структуру базы данных. Классы объектов базы данных можно восстановить по названиям столбцов, содержимое горизонтальных списков — по их уникальным значениям, а связи между элементами разных классов — по строкам. Таким образом, именно вертикальный список отражает главную идею базы данных.

***

Задание. Определить из текста, содержащего неструктурированную информацию, классы объектов. Представить ее в виде графа (иерархического или сетевого), а также составить вертикальный список.

Вариант 1. *“Поезд № 22 приезжает на станцию Клубникино в 12.20, а на станцию Земляникино — в 12.50. На станции Клубникино он стоит 10 минут и отправляется в 12.30. На станции Земляникино стоянка составляет 5 минут. Поезд № 26 приезжает на станцию Клубникино в 15.10, а на станцию Земляникино — в 16.00. На станции Клубникино стоянка составляет 7 минут, а на станции Земляникино — 13. Поезд № 30 приезжает на станцию Клубникино в 18.50, а на станцию Земляникино — в 19.05. Стоянка на станции Клубникино составляет 7 минут, а отъезд со станции Земляникино происходит в 19.08…”*

Решение. 1-й класс объектов — станции, 2-й класс объектов — поезда. Каждый элемент из 1-го класса связан с каждым элементом из 2-го класса такими характеристиками, как время прибытия, стоянки и отправления. Следовательно, это сетевая организация, которую можно изобразить так:

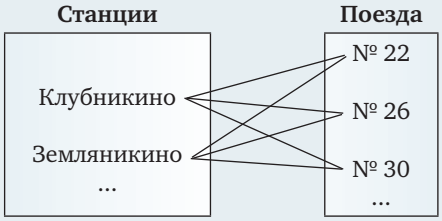


Рис. 8

Горизонтальные списки представляют собой перечни станций и поездов. А вертикальный — перечень остановок:

№ остановки	Поезд	Станция	Время прибытия	Стоянка (мин.)	Время отправления
1	22	Клубникино	12.20	10	12.30
2	22	Земляникино	12.50	5	12.55
3	26	Клубникино	15.10	7	15.17
4	26	Земляникино	16.00	13	16.13
5	30	Клубникино	18.50	7	18.57
6	30	Земляникино	19.05	3	19.08

Рис. 9

Независимость элементов этого списка выражается в том, что отмена одной остановки в маршруте какого-либо поезда никак не повлияет на существование остальных остановок.

В то же время, удаление какого-либо элемента из горизонтального списка (например, поезда № 26) повлечет за собой удаление всех связей, в которые он вовлечен (3-я и 4-я строки вертикального списка — *рис.* 9).

В теории баз данных связь, подобную связи горизонтального списка с вертикальным, называют отношением “**один-ко-многим**”. А обозначают так: “1 – ∞”.

Из этого примера видно, что связи в сетевой структуре (см. *рис.* 8) “**многие-ко-многим**” (∞ – ∞) между горизонтальными списками с помощью вертикального списка можно заменить на две связи “**один-ко-многим**”.

Вариант 2. *“Миша Иванов посещает танцевальные занятия, которые ведет преподаватель Сергеев по понедельникам и четвергам, в 15.00. Туда же приходят Саша Андреев и Валя Марченко. А вот их приятели Петя Карпов и Яна Молчанова занимаются танцами у другого преподавателя — Макаровой, по вторникам и пятницам. В эти же дни, но на два часа позже к Макаровой на занятия приходят дети помладше: Вова Иванов, брат Миши, Степа и Соня Емцевы”.*

Решение. 1-й класс объектов — дети, 2-й класс объектов — преподаватели. Каждый ребенок (объект из 1-го класса) посещает занятия только у одного преподавателя (объект 2-го класса). При этом у Макаровой ученики разбиты на младшую и старшую группы. Значит, можно выделить 3-й класс объектов — группы. Каждый ученик ходит в одну из групп. При этом ни один ученик не посещает сразу две группы. Налицо иерархические отношения между элементами:

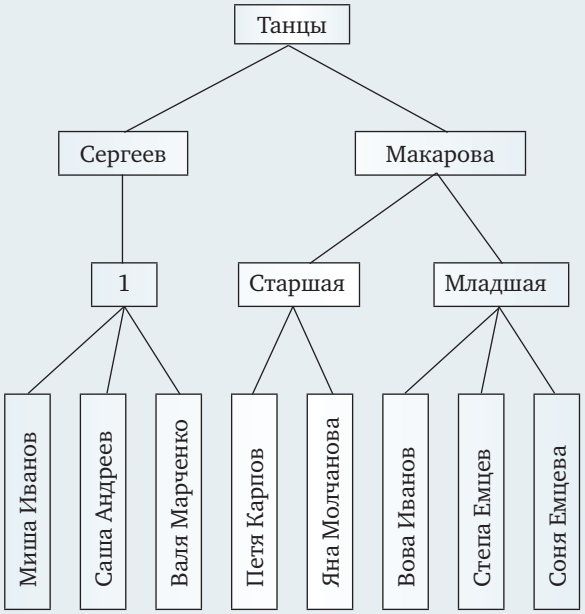


Рис. 10

Обратите внимание: чтобы всех детей, посещающих занятия, представить элементами одного уровня, мы добавили промежуточный элемент (1-я группа у преподавателя Сергеева), который не был упомянут в тексте, но не противоречит его смыслу.

Таким образом, горизонтальных списков здесь три:

Ученики

№	Фамилия, имя
1	Иванов Миша
2	Андреев Саша
3	Марченко Валя
4	Карпов Петя
5	Молчанова Яна
6	Иванов Вова
7	Емцев Степа
8	Емцева Соня

Преподаватели

№	Фамилия
1	Сергеев
2	Макарова

Группы

№	Название
1	Первая
2	Старшая
3	Младшая

Рис. 11

А вертикальный список (связи) представляет собой **распределение учеников по танцевальным группам**:

№	Ученик	Группа	Преподаватель
1	Иванов Миша	Первая	Сергеев
2	Андреев Саша	Первая	Сергеев
3	Марченко Валя	Первая	Сергеев
4	Карпов Петя	Старшая	Макарова
5	Молчанова Яна	Старшая	Макарова
6	Иванов Вова	Младшая	Макарова
7	Емцев Степа	Младшая	Макарова
8	Емцева Соня	Младшая	Макарова

Рис. 12

Поскольку вертикальный список является расширением горизонтального (в данном случае — списка *Ученики*), то удаление какого-либо элемента из соответствующего ему горизонтального списка влечет за собой удаление ровно одного элемента из вертикального. В этом случае говорят, что между этими списками существует связь “**один-к-одному**”, которая обозначается “1 – 1”.

Если два класса объектов связаны друг с другом соотношением “1 – 1”, то они могут рассматриваться как объект и атрибут. Следовательно, в иерархической базе данных элементы горизонтальных списков являются атрибутами связанного с ними элемента нижнего уровня.

Удаление же одного элемента из любого другого (не нижнего) горизонтального списка, вообще говоря, означает удаление нескольких элементов из вертикального списка. Например, увольнение преподавателя Серова отменяет занятия у всех учеников первой группы, т.е. у Иванова Миши, Андреева Саши и Марченко Вали. Аналогично, если младшую группу решат распустить, то из списка учеников будут исключены Иванов Вова и близнецы Емцевы. Значит, остальные горизонтальные списки связаны с вертикальным отношениями “1 – ∞”.

Из этих двух примеров видны основные различия между иерархической и сетевой структурами. В первой вертикальный список связан с горизонтальными одной связью “1 – 1” и несколькими “1 – ∞”. В сетевой структуре все горизонтальные списки находятся по отношению к вертикальному в связи “1 – ∞”.

Реляционные базы данных

Теперь пора перейти на терминологию баз данных. Заменим слово “список” на слово “таблица”. Таблица относится к **способу представления** данных и действительно является наиболее удобной формой представления списка.

Таблица состоит из строк и полей. Строки в базах данных принято называть **записями**, а столбцы — **полями**. Записи описывают элементы списка, или **объекты**, а поля — их характеристики, или **атрибуты**.

Атрибуты делятся на параметры, признаки и свойства. **Параметры** отражают количественные характеристики объекта (числовой тип или тип дата/время). **Признаки** описывают качественные характеристики объекта (текстовый тип). **Свойства** обозначают способность объекта получить некоторый результат (логический тип).

Чтобы отличать одну запись от другой, в таблицах вводится понятие **ключа**. Так называют поле или комбинацию полей, по которым одну запись в таблице можно отличить от другой. В наших предыдущих примерах в качестве ключа можно использовать порядковый номер в списках.

База данных, представленная в виде набора связанных между собой простых таблиц, называется **реляционной**. Поскольку, как показано выше, любая информационная структура представима в виде набора списков, то реляционные базы данных являются универсальными.

***

Вернемся теперь к вертикальному списку из первого примера (*рис.* 4). Как уже было сказано, вертикальный список содержит полную информацию о структуре базы данных. Однако, как видно из таблицы, в нем много повторов. А это значит, что информация занимает в компьютерной памяти излишне много места. Кроме того, в базе данных могут возникнуть так называемые “аномалии обновления”. Это значит, что если, например, пользователями базы будет решено заменить русское название отряда *Хищные* на латинское — *Carnivora*, то вносить исправления придется во все записи, где встречается это название. Кроме того, если два разных пользователя одной базы данных назовут одно и то же семейство по-разному (например, *Собачьи* и *Псовые*), то компьютер воспримет эти семейства как разные.

Кроме аномалий обновления, существуют еще аномалии добавления и удаления. Они возникают в сетевых структурах, если в качестве атрибута в вертикальном списке берется атрибут(-ы) какого-либо объекта из горизонтального списка.

Например, если в базу данных “Спортсмены”

Спортсмены

№ спортсмена	Ф.И.О.	Код спортивной секции	Название секции
1	Иванов С.И.	1	Плавание
2	Петров П.К.	2	Легкая атлетика
3	Сидоров А.К.	1	Плавание

добавят следующую запись:

4	Косых Н.В.	1	Теннис

то возникнет противоречие: код секции не будет совпадать с названием этой секции (аномалия добавления).

Если же из секции *Плавание* решат уйти все спортсмены, то из базы данных исчезнет и вся информация об этой секции (аномалия удаления).

Очевидно, что *Название секции* является характеристикой *Секции*, т.е. атрибутом соответствующего ей горизонтального списка:

Спортивные секции

Код спортивной секции	Название секции
1	Плавание
2	Легкая атлетика
3	Теннис

Чтобы избавиться от всевозможных аномалий при работе с информацией, проводят **нормализацию** базы данных.

Нормализация — это процедура, которая позволяет устранить дублирование и потенциальную противоречивость хранимых данных.

В идеале при нормализации надо добиться того, чтобы любое значение хранилось в единственном экземпляре, причем значение это не должно быть получено расчетным путем из других данных, хранящихся в базе.

Первый шаг к нормализации БД — это избавление от повторов. Для этого воспользуемся тем, что таблица вертикального списка связана с таблицами горизонтальных списков отношениями “1 – ∞” и “1 – 1”. Это значит, что мы можем заменить значения всех полей в вертикальном списке на соответствующий им ключ в горизонтальных списках. В нашем примере вертикальный список получит вид:

№	Род	Семейство	Отряд
1	Волк	1	1
2	Шакал	1	1
3	Енотовидная собака	1	1
4	Медведь бурый	2	1
5	Медведь белый	2	1
6	Медведь гималайский	2	1

Рис. 13

Но теперь его надо дополнить горизонтальными списками с указанием связей:

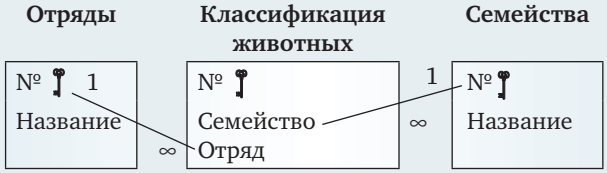


Рис. 14

Проделав аналогичную процедуру с базой данных сетевой структуры (см. *рис.* 7), вертикальный список приобретет вид:

Вклады

№ вклада	Вкладчик	Банк	Сумма, руб.
1	1	1	100 000
2	1	2	200 000
3	2	2	250 000
4	3	2	153 000
5	3	1	200 000
6	3	3	600 000
7	4	3	450 000

Рис. 15

А структура базы данных выразится в следующем наборе таблиц:

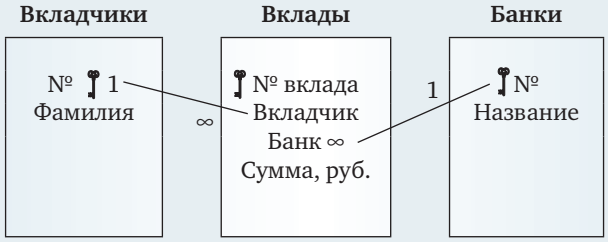


Рис. 16

***

Дополнительными требованиями к нормализованной базе данных являются следующие:

1. Любое поле должно быть неделимым. Например, поле **Ф.И.О.** следует разделить на **Фамилия**, **Имя** и **Отчество**, если перед пользователями базы данных может стоять задача поиска людей по отчеству или имени. (Если такой задачи нет, то поле Ф.И.О. будет восприниматься СУБД как неделимое.)

Фамилия и имя
Иванов Петр
Петров Иван
…

→

Фамилия	Имя
Иванов	Петр
Петров	Иван
…	…

Рис. 17

Нарушение этого требования может привести к так называемым **аномалиям удаления** и **добавления**. Например, если в БД “Ученик” будет решено избавиться от отчества и тем самым сократить объем базы, то выудить отчество из поля **Ф.И.О.** во всех записях гораздо более затратно, чем просто удалить из структуры БД поле **Отчество**.

2. Не должно быть полей, которые могут быть найдены с помощью остальных. Нарушение этого требования приводит к неоправданному увеличению объема базы данных. Например, в таблице, отражающей сведения о хранящихся на продуктовой базе фруктах, не должно быть поля **Стоимость**, значение которого может быть получено как произведение значений полей **Цена** и **Количество**.

Товар	Цена за тонну	Кол-во, тонн	Стоимость
Бананы	1200	10	12 000
Киви	1500	20	30 000
…			

Рис. 18

3. Не должно быть полей, которые обозначают различные виды одного и того же:

№	Студент 1	Студент 2	Студент 3
1	Иванов И.И.	Петров П.П.	Сидоров С.С.

Рис. 19

Такая ситуация может возникнуть в том случае, если в сетевой базе данных классы первых и вторых объектов совпадают. Например, на шахматном турнире каждый участник соревнования должен сразиться с остальными. Логично было бы составить вертикальный список “Игры” следующим образом:

№ игры	1-й игрок	2-й игрок	Результат

Но тогда нарушалась бы однозначность связей между таблицами *Игры* и *Игроки*:

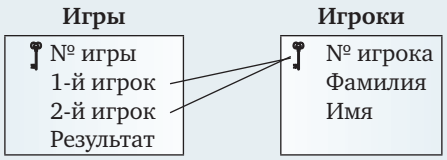


Рис. 20

Поскольку очевидно, что в подобных случаях связь между таблицами сетевая (несколько игроков одновременно принимают участие в одной игре, и один игрок может участвовать в нескольких играх), используют третью таблицу “Участие в играх”:

№ участия	№ игры	Игрок	Результат

Здесь связь двух игроков, участвующих в одной и той же игре, осуществляется по общему значению в поле “№ игры”.

***

Теперь можно полностью сформулировать алгоритм нормализации базы данных:

1. Определение классов объектов, входящих в ее состав.

2. Замена связи “1 – 1” объединенным списком, в котором объект одного класса выступает в качестве атрибута для объекта другого класса.

3. Определение структуры связи между классами.

4. Замена каждой сложной структуры на вертикальный и горизонтальные списки с указанием связей по ключевым полям.

5. Внесение дополнительных атрибутов для каждого класса объектов — в зависимости от степени детализации объектов.

6. Обеспечение требования неделимости полей — в зависимости от целей, стоящих перед будущими пользователями базы данных.

7. Исключение вычисляемых полей.

Следование этому алгоритму обеспечивает выполнение еще одного требования к нормализованной базе данных: **каждое неключевое поле должно однозначно зависеть от ключа**.

Пример задачи. Нормализовать базу данных, заданную схемой *рис*. 21.

Решение

1. Определим классы объектов из представленных на схеме категорий: *врачи, пациенты, кабинеты, годы рождения, адреса, время.*

2. Тип связи “*врачи – кабинеты*” определяется отношением “1 – 1” — значит, номер кабинета можно считать атрибутом врача (и наоборот). Точно те же рассуждения применимы к связи “*пациент – год – адрес*”. Будем считать *год* и *адрес* атрибутами *пациентов*. Таким образом, остается два класса объектов: *врачи* и *пациенты*.

3. Определим статус категории *время*. Видно, что время не является ни атрибутом *пациентов*, ни атрибутом *врачей*. Однако оно характеризует связь между ними и, следовательно, относится к атрибутам связи.

4. Определяем структуру между классами. Одному врачу соответствует несколько пациентов. Один и тот же пациент — например, Кузин — записан к нескольким врачам. Значит, структура — сетевая.

5. Составляем горизонтальные и вертикальный списки и связи между ними:

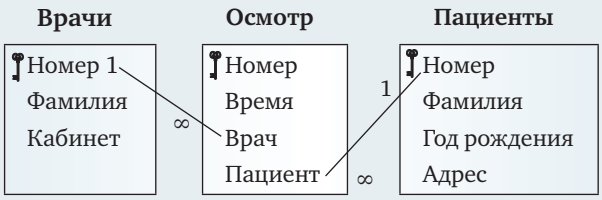


Рис. 22

Выполнение пунктов 5–7 не требуется. Задача решена.

***

Несколько сложнее обстоит дело с обратной задачей: по заданному списку нормализовать базу данных. Для этого надо восстановить ее структуру, т.е. определить, какие поля списка соответствуют классам объектов, какие — их атрибутам и какого типа — иерархические или сетевые — связи между объектами разных классов.

Рассмотрим следующий пример. Дана таблица, содержащая сведения об учебной литературе.

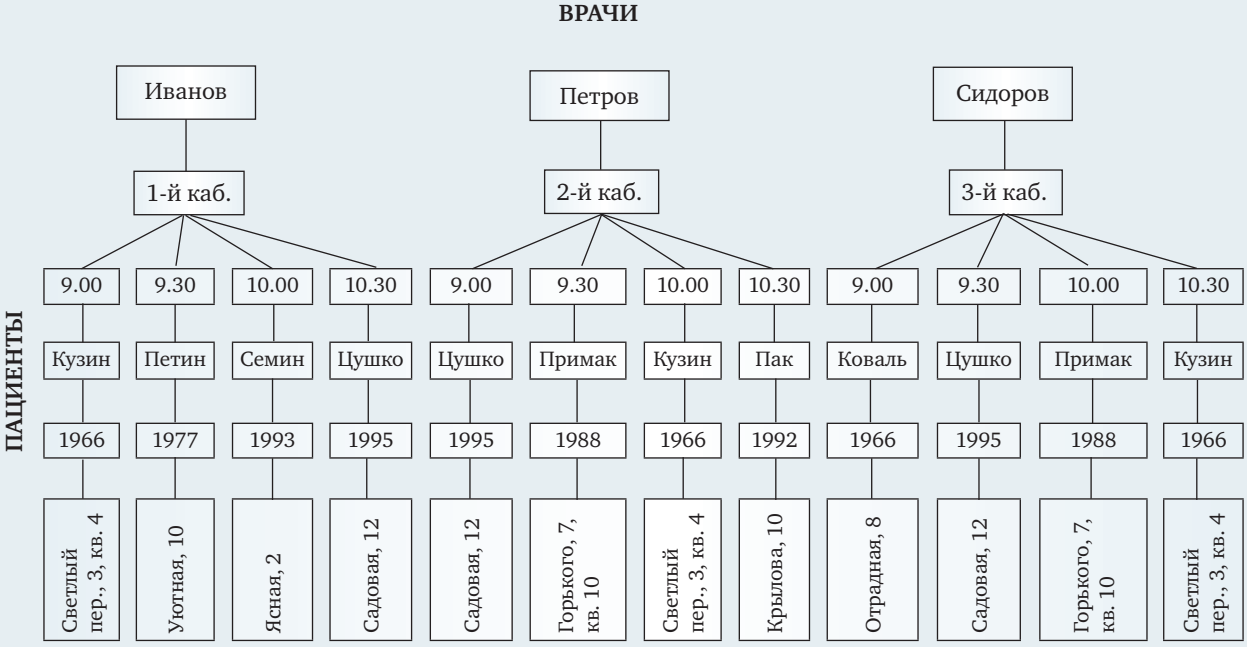


Рис. 21

ID	Код	Тема	Автор	Название	Тип	Год	Стр.
20	22.18	МК	Бочков С. Субботин Д.	Язык программирования Си	Учебник	1990	384
10	22.18	МК	Джехани Н.	Язык Ада	Учебник	1988	552
35	32.97	ВТ	Соловьев Г. Никитин В.	Операционные системы ЭВМ	Учебное пособие	1992	208
11	32.81	Кибернетика	Попов Э.В.	Общение с ЭВМ на естественном языке	Учебник	1982	360
44	32.97	ВТ		ПУ для ПЭВМ	Справочник	1992	208
89	32.973	ЭВМ	Коутс Р. Влейминк И.	Интерфейс “человек – компьютер”	Учебник	1990	501

Рис. 23

Требуется нормализовать соответствующую ей базу данных.

Решение:

1. Определим классы, связанные соотношением “1 – 1”. Это все поля, за исключением поля *автор*:

ID	Код	Тема	Название	Тип	Год	Стр.
20	22.18	МК	Язык программирования Си	Учебник	1990	384
10	22.18	МК	Язык Ада	Учебник	1988	552
35	32.97	ВТ	Операционные системы ЭВМ	Учебное пособие	1992	208
11	32.81	Кибернетика	Общение с ЭВМ на естественном языке	Учебник	1982	360
44	32.97	ВТ	ПУ для ПЭВМ	Справочник	1992	208
89	32.973	ЭВМ	Интерфейс “человек – компьютер”	Учебник	1990	501

Рис. 24

Это значит, что каждая запись описывает один объект — *книгу*. Это — первый класс объектов.

2. Объекты других классов выдают повторения. Это *коды, темы и типы*. Вынесем их в отдельные списки:

Коды

ID	Код
1	22.18
2	32.97
3	32.973
4	32.81

Темы

ID	Тема
1	МК
2	ВТ
3	ЭВМ
4	Кибернетика

Вид издания

ID	Тип
1	Учебник
2	Учебное пособие
3	Справочник

Рис. 25

Обратим внимание на то, что коды и темы входят в исходную таблицу, образуя связь “1 – 1”. Это значит, что *тему* можно считать атрибутом *кода*. А поскольку сам *код* представляет собой неповторяющиеся числа, то его можно считать ключом:

Рубрикатор

Код	Тема
22.18	МК
32.97	ВТ
32.973	ЭВМ
32.81	Кибернетика

Рис. 26

3. Теперь список *книг* будет выглядеть так:

Книги

ID	Код	Название	Тип	Год	Стр.
20	22.18	Язык программирования Си	1	1990	384
10	22.18	Язык Ада	1	1988	552
35	32.97	Операционные системы ЭВМ	2	1992	208
11	32.81	Общение с ЭВМ на естественном языке	1	1982	360
44	32.97	ПУ для ПЭВМ	3	1992	208
89	32.973	Интерфейс “человек – компьютер”	1	1990	501

Рис. 27

А структура связей между классами *книги – рубрикатор – темы* так:

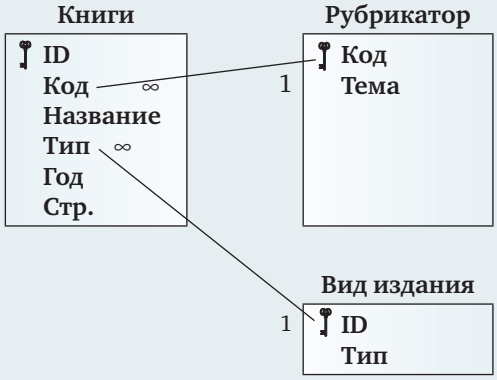


Рис. 28

4. Определим тип связи между объектами классов *книги* и *авторы*: одна книга может быть написана несколькими авторами; один автор может написать несколько книг. Значит, структура связи сетевая.

5. Следовательно, исходная таблица преобразуется в таблицу, отражающую **связи** между книгами и авторами, т.е. публикации:

Публикации

ID	Книга	Автор
1	20	Бочков С.
2	20	Субботин Д.
3	10	Джехани Н.
4	30	Крон Г.
5	87	Гик Е.Я.
6	38	Фролов Г.
7	38	Кузнецов Э.

Рис. 29

Одновременно мы выполнили требование неделимости для поля *Автор*.

Общая структура базы данных будет выглядеть так:

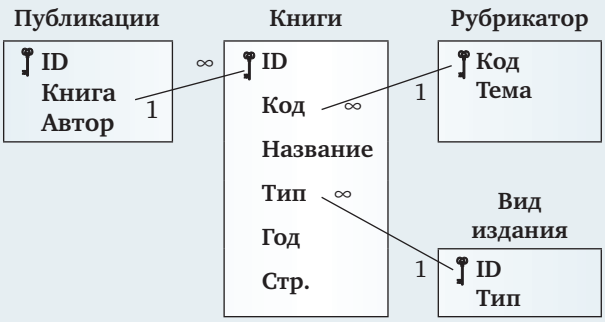


Рис. 30

Приложение в электронном виде: самостоятельные работы по вариантам

1. Презентация “Базы данных”: файл “Нормализация БД.ppt”.

2. Самостоятельная работа по вариантам “Сведение произвольной структуры к табличному виду”: файл “Сведение произвольной структуры к табличному виду.doc”. Дополнительные задачи по этой теме можно взять из [2], с. 114–118, изменив условие следующим образом: представить данные в виде набора простых таблиц.

3. Самостоятельная работа по вариантам “Создание БД”: файл “Создание БД.doc”. Работа подразумевает знакомство с СУБД OpenOfficeBase. Работа с таблицей в двух режимах — дизайнера и пользователя.

4. Самостоятельная работа по вариантам “Усложнение БД: добавление в БД новых таблиц”: файл “Усложнение БД.doc”.

5. Самостоятельная работа по вариантам “Нормализация БД, представленных таблицей или схемой”: файл “Сам. работа — нормализация БД.doc”.

6. Самостоятельная работа по вариантам “Проектирование БД: по описанию данных, которые должны быть отражены в базе данных, спроектировать многотабличную БД”: файл “Проектирование БД.doc”.

Литература

1. Информатика и ИКТ. Практикум, 8–9-е классы / под ред. Н.В. Макаровой. СПб.: Питер, 2010.

2. Информатика и ИКТ. Задачник-практикум. В 2 т. / под ред. И.Г. Семакина, Е.К. Хеннера, 3-е изд. М.: Бином. Лаборатория знаний, т. 1, 2011.

3. Информатика и ИКТ. Задачник-практикум. В 2 т. / под ред. И.Г. Семакина, Е.К. Хеннера, 3-е изд. М.: Бином. Лаборатория знаний, т. 2, 2011.

годовая подшивка журнала **«ИНФОРМАТИКА»**

ПОЛНАЯ ПОДБОРКА БУМАЖНОЙ ВЕРСИИ ЖУРНАЛА НА КОМПАКТ-ДИСКЕ ЗА **2011** ГОД

=

Цена с доставкой – 699 РУБ.

Вы можете оформить заказ:

- на сайте **shop.1september.ru**
- по электронной почте **podpiska@1september.ru**
- по телефону **(499) 249–47–58**

Также можно приобрести:

ПОДШИВКУ ИЗДАНИЯ НА КОМПАКТ-ДИСКЕ ЗА 2010 ГОД

Цена диска с доставкой – 499 РУБ.

ВИДЕОЗАПИСИ ЛЕКЦИЙ И МАСТЕР-КЛАССОВ ВЕДУЩИХ ПРЕПОДАВАТЕЛЕЙ

Цена с доставкой – от 299 РУБ.

Цены действительны до 30 июня 2012 года.
Рассылка производится только на территории РФ.

Регулярно выражаясь

И.А. Сукин,
г. Переславль-Залесский

Немного истории

Задача обработки текстовой информации поставлена и решается в информатике и программировании с самого зарождения этой отрасли. Еще в сороковых годах прошлого века проводились теоретические исследования на эту тему, которые были затем воплощены на вычислительных машинах. В частности, эти исследования затрагивали теорию автоматов, где заметной фигурой был Джон фон Нейман, и теорию формальных языков, в которой признанным лидером в то время был Ноам Хомский с его теорией и классификацией грамматик. Позже единая наука обработки текстов при реализации на ЭВМ, как это обычно происходит, разделилась на несколько путей.

В начале 60-х годов XX века на сцену вышел один из первых языков высокого уровня, нацеленных именно на обработку текстовой информации, — Snobol. В отличие от всех своих предшественников он был языком промышленного качества и удовлетворял самым высоким требованиям своего времени, успешно используясь многими компаниями. В дальнейшем Snobol претерпел множество изменений, которые вылились в языки Icon и Unicon, введя мощную концепцию “генераторов”. К сожалению, в силу распространенности в основном в университетской среде, эти возможности не получили широкой огласки и сейчас существуют только в виде академического проекта. Вторым вариантом был Рефал — функциональный язык, использующий технику “сравнения с образцом”, не нацеленный исключительно на обработку текстов, но создававшийся с учетом возможности применения в решении проблем искусственного интеллекта и метапрограммирования, автоматически подразумевающих данную подзадачу. Рефал изначально был чисто академическим языком, поэтому его популярность была и остается не очень высокой. Третьим важнейшим подходом было создание математиком Стивеном Клини языковой математической концепции, названной “ре-

гулярными множествами". Эта концепция была прообразом современных регулярных выражений. Один из основоположников Unix Кен Томпсон в конце 60-х реализовал "регулярные множества" в своем текстовом редакторе QED и в стандартном текстовом редакторе ed. С тех пор регулярные выражения прочно закрепились в качестве индустриального стандарта, активно применяясь при работе с Unix-системами.

Однако долгое время все эти возможности, новшества и концепции оставались уделом "бородатых гуру". Второе рождение регулярные выражения получили в конце 80-х годов прошлого века вместе с созданием и распространением языка Perl, который на долгое время определил свой стандарт обработки строковых данных. С тех пор практически для любого языка программирования есть своя реализация PCRE (***P**erl **C**ompatible **R**egular **E**xpressions*) или BRE/ERE (***B**asic **R**egular **E**xpressions/**E**xtended **R**egular **E**xpressions*, традиционные регулярные выражения Unix, теперь определенные в стандарте POSIX). Регулярные выражения вошли даже в жизнь обычных пользователей персональных компьютеров: многие текстовые редакторы и процессоры, как потоковые, так и более традиционные, предлагают их в качестве средства для поиска по тексту. Вероятно, в не таком уж далеком будущем умение составлять и использовать регулярные выражения станет частью компьютерной грамотности.

В заключение исторического экскурса хотелось бы отдельно отметить библиотеку Oniguruma и новую версию Perl — Perl6. Эти технологии являются самым современным развитием регулярных выражений, добавляя новые возможности в классический вариант.

Зачем мне регулярные выражения?

Если вы раньше никогда не пользовались аппаратом регулярных выражений, то вы можете задаться вопросом: "А зачем мне, собственно, разбираться в новом, не слишком понятном для меня языке?". Ответ на этот вопрос прост и очевиден: регулярные выражения предоставляют мощные средства поиска и редактирования текстов, с их помощью можно описать очень сложные шаблоны поиска. Классическим примером является, например, вычленение IP-адреса из журнала. Если же вы самый обычный пользователь ПК, то регулярные выражения вам понадобятся, конечно, гораздо реже, однако можно привести пример запроса, требующего комплексного решения: "Найти все включения слова "шарик", перед которым стоит слово "желтый" или слово "красный", так чтобы слово "шарик" находилось в конце строки". В таких случаях не обойтись без использования регулярных выражений.

Программистам же просто жизненно необходимо знать основные принципы обработки строк, потому что так или иначе каждый в своей карьере сталкивается с разбором текстовых данных. Ну и безоговорочно регулярные выражения нужны разработчикам компиляторов, интерпретаторов и диалоговых интерфейсов пользователя, а также исследователям в области искусственного интеллекта.

Так или иначе, знание и умение использовать регулярные выражения пригодится любому человеку, использующему в своей жизни вычислительную технику, ведь, учитывая огромные темпы роста количества текстовой информации, средства, позволяющие найти в ней нужные именно тебе сведения, ценятся на вес золота.

Что же такое регулярное выражение?

Регулярное выражение — это строка. Да, это самая обыкновенная строка, просто сформированная особым образом и содержащая особые последовательности символов. Такую строку принято называть *шаблоном*. Все виды регулярных выражений во всех возможных языках программирования и программах выглядят примерно одинаково. Можно выделить следующие составляющие любого регулярного выражения:

- *Обычные символы.* Обычные, чаще всего печатные символы. Чаще всего это буквы, цифры или знаки пунктуации. Необходимы для задания значащих частей шаблона. В вышеприведенном примере слова "шарик", "желтый" и "красный" — это значащие части.
- *Специальные символы.* Управляющие символы, определяющие структуру шаблона и результат сопоставления. В них входят:
  - *Якоря,* служащие для обозначения конкретного места в сопоставляемой строке, например, "начало строки" или "конец строки";
  - *Кванторы,* необходимые для количественных сопоставлений шаблона;
  - *Заглядывания, ссылки, подвыражения* и другие управляющие последовательности, необходимые для более тонкого управления поиском;
  - *Классы символов*, обозначения для целых множеств символов, необходимые для более компактного их описания. Например, класс всех букв, класс всех цифр, класс всех печатных символов;
- *Модификаторы,* определяющие некоторое общее поведение алгоритма сопоставления, например, игнорирование регистра. Модификаторы часто бывают зависимыми от реализации механизма регулярных выражений.

В разных реализациях вышеописанные символы могут выглядеть по-разному. Обычно для записи регулярных выражений используют так называемые "сырые" строки, которые содержат именно те символы, которые написал автор, не экранируя специальные символы, такие, как перевод строки "\n", звонок "\a", табуляцию "\t", и другие. Это связано с тем, что синтаксис регулярных выражений содер-

жит множество символов вроде бэкслеша и другой пунктуации, следовательно, можно избежать лишнего экранирования. Однако бывают случаи, когда это не так, и в следующем разделе мы это увидим.

Вы уже знаете все общие сведения о построении регулярных выражений, поэтому сейчас самое время перейти к конкретике.

Простейшие примеры: OpenOffice Writer и GNU sed

Классический синтаксис

Многие программы используют классический POSIX-синтаксис для регулярных выражений с некоторыми расширениями, о которых я упомяну отдельно. Согласно вышеприведенной классификации, все операции разделяются на следующие виды:

Обычные символы.

При этом символы, используемые в других элементах синтаксиса, надо экранировать (например, * записывается как *).

- Кроме этого, символ “.” сопоставляется с любым другим символом, включая символ перевода строки (в однострочном режиме).
- Escape-последовательности или экранированные символы используются для обозначения символов, имеющих специальное назначение или не имеющих графического/печатного представления. Большинство из них знакомо людям, знающим язык C:
  - \a, \f, \n, \r, \t, \v;
  - \cX — то же, что и Ctrl-<X>;
  - \dXXX — символ с десятичным кодом XXX;
  - \oXXX — символ с восьмеричным кодом XXX;
  - \xXX — символ с шестнадцатеричным кодом XX;
  - \w — любой символ, который может входить в состав “слова”, — буква, цифра или подчеркивание;
  - \W — любой символ, который не может входить в состав “слова”;
  - \b — граница “слова”, сопоставляется с пустой строкой, сбоку от которой находится символ, входящий в состав “слова”;
  - \B — не-граница “слова”, сопоставляется с пустой строкой, по обоим бокам которой находятся символы, входящие в состав “слова”.

Кванторы:

- * — сопоставляется с любым (в том числе нулевым) количеством повторений предыдущего регулярного выражения. Например, “a*” сопоставляется с “ ”, “a”, “aa”, “aaa” и так далее. При этом выбирается самое длинное сопоставление (жадный алгоритм);
- + — сопоставляется с любым положительным количеством повторений предыдущего регулярного выражения. В других реализациях может экранироваться;
- ? — сопоставляется с нулевым или единичным вхождением предыдущего регулярного выражения. Так, “a?” сопоставляется с “ ” или “a”;
- {*i*} — сопоставляется с ровно *i* вхождениями предыдущего выражения;
- {*i*,*j*} — сопоставляется с количеством вхождений предыдущего выражения от *i* до *j* (включительно);
- {*i*,} — сопоставляется с количеством вхождений от *i* и больше.

Якоря:

- ^ — сопоставляется с пустой строкой, находящейся в начале сопоставляемой строки, следовательно, этот якорь можно использовать только в начале регулярного выражения;
- $ — сопоставляется с пустой строкой, находящейся в конце сопоставляемой;
- \`, \' — конец и начало сопоставляемого текста при многострочном режиме, в котором ^ и $ сопоставляются с концом и началом каждой отдельной строки.

Управляющие конструкции:

- подвыражение1|подвыражение2 — сопоставляется либо с подвыражением1, либо с подвыражением2, причем сопоставление производится слева направо;
- подвыражение1подвыражение2 — сопоставляется с конкатенацией двух подвыражений.

Подвыражения и ссылки:

- (выражение) — самые обычные скобки. Необходимы для группировки частей регулярного выражения, например: “abcd*” сопоставляется со строками “abc”, “abcd”, “abcdd”, в то время как “(abcd)*”, сопоставляется с “ ”, “abcd”, “abcdabcd” и т.д.;
- \цифра — так называемая “обратная ссылка”. Позволяет сослаться на результат сопоставления подвыражения, идущего в целом выражении под определенным порядковым номером. Например, “(a)\1” сопоставляется с “aa”.

Классы символов:

- [символы] — канонический пример определения класса символов, сопоставляется с любым символом из списка, при этом особое внимание надо обратить на то, что пробелы тоже считаются включенными в класс, если вы вдруг неосмотрительно поставите их между символами. Пример: [aeiou] сопоставляется с любой гласной английского алфавита;
- [^символы] — отрицание класса, сопоставляется с любым символом, не входящим в класс. Важно заметить, что [^aeiou] сопоставляется не с любой согласной, а с любыми символами в рамках текущей кодировки, кроме английских гласных. Также стоит сказать, что символы “*”, “+”, “.”, “[” и “\” внутри описаний классов не нуждаются в экранировании.
- Предопределенные множества символов:
  - [:alnum:] — цифры или буквы;
  - [:alpha:] — буквы;

- [:blank:] — пробел или табуляция;
- [:cntrl:] — управляющие символы;
- [:digit:] — цифры;
- [:graph:] — “видимые” символы, имеющие графическое начертание (пробел сюда не входит);
- [:lower:] — буквы нижнего регистра;
- [:print:] — печатаемые символы (а сюда пробел уже входит);
- [:punct:] — символы пунктуации;
- [:space:] — все виды пробелов;
- [:upper:] — буквы верхнего регистра;
- [:xdigit:] — шестнадцатеричные цифры.

Использовать предопределенные классы символов нужно с осторожностью, потому что в некоторых версиях они могут не включать в себя символы языков, отличных от английского. Однако большинство современных реализаций регулярных выражений во всех популярных дистрибутивах GNU/Linux и под Windows отлично понимают кириллицу.

Существуют также особые классы “склеивания” и “эквивалентности” символов, однако мы не будем на них подробно останавливаться, поскольку ни в английском, ни в русском языке они особо не нужны.

Кроме всего прочего, существуют также следующие общепринятые модификаторы, управляющие поведением сопоставляющего алгоритма:

• g — заменить все подходящие под шаблон совпадения;

• i — игнорировать регистр символов при сопоставлении;

• m — включить многострочный режим (см. пункт “Якоря” выше).

Приведенный выше синтаксис довольно прост и подходит ко многим другим реализациям регулярных выражений.

OpenOffice и Microsoft Office

В наше время возможность поиска и замены с помощью регулярных выражений встроена во многие текстовые редакторы, и Microsoft Word с его свободным аналогом OpenOffice Writer — не исключение. Синтаксис регэкспов (так жаргонно называют регулярные выражения) в этих текстовых процессорах практически не отличается от описанного мной ранее синтаксиса POSIX и других традиционных реализаций, за исключением следующих небольших замечаний:

• Кванторы и скобки не экранируются;

• Границы слов обозначаются как “\>” и “\<”;

• Якоря “^” и “$” соответствуют началу и концу абзаца;

• Символ амперсанда “&” в поле замены соответствует найденному шаблону;

• Отсутствуют многие полезные escape-последовательности, например, \w;

• Модификаторы задаются установкой/снятием соответствующих галочек;

• Проверка регистра по умолчанию отключена;

• В поле “Замена” для ссылки на результаты сопоставления подвыражений используется знак доллара: “$номер”;

• Реализованы только базовые встроенные классы символов, при этом само их название является классом, то есть надо писать “[:alpha:]”, а не “[[:alpha:]]”.

Вот несколько примеров простых регулярных выражений в OpenOffice Writer'е. Для их использования необходимо зайти в “Правка” → “Найти и заменить”, после чего выставить галочку в поле “Регулярное выражение” (поле в некоторых версиях может быть скрыто, для показа надо нажать кнопку “Больше параметров”).

Выражение “\<([:alpha:]+)\> +\1\>” находит в тексте повторяющиеся слова, например, “слово слово”, но не “слово словообразование”. Выражение “\<[:upper:]{2,}[:lower:]+\>” при включенной галочке “соблюдать регистр” найдет все слова, начинающиеся с нескольких прописных букв, но не состоящие из них целиком (пропустит аббревиатуры), что является довольно распространенной опиской. Ну и, наконец, выражение “([:upper:][:alpha:]*) ([:upper:]\. *[:upper:]\.)” найдет фамилии с инициалами, записанные в виде “Фамилия И.О.” и может быть использовано для замены такой записи на “И.О. Фамилия”. Для этого в поле “Заменить на” надо указать “$2 $1”. Кроме того, разумеется, в OpenOffice Writer сработают все более сложные примеры для sed, описанные далее (необходимо учесть только небольшие различия в синтаксисе из начала подраздела).

Синтаксис регулярных выражений в MS Word (там они называются “подстановочными знаками”) сильнее отличается от привычного, особенно в плане записи escape-последовательностей, однако общие принципы остаются такими же. Основные отличия Word от Writer состоят в следующем:

• Квантор “+” заменен на “@”;

• Инверсия в списке символов задается как [!символы], а не [^символы], встроенные POSIX-классы символов отсутствуют;

• Границы слов — “>” и “<” не экранируются;

• Самые заметные отличия проявляются в синтаксисе и видах escape-последовательностей:
  - Все они начинаются с символа “^”;
  - Цифрам соответствует “^#”;
  - Буквам — “^$”, классическое значение “^” и “$” утеряно;
  - “Точки” (соответствующей любому символу) нет, вместо нее есть последовательность “^?”.

Например, приведенное выше выражение для поиска дублированных слов в Word может быть записано как “<(^$@)> @\1>”. В целом синтаксис регулярных выражений в Word проще, но имеет меньше возможностей, чем во Writer.

GNU sed

Для иллюстрации простейших классических примеров регулярных выражений я буду использовать стандартные для операционных систем на базе Unix (GNU/Linux в том числе) утилиты cat, echo и sed, поскольку в них впервые появилась поддержка регэкспов, и их использование там считается уже классическим. При этом синтаксис регулярных выражений и поведение алгоритма-сопоставителя в sed, за некоторыми небольшими исключениями, соответствуют общепринятым.

Позволю себе немного отвлечься, чтобы описать философию Unix, поскольку обработка текстов с помощью регэкспов имеет в ней прочные корни. Каждая типичная Unix-программа имеет два[1] предопределенных потока ввода-вывода — "стандартный ввод", обычно являющийся клавиатурой, и "стандартный вывод", обычно являющийся консолью, терминалом; соответственно определены средства для чтения из стандартных потоков и записи в них. Важнейшим свойством Unix является то, что стандартные потоки программ можно перенаправлять в файлы или в другие потоки, делая довольно сложные конвейеры из операций. Нас в данном случае будет интересовать перенаправление потока вывода одной программы в поток ввода другой, осуществляемое операцией "|" (вертикальная черта).

Утилита cat просто-напросто выводит содержимое файла, чье имя передано ей в качестве аргумента, на стандартный вывод[2]. Утилита GNU sed (от ***S**tream **ED**itor*) — это потоковый текстовый редактор, принимающий текст на стандартном вводе и выдающий отредактированную версию на стандартный вывод. Редактирование производится как раз с помощью ключей и регулярных выражений. Утилита echo просто выводит переданную в качестве аргумента строку на стандартный вывод. Я буду использовать ее в простых примерах, передавая строку в кавычках, во избежание ненужного экранирования символов. С помощью операции "|" я буду создавать так называемый "пайп", перенаправляя вывод cat или echo в sed так[3]:

```
> echo "test string" | sed <команда>
>
```

После чего результат редактирования будет выведен на экран. Нужная нам команда sed называется s и имеет синтаксис "s/регулярное_выражение/на_что_заменить/модификатор", заменяя подходящие под шаблон подстроки заданной строкой. Шаблон будем задавать в кавычках. Дальше мы увидим, как оно работает.

[1] На самом деле три, но сейчас нам важны только два.

[2] В реальности поведение cat сложнее — она выполняет конкатенацию файлов, а затем вывод, но нам достаточно простого объяснения.

[3] ">" — это приглашение командной строки.

Кванторы *, +, все варианты {}, группировка подвыражений скобками и операция "|" в sed экранируются для удобства использования из консоли. Модификаторы в sed могут отсутствовать, тогда заменяется только первое вхождение в однострочном режиме с учетом регистра. Существуют также некоторые другие модификаторы, но они важны только для более серьезного использования sed, а не для разговора о регулярных выражениях.

Рассмотрим некоторые примеры использования sed и регулярных выражений. Начнем с довольно надуманных вариантов (чтобы показать работу регулярных выражений, сопоставления я буду заменять на символ "*"):

```
> echo "aabbcc 34"
  | sed s/"[[:digit:]]"/*/
aabbcc *4
>
> echo "aabbcc 34"
  | sed s/"[[:digit:]]"/*/g
aabbcc **
>
> echo "aabbcc 34"
  | sed s/"[[:digit:]]*"/*/
*aabbcc 34
>
```

Последний пример явно нуждается в пояснении. Почему же звездочка поставилась в начале строки, а не вместо числа 34? Проблема здесь в том, что, несмотря на то, что жадный алгоритм выбирает наиболее длинное совпадение из возможных, он также выбирает первое удовлетворяющее шаблону совпадение. Чтобы заменить число, нужно использовать квантор "\+".

```
echo "aabbcc 34"
| sed s/"[[:digit:]]\+"/*/
aabbcc *
>
```

Рассмотрим более близкие к жизни примеры. Допустим, sed используется в некотором скрипте, который получает на входной поток множество имен программ в типичной записи "имя-версия.подверсия.модификация". При этом у вас возникает несколько подзадач, легко решаемых с помощью sed и регулярных выражений: выделить имя программы, выделить полную версию, выделить мажорный номер версии (первую цифру). Эти задачи можно решить так:

```
sed s/"-[[:digit:]]\.*$ "//
```

Выделяем имя программы.

```
sed s/"$[[:alpha:]]\+"//
```

Выделяем полную версию.

```
sed   s/"^[[:alpha:]]*-\([[:digit:]]\+\)*$
"/"\1"/
```

Выделяем мажорную версию. Заметьте, что в данном случае в качестве выражения для замены мы используем результат сопоставления подвыражения в круглых скобках (оно имеет номер 1).

Обычно слишком сложные выражения в sed не используются, а сам он применяется для решения задач, подобных вышеописанной.

Примеры посложнее: Javascript и Perl

Настало время поговорить об использовании регулярных выражений в языках программирования, при этом в качестве элементарного примера, который всегда под рукой, может служить Javascript, в котором для этой цели служит объект RegExp. Синтаксис их описания прост и похож на sed (кванторы и скобки не экранируются):

```
var p = /\w+/i;
var имя = /выражение/модификатор;
```

При этом возможности регэкспов в JS просты:

- Нет предопределенных классов символов, вместо них надо использовать escape-последовательности;
- Всего 3 модификатора: i, g, m;
- Две простейшие операции: проверка на соответствие и вывод соответствий;
- Объект RegExp имеет два метода: test, проверяющий на соответствия (результат — булево значение), и exec, выводящий соответствия (результат — строка или список строк, результирующий объект имеет свойство index, обозначающее место совпадения), методы замены подстроки принадлежат строкам;
- Простейшее “заглядывание” (см. дальше в разделе про Perl).

Приведенный выше пример поиска повторных слов на Javascript может быть записан так[4]:

```
> var p = /(\b\w+\b) +\1/;
> p.exec("Now let's use use JS!");
use use,use
```

Как видно, результатом является список из строк “use use” и “use”.

При использовании модификатора “g” каждый последующий вызов метода “exec” возвращает новое соответствие:

```
> var p = /\d+/g; // число
> var s = "1,2,45";
> p.exec(s);
1
p.exec(s);
2
> p.exec(s);
45
> p.exec(s);
null
```

Вершина эволюции: Perl

Настоящая мощь регулярных выражений полностью проявляется в их “второй колыбели” — языке программирования Perl.

В Perl (и во многих других языках, например, Ruby или уже рассмотренном Javascript) регулярные выражения записываются между двух слэшей: “/I am regexp!/”. Простейший оператор для работы с регэкспами — это оператор проверки на сопоставимость — “=~”, бинарный оператор, возвращающий истину, если левая часть — строка — удовлетворяет правой части — регулярному выражению[5].

```
> perl -e 'print "Hello, World!" = ~ /World/'
1
>
> perl -e 'print "Hello, World!" = ~ /Word/'
>
```

Обратный оператор записывается как “!~”.

```
perl -e 'print "Hello, World!" ! ~ /Word/'
1
>
```

Следует быть аккуратным с пробелами:

```
> perl -e 'print "Hello, World!" =~ /World /'
>
```

Как и в sed, обработчик регулярных выражений в Perl находит самое первое совпадение. При использовании регулярных выражений Perl в Perl-скриптах в них можно указывать имена переменных (обязательно начинаются со знака “$”).

```
# Это скрипт на Perl, со знака "#"
# начинаются комментарии

$test = "World";
"Hello World" =~ /$test/; # удовлетворяет
```

Особенности общего синтаксиса

Кванторы в Perl точно такие же, как и в sed, и в JS (не экранируются). Такое поведение сохраняется и во многих других языках.

[4] “>” — приглашение консоли Javascript.

[5] В примерах далее используется опция командной строки –e, чтобы передать интерпретатору Perl простую однострочную программу.

```
> perl -e 'print "Hello, World!" =
  ~ /l{2}/'
1
>
```

А вот для указания этих символов “как есть” их уже нужно экранировать:

```
> perl -e 'print "2+2" =~ /\+/'
1
>
```

Escape-последовательности в Perl такие же, как и в sed, и в JS, за исключением того, что возможность записать символ с помощью десятичного кода отсутствует, а в восьмеричном коде символы записываются как “\0xxx”. Классы символов, включая предопределенные, аналогичны таковым в sed, однако стоит помнить про специфическое значение символа “$”:

```
# test.pl

$x = "el";
"Hello World" =~ /[$x]/; # удовлетворяет
```

Модификаторы тоже идентичны:

```
perl -e 'print "Hello World" = ~ /WoRlD/i'
1
>
```

При этом, в самых последних версиях Perl (5.14.*) появился модификатор “a”, позволяющий ограничить область предопределенных классов символов только ASCII-кодировкой, а также более строгий модификатор “aa”, с помощью которого можно указать, например, на неравенство внешне одинаковых буквы “K” и символа “K” для градуса Кельвина.

После выполнения сопоставления результаты удачно сопоставленных подвыражений помещаются в переменные $1, $2 и так далее (как и в OpenOffice Writer):

```
# test.pl
"Hello World" =~ /\w+/; # сопоставляется
со словом - Hello
print $1;                # выведет "Hello"
```

Также эти значения являются результатами выполнения всего сопоставления:

```
# test.pl
$time = "22:30:04";
($hours, $minutes, $seconds) =
($time =~ /(\d{1,2}):(\d{2}):(\d{2})/);
#теперь переменные $hours, $minutes
# и $seconds содержат соответственно
# часы, минуты и секунды
```

Возможности обратных ссылок в PCRE расширены. Для записи уже знакомых нам ссылок можно использовать нотацию “\цифра”, однако рекомендуется писать “\g<цифра>” для совместимости с более сложными вариантами. Одно из расширений — относительные отрицательные смещения, “\g{–1}”, например, соответствует предыдущему подвыражению, относительно текущего положения в строке:

```
# test.pl
"Hello World" =~ /((l)\g1)/;
# не удовлетворяет, подвыражение1 -
# это "((1)\1)"
"Hello World" =~ /((l)\g{-1}/;
# удовлетворяет
```

Второе расширение — это именованные подвыражения и ссылки на них. Для использования этой возможности подвыражения должны выглядеть как “(?\<имя\>\<подвыражение>)”:

```
# test.pl
$time = "22:45:32";
$time  =~  /(?<hours>\d{1,2}):(?<minutes>\
d{2}):(?<seconds>\d{2})/;
```

Ссылаться на такие подвыражения можно с помощью конструкции “\g{имя}”. Кроме того, можно сделать так, чтобы на подвыражение нельзя было сослаться вообще (например, скобки нужны просто “для красоты” или для группировки), для этого его необходимо записать как “(?:<подвыражение>)”. При этом результат его сопоставления не попадет в список общих результатов.

В предыдущем разделе мы видели, что стандартные кванторы используют “жадные” алгоритмы, то есть стараются выбрать самое длинное сопоставление из всех возможных. Это поведение не всегда желаемо, поэтому большинство реализаций регулярных выражений, в том числе и используемая в Perl, позволяют использовать нежадные кванторы, для этого после каждого квантора ставится знак вопроса: ??, *?, +?, {*i*,*j*}?, {*i*,}?.

```
> perl -e 'print "Hello World" = ~ /(l+)/'
ll
>
> perl -e 'print "Hello World" = ~ /(l+?)/'
l
>
```

Повышение удобочитаемости регулярных выражений

Поскольку синтаксис регулярных выражений старается быть немногословным, чтобы простые сопоставления не занимали чересчур много места, длинные регэкспы могут быть слишком запутанными и непонятными. Для избавления от этого во многих языках (Perl, Python, Ruby) используется модификатор “x”, позволяющий за-

писывать выражения в так называемом “расширенном” синтаксисе. При этом все выражение можно разбить на несколько строк и снабдить их комментариями:

```
# test.pl

$time = "22:30:04";
$time =~ /(\d{1,2}):  # hours
          (\d{2}):    # minutes
          (\d{2})     # seconds
         /x;
```

Расширенные возможности регэкспов в Perl

Помимо описанных выше простейших вариантов регулярных выражений, Perl позволяет использовать дополнительные возможности для достижения дополнительной гибкости. Одна из таких возможностей — это заглядывание. Заглядывание позволяет указывать часть шаблона, которая должна сопоставляться с исходной строкой, но так, что полученное сопоставление не должно включаться в результат. Например:

```
> perl -e 'print "Hello World" =
  ~ /((?<=Hello )World)/ '
World
>
```

В этом примере результат сопоставления подвыражения “(?<=Hello)” не включается в результат сопоставления целого выражения. Словами это можно выразить как “Найти и включить в результат все слова World, такие, что перед ними идет слово Hello” (так называемое “позитивное заглядывание”). Заглядывание вперед выглядит так:

```
> perl -e 'print "Hello World" =
  ~ /(Hello(?= World))/
Hello
>
```

Существуют также обратные варианты этих выражений, используемые для выражения таких фраз, как “Найти и включить в результат все слова World, такие, что перед ними нет слова Hello” (так называемое “негативное заглядывание”). Они записываются как “(?!подвыражение)” и “(?<!подвыражение)” для заглядывания вперед и назад соответственно. В данный момент поддерживаются только заглядывания фиксированной длины, то есть выражение “(?=\w+)” будет ошибочным.

Еще одной расширенной возможностью регулярных выражений в Perl являются условия, подобные условным операторам в языках программирования. Они имеют следующий вид: “(?(условие)выражение-для-истины|выражение-для-лжи)” или “(?(условие)выражение)”. При этом, если условие истинно, то сопоставляется выражение-для-истины, в противном случае сопоставляется либо следующая часть целого шаблона, либо выражение-для-лжи. Условия могут иметь вид числа в скобках, обозначающего нумерованное подвыражение. Если это подвыражение было успешно сопоставлено, то условие истинно. Также условие может иметь вид заглядывания, определение истинности которого очевидно.

Другим полезным новшеством является возможность запоминания именованных подвыражений и использования их при сопоставлении. Это очень сильно помогает, если есть какое-то подвыражение, часто встречающееся в шаблоне, или просто необходимо сделать регулярное выражение более понятным. Вот тривиальный пример:

```
perl -e 'print "Hello World" =
   ~ /((?&word))(?(DEFINE)(?<word>\w+))/'
Hello
>
```

Подвыражение “(?(DEFINE)(?<word>\w+))” служит для запоминания именованных групп и может иметь вид “(?(DEFINE)(?\<имя1\><определение>)...(?\<имяN\><определение>))”. Обращение к именованному подвыражению производится с помощью конструкции “(?&<имя>)”, которая сама должна быть членом другого подвыражения (заключена в скобки).

В Perl существует еще несколько интересных возможностей для обработки регулярных выражений, например, поддержка рекурсивных шаблонов, однако многие специалисты склонны считать, что такие нововведения лишь перегружают довольно простой аппарат регэкспов и в новой версии Perl — Perl6 рекомендуется использовать для обработки таких сложных моментов механизм грамматик. Однако описание обработки грамматик выходит за рамки этой статьи, поэтому я не буду на нем останавливаться, а вышеприведенных средств с лихвой хватит для разбора большинства встречающихся строк и шаблонов.

Конструкторы регулярных выражений

Для новичков или при составлении очень сложных регулярных выражений хорошим подспорьем могут быть визуальные конструкторы регэкспов, позволяющие составлять, тестировать, отлаживать и анализировать регулярные выражения. Одной из таких программ является Ultrapico Expresso. Кратко рассмотрим ее использование.

Expresso использует практически совместимый с Perl синтаксис регулярных выражений. Основной интерфейс программы разделен на 3 вкладки: режим тестирования, режим конструирования и библиотеку.

Вкладка режима тестирования разделена на четыре окна: содержащее собственно регулярное вы-

ражение, окно-анализатор, тестовый текст и результаты поиска. В первом окне содержится выражение, которое могло быть сконструировано, взято из библиотеки или просто написано от руки. Выражение может быть сопоставлено с тестовым текстом, при этом в окошке результатов отобразятся все результаты сопоставления. Анализатор (присутствующий во всех вкладках) разбивает выражение на логические части и выдает краткую информацию о них.

С помощью вкладки конструирования можно создать регулярное выражение с помощью графических средств — кнопок, чекбоксов и радиокнопок. Отдельно сгруппированы классы символов и кванторы. Например, для создания уже знакомого нам выражения проверки на повторяющиеся слова нужно выполнить следующие действия:

- В подвкладке “Position” выбрать радиокнопку “First or last character in the word” и нажать “Insert”;
- Ввести открывающую скобку;
- В подвкладке “Character” выбрать “Alphanumeric” в группе “Character class”;
- Выбрать “One or more” в группе “Repetitions” (по умолчанию кванторы жадные) и нажать “Insert”;
- Снова выбрать “First or last character in the word” в “Position” и нажать “Insert”;
- Ввести закрывающую скобку;
- В “Character” выбрать “Specific character”, ввести в поле пробел, выбрать “One or more” и нажать “Insert”;
- и так далее.

При этом одновременно работает анализатор.

Библиотека содержит множество примеров уже готовых и полезных регулярных выражений, которые могут быть проанализированы, разобраны, оттестированы и, разумеется, использованы в реальных программах.

Такие программы могут быть незаменимы для тех, кто использует регулярные выражения постоянно, особенно в тех случаях, когда выражения должны быть достаточно сложными и объемными.

Заключение

Регулярные выражения прочно заняли свое место в современной информатике, встречаясь практически в каждом языке программирования и уже практически в каждом текстовом редакторе и процессоре. Их влияние на сферу обработки текстовой информации трудно переоценить.

Регулярные выражения — это мощнейший инструмент для обработки строковых данных, имеющий довольно небольшой синтаксис. К сожалению, несмотря на внешнюю простоту, составление хороших и правильных регулярных выражений — это целое искусство, овладение которым может потребовать продолжительного времени. Эта статья не призвана научить вас такому искусству, ее задача состоит лишь в том, чтобы показать, как делать первые шаги в невероятно выразительном мире регэкспов. Многие простые регулярные выражения, служащие для обработки типовых ситуаций, пишутся довольно элементарно, как вы сможете увидеть в разделе примеров, поэтому не стоит бояться пробовать и экспериментировать. В моем описании некоторые специальные возможности регулярных выражений и особенности их использования в различных языках программирования остались за бортом, о них вы сможете прочитать в дополнительной литературе.

Примеры

Все примеры ниже используют Perl.

1. Разбор IPv4 IP-адреса.

IP-адреса в модели IPv4 состоят из четырех чисел от 0 до 255, разделенных точками. Пример проверяет, является ли заданная строка верным IP-адресом, и в случае успеха возвращает ее же. Оператор “?:” нужен, чтобы зря не возвращался результат сопоставления каждого отдельного числа, а вывелся сразу результат сопоставления всех чисел.

```
# example1.pl
/^((?:(?&part)\.){3})
# три первых числа и точки
  ((?&part))$
# последнее число
   (?(DEFINE)
# определение подвыражения, разбирающего
# отдельные числа
      (?<part>(\d|
      # либо цифра
         [01]?\d{2}|
         # либо опциональная цифра 0 или
         # 1 и еще 2 цифры
         2[0-4]\d|
         # либо 2, цифра от 0 до 4 и
         # еще одна цифра
         25[0-5]
         # либо 25 и цифра от 0 до 5
      )))/x
```

2. Разбор IPv6 IP-адреса.

С адресами в IPv6 ситуация гораздо проще, потому что они состоят из любых шестнадцатеричных цифр. IPv6 адрес выглядит как восемь шестнадцатеричных чисел до четырех цифр каждое, разделенных двоеточиями.

```
# example2.pl
/^((?:[[:xdigit:]]{0,4}:){7})
  ([[:xdigit:]]{0,4})$/x
```

3. Сопоставление с простейшей парой “ключ=значение”.

Пары рассматриваются в виде слова (которое может содержать цифры и знак подчеркивания) и значения (из произвольных символов), разделенных знаком равенства, который может быть окружен

пробелами. Сопоставление возвращает по отдельности ключ и значение.

```
# example3.pl
/^(\w+)\s*=\s*(.*?)$/
```

4. Сопоставление с римскими числами.

Числа в Древнем Риме записывались по особым правилам.

```
# example4.pl
/^((?&rom1)(?&rom2)(?&rom3)(?&rom4))$
   (?(DEFINE)
      (?<rom1>(m{0,3}))
      (?<rom2>(d?c{0,3}|c[dm]))
      (?<rom3>(l?x{0,3}|x[lc]))
      (?<rom4>(v?i{0,3}|i[vx])))
      /xi
```

5. Сопоставление с числами.

Пример регулярного выражения, сопоставляющегося с числами, как имеющими знак, так и нет, как с целыми, так и с вещественными.

```
# example5.pl
/^((?&osg))\ *
# знак, после которого могут быть пробелы
  ((?&int)(?&dec)?|
# десятичная дробь
   (?&dec))
# целую часть можно опустить, если она 0
   ([eE](?&osg)(?&int))?
# экспоненциальная часть
  $
   (?(DEFINE)
      (?<osg>[-+]?)
# опциональный знак
      (?<int>\d++)      # целая часть
      (?<dec>\.(?&int)) # дробная часть
   )/x
```

6. Правильное распознавание времени.

В тексте был приведен пример простого распознавания времени в формате "чч:мм:сс". Однако этот пример не проверял правильность введенного времени. Приведенный ниже пример призван исправить эту оплошность. Примечание: секунды могут быть необязательны.

```
# example6.pl
/((?&hh)):((?&mm))(?::((?&ss)))?
   (?(DEFINE)
      (?<hh>([01]?[0-9]|2[0-3]))
      (?<mm>([0-5]\d))
      (?<ss>([0-5]\d)))
   /x
```

7. Обнаружение повторяющихся подряд слов в тексте.

Дважды написанные слова — частая опечатка. В этом примере результатом сопоставления является одно такое слово (повтор не включается в результат).

```
# example7.pl
/\b([[:alpha:]]+) +\g1\b/i
```

8. Поиск слов, написанных заглавными буквами.

Еще один простейший пример, позволяющий найти слова, целиком написанные прописными буквами.

```
# example8.pl
/\b[[:upper:]]+\b/
```

9. Пример поинтереснее на Javascript: генерация оглавления в HTML-файле.

Приведен скрипт, основная функция в котором ("make_toc") должна вызываться при загрузке элемента body (onload). Подразумевается, что существует div с идентификатором "toc".

```
function make_toc()
{
  var toc = "<h1>Оглавление</h1>";
  var toc_p = new RegExp("<h(\\d).*>
              (.+)</h\\1>", "igm");
  var toc_d = document.getElementById("toc");
  while (match = toc_p.exec(document.body.
innerHTML))
  {
   toc += new Array(parseInt(match[1])+1).
join("**") + " " + match[2] + "<br/>";
  }
  toc_d.innerHTML = toc;
}
```

Использованные источники информации

1. *Doughery D., Robbins A.* Sed & awk, 2nd edition. O'Reilly Media, 1997, 432 с., ISBN: 978-1-56592-225-9.

2. Официальная документация GNU sed [Электронный ресурс], URL: *http://www.gnu.org/software/sed/manual/sed.html*.

3. Регулярные выражения в Javascript [Электронный ресурс], URL: *http://www.w3schools.com/jsref/jsref_obj_regexp.asp*.

4. *Уолл Л., Кристиансен Т., Орвант Д.* Программирование на Perl. М.: Символ-Плюс, 2010, 1150 с., ISBN: 978-5-93286-020-5.

5. Учебник по регулярным выражениям в Perl [Электронный ресурс], URL: *http://perldoc.perl.org/perlretut.html*.

6. *Гойвертс Я.* Регулярные выражения. Сборник рецептов. М.: Символ-Плюс, 2010, 608 с., ISBN: 978-5-93286-181-3.

7. *Фридл Дж.* Регулярные выражения. М.: Символ-Плюс, 2008, 598 с., ISBN: 5-93286-121-5.

“Восьмибитовый художник”, или Задачи ЕГЭ о цветах

О.Б. Богомолова,
д. п. н., учитель информатики и математики ГОУ СОШ № 1360, Восточный округ г. Москвы
Д.Ю. Усенков,
ст. н. с. Института информатизации образования Российской академии образования, Москва

Среди задач, предлагаемых на ЕГЭ начиная с 2009 г., а также в тренажных работах при подготовке к ЕГЭ есть несколько интересных заданий, касающихся *кодирования цветов* — не тех, конечно, которые растут в саду, а тех, которые можно создавать на экране монитора, используя цветовую модель RGB.

Одна из разновидностей таких задач касается умения определять получаемый цвет по его “кодовой записи” в виде 6-значного шестнадцатеричного числа, как это принято в языке HTML.

2009–A15[1]. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor="#FFFFFF">`?

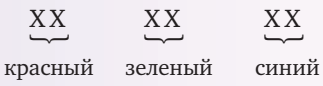
1) белый
2) зеленый
3) красный
4) синий

2010–A15. Для кодирования цвета фона web-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. Какой цвет будет у страницы, заданной тегом `<body bgcolor="#00FF00">`?

[1] Подобное обозначение для заданий ЕГЭ указывает год и обозначение задания в демоварианте ЕГЭ соответствующего года.

1) белый
2) зеленый
3) красный
4) синий

2011–A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели следующим образом:

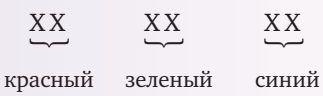


К какому цвету будет близок цвет страницы, заданный тегом `<body bgcolor="#747474">`?
1) серый
2) белый
3) фиолетовый
4) черный

2010/11_T1_B1_A14[2]**.** Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. К какому цвету будет близок цвет страницы, заданной тегом `<body bgcolor="#0FF00F">`?
1) красный
2) синий
3) белый
4) зеленый

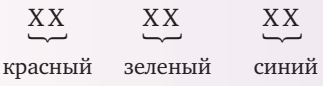
2010/11_T1_B2_A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели. К какому цвету будет близок цвет страницы, заданной тегом `<body bgcolor="#F00F0F">`?
1) красный
2) синий
3) белый
4) зеленый

2010/11_T2_B1_A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели следующим образом:



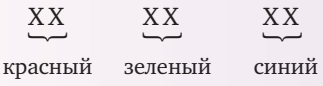
К какому цвету будет близок цвет страницы, заданный тегом `<body bgcolor="#0F0F0F">`?
1) оранжевый
2) белый
3) коричневый
4) черный

2010/11_T2_B2_A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели следующим образом:



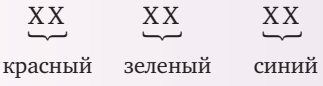
К какому цвету будет близок цвет страницы, заданный тегом `<body bgcolor="#F9F9F9">`?
1) белый
2) оранжевый
3) коричневый
4) черный

2010/11_T3_B1_A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели следующим образом:



К какому цвету будет близок цвет страницы, заданный тегом `<body bgcolor="#9FF9F9">`?
1) белый
2) голубой
3) желтый
4) малиновый

2010/11_T3_B2_A14. Для кодирования цвета фона интернет-страницы используется атрибут `bgcolor="#XXXXXX"`, где в кавычках задаются шестнадцатеричные значения интенсивности цветовых компонент в 24-битной RGB-модели следующим образом:



К какому цвету будет близок цвет страницы, заданный тегом `<body bgcolor="#F90FF9">`?
1) белый
2) голубой
3) желтый
4) малиновый

[2] Подобное обозначение для заданий тренировочных работ, предлагаемых МИОО, указывает учебный год, номер тренировочной работы, номер варианта и обозначение задания в указанном тренажном варианте.

Принципы получения цветов на компьютере (на мониторе, на цветном принтере и т.д.) были подробно описаны в первом выпуске журнала “Информатика” за 2011 г. Однако не зря говорят, что “повторение — мать учения”, поэтому давайте еще раз вспомним, как осуществляется кодирование цветов в системе **RGB**. В ней каждый пиксель представляет собой “смесь” трех точек красного (**R** — ***Red***), зеленого (**G** — ***Green***) и синего (**B** — ***Blue***) цветов, которые называют “основными” для этой цветовой системы и первые буквы английских названий которых как раз и “породили” ее название. При этом каждый из этих основных цветов может иметь различную яркость, которая кодируется двоичным числом и, соответственно, может иметь определенное число градаций.

В рассматриваемой в приведенных выше задачах 24-битной RGB-модели каждый из трех основных цветов кодируется двоичным числом, разряды которого нумеруются от 0 до 7. То есть речь идет о *8-битном кодировании цветов в системе RGB*.

Red:

7	6	5	4	3	2	1	0

8 бит $\Rightarrow 2^8 = 256$ градаций

Green:

7	6	5	4	3	2	1	0

8 бит $\Rightarrow 2^8 = 256$ градаций

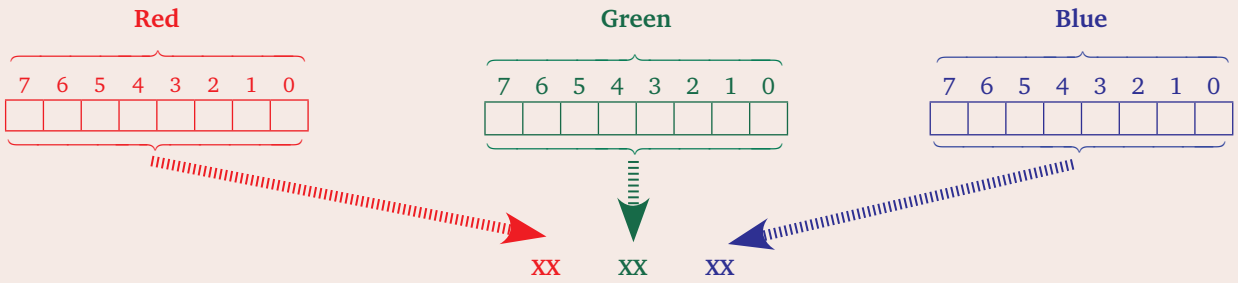
Blue:

7	6	5	4	3	2	1	0

8 бит $\Rightarrow 2^8 = 256$ градаций

Данный способ кодирования цветовых оттенков соответствует широко используемому в современных ПЭВМ 24-битному методу цветового кодирования TrueColor (по-английски дословно: “истинный цвет”). Существует также 32-битный вариант метода TrueColor, где, кроме обычных трех основных цветов R, G и B, предусматривается еще один восьмибитный код — так называемый *“альфа-канал”*, определяющий “прозрачность” пикселя при взаимном наложении нескольких изображений друг на друга. Нетрудно подсчитать, что метод TrueColor обеспечивает, таким образом, $256 * 256 * 256 = 2^{8+8+8} =$ $= 16\ 777\ 216$ различных цветовых оттенков. Именно это “цветовое богатство” и превращает современный компьютер в великолепного “художника”, способного создавать на экране практически любые картины, пейзажи, интерьеры и пр., — как отображая реальные фотографии, так и создавая “виртуальные” объекты, не существующие в природе.

При использовании 24-битной RGB-модели TrueColor для кодирования цветовых оттенков при создании web-страниц обычно используется шестнадцатиразрядная запись, в которой подряд записывается шесть шестнадцатиразрядных цифр: первые две цифры соответствуют шестнадцатеричной записи значения яркости **красного** цвета (***Red***), вторые две — шестнадцатеричной записи значения яркости **зеленого** цвета (***Green***), а третья пара цифр определяет шестнадцатеричную запись значения яркости **синего** цвета (***Blue***):

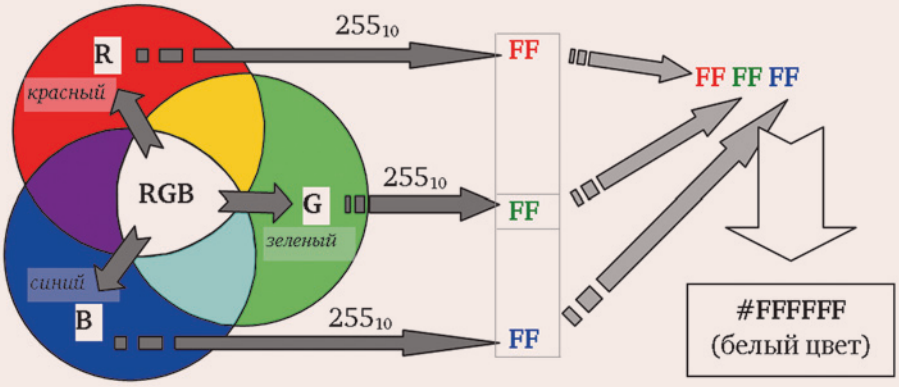


— где стрелки обозначают перевод из двоичной в шестнадцатеричную систему счисления согласно знакомому любому школьнику правилу: исходное двоичное число (дополненное, если требуется, до 8 разрядов незначащими нулями слева) делится на две половины по 4 бита в каждой, а затем каждая такая “тетрада” битов заменяется одной шестнадцатеричной цифрой по следующей таблице:

0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Для облегчения работы школьникам авторы данной статьи предлагают использовать схему, которую можно оформить и в форме плаката.

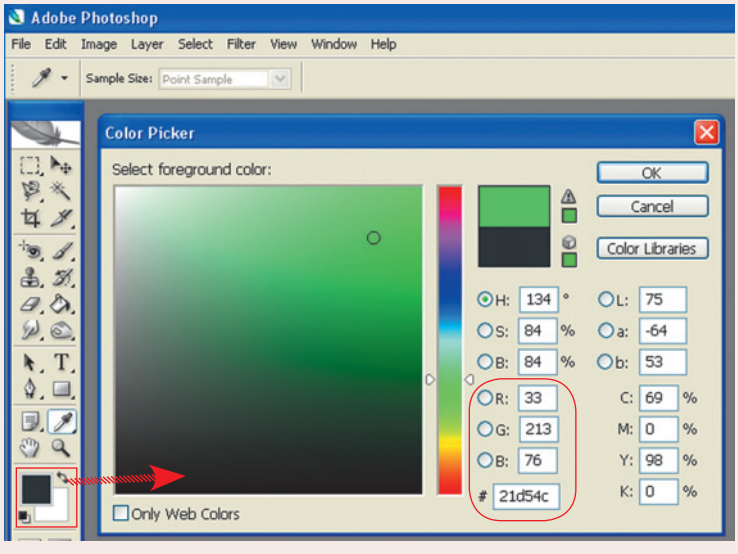
Принципы кодирования цветовых оттенков. Модель RGB



Коды и названия основных цветовых оттенков

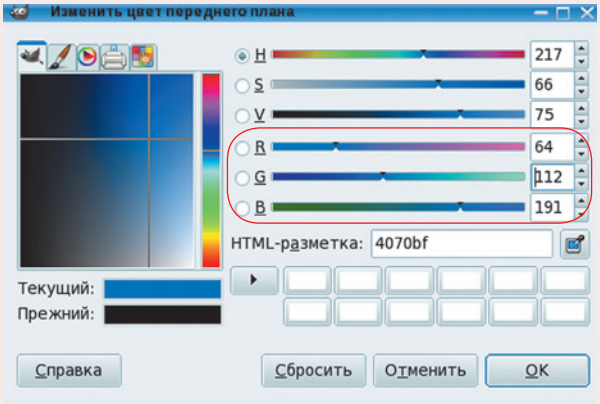
Оттенок	Код	Название	Образец цвета
Черный	#000000	black	
Белый	#FFFFFF	white	
Красный	#FF0000	red	
Светло-зеленый	#00FF00	lime	
Синий	#0000FF	blue	
Желтый	#FFFF00	yellow	
Фиолетовый	#FF00FF	fuchsia	
Голубой	#00FFFF	cyan	
Серый	#808080	gray	
Коричневый (темно-красный)	#800000	darkred	
Зеленый	#008000	green	
Ультрамариновый (темно-синий)	#000080	navy	
Пурпурный	#800080	purple	
Оливковый	#808000	olive	
Темно-голубой	#008080	teal	
“Серебряный” (светло-серый)	#C0C0C0	silver	

Выполнять пересчет двоичных значений кодов основных цветов RGB в шестнадцатеричные и обратно можно при помощи стандартного приложения Windows “Калькулятор” (в варианте **Вид — Инженерный**). Существует также немало специальных программ “в помощь web-дизайнеру”, которые позволяют работать с цветовыми оттенками, в том числе просматривать оттенки, соответствующие заданному числовому коду, выполнять пересчет числовых кодов из одной системы счисления, способа записи или цветовой модели в другую, а нередко — и получать числовую запись для произвольного цветового оттенка, “взятого” с экрана при помощи инструмента **Пипетка**.



Не менее удобные возможности предоставляют и современные графические редакторы. Например, в графическом редакторе Adobe PhotoShop после двойного щелчка мышью на “образце” цвета в текущей рабочей палитре (цвет переднего плана / цвет фона) раскрывается диалоговое окно задания произвольного цвета, где есть цветовая панель, а также поля для ввода числовых значений основных цветов в различных цветовых моделях, в том числе в модели RGB и в интересующем нас способе кодирования цветов в HTML (поле, “озаглавленное” символом “#”):

Аналогичным способом устроено и окно задания произвольного цвета в свободно распространяемом графическом редакторе GIMP, — разве что интересующее нас поле здесь названо гораздо понятнее: **HTML-разметка**:



Наконец, очень интересные возможности предоставляет нам… поисковый сервис Яндекс, в котором со сравнительно недавних пор реализован один из самых удачных по исполнению (по полезности, удобству использования и дизайну) модулей — **Калькулятор цветов**.

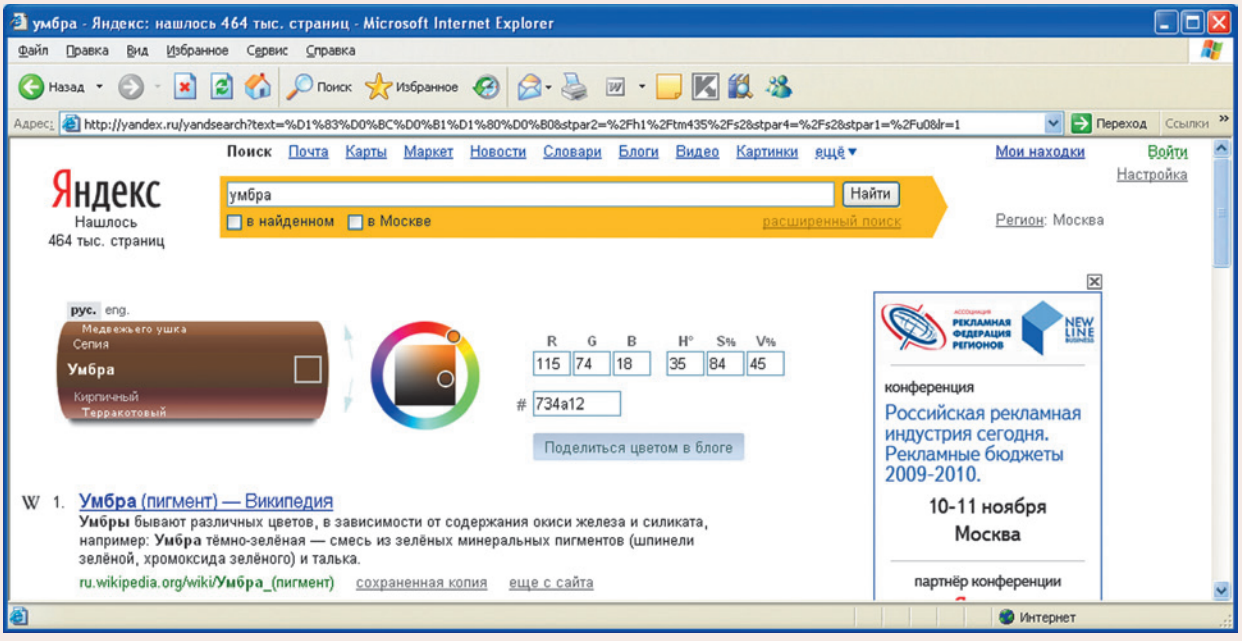
Чтобы вызвать этот модуль в работу, достаточно ввести в поисковой строке либо словесное название интересующего вас (или вообще любого) цвета, либо обозначение `rgb` или `hsv`, а затем через пробелы — три числа, кодирующие цвет.

Сам модуль состоит из трех компонентов. Слева располагается цветовой образец в виде своеобразного “барабана”, в котором посередине располагается искомый оттенок (с надписанным его названием), а выше и ниже — другие оттенки, близкие к найденному. Щелкая мышкой на расположенных справа от “барабана” изогнутых стрелках, можно “вращать барабан”, просматривая эти соседние цвета (вместе с их названиями), — при этом в маленьком квадратике справа на “барабане” по-прежнему остается первый найденный цвет, что позволяет сравнивать его с просматриваемыми оттенками. А если щелкнуть на полоске с любым “боковым” оттенком (или на его названии), то уже выбранный оттенок станет текущим.

В середине располагается интерактивная палитра в виде радужного круга (где выбирается нужный цвет) и квадрата с выбранным оттенком, кружочек на котором позволяет выбирать яркость и насыщенность.

Наконец, справа находятся поля ввода, где можно прочитать (либо, наоборот, ввести — для просмотра соответствующего нового цвета) цифровые значения параметров в кодировке RGB и HSV или шестнадцатеричный код цвета для использования в языке HTML, — точно такое же поле “#”, как и в графическом редакторе Adobe PhotoShop.

А теперь, когда мы подробно усвоили принципы цветового кодирования в 24-битной RGB-модели TrueColor и в языке HTML, в частности, — можно перейти к задачам. При этом, в отличие от статьи в № 1/2011, где для таких задач приводились только готовые ответы, а пояснения были даны отдельно ко всем таким задачам сразу (что для школьников не очень понятно), дадим к каждой задаче отдельно необходимые решение и пояснения к решению. Кроме того, для всех задач (кроме одной, где в итоге получается белый цвет) мы будем давать “цветовой образец” получаемого оттенка, чтобы читатели сами могли решить, насколько этот оттенок близок к названному в правильном ответе. А поскольку их полные условия были приведены в начале статьи и повторять их еще раз не хочется, будем далее указывать только соответствующее “кодовое обозначение” каждой задачи и “ключевую” часть ее условия — запись тега HTML и предлагаемые варианты ответов.



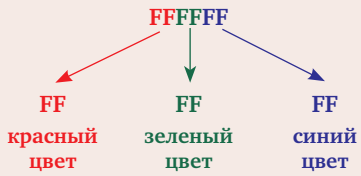
2009–A15. Тег: `<body bgcolor="#FFFFFF">`.

Варианты ответов:

1) белый
2) зеленый
3) красный
4) синий

Решение

Рассмотрим шестнадцатеричную запись искомого цветового оттенка, указанную в записи тега `<body bgcolor=...>`. В данном случае она представляет собой шесть цифр F, которые нужно разбить на три пары:



Прежде всего нужно вспомнить, что шестнадцатеричное значение FF соответствует двоичному 11111111 или десятичному 255, т.е. каждый из записанных этим кодом основных цветов системы RGB имеет максимально возможную яркость. А это, как мы знаем, соответствует белому цвету.

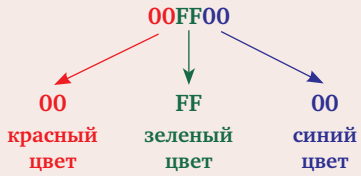
Ответ: белый цвет (вариант 1).

2010–A15. Тег: `<body bgcolor="#00FF00">`.

Варианты ответов:

1) белый
2) зеленый
3) красный
4) синий

Решение



Нетрудно видеть, что в данном случае яркости красного и синего основных цветов — минимально возможные (значение 00, т.е. нуль), а яркость зеленого цвета — максимально возможная (FF_{16} = $= 11111111_2 = 255_{10}$). Очевидно, что это — чистый зеленый цвет:

Ответ: зеленый цвет (вариант 2).

2011–A14. Тег: `<body bgcolor="#747474">`.

Варианты ответов:

1) серый
2) белый
3) фиолетовый
4) черный

Решение

Такая задача в отличие от двух предыдущих несколько сложнее. Если ранее в теге была задана запись какого-либо чистого основного цвета или смеси таких чистых цветов, которые кодировались значениями 00 или FF, то теперь значения основных цветов заданы произвольные (из допустимого интервала от 00 до FF), а учащемуся требуется сформулировать "экспертную" оценку — понять, к какому из названных цветов ближе всего оттенок, заданный таким кодом. Однако принцип решения таких "экспертных" задач — тот же самый.

Обратим внимание на то, что все три значения яркостей для всех трех основных цветов абсолютно одинаковы. Это означает, что какими бы ни были эти значения, получаемый оттенок будет *монохромным* — это будет серый цвет некоторой яркости. (Если все три значения яркостей основных цветов равны 00, то получается "самый темный серый цвет" — черный, а если все три основных цвета имеют яркость FF, то получается "самый светлый серый цвет" — белый. Во всех же "промежуточных" случаях серый цвет тем темнее, чем меньше числовое значение яркости основных цветов, и наоборот.)

В нашем случае среди приведенных в задаче вариантов ответов имеется три "монохромных" цвета (т.е. фиолетовый отпадает сразу). Далее, — у нас не белый цвет (белый кодировался бы числовой записью FFFFFF) и не черный (для которого кодовая запись соответствует 000000). Следовательно, записанный в теге цвет — серый:

Ответ: серый цвет (вариант 1).

2010/11_T1_B1_A14.

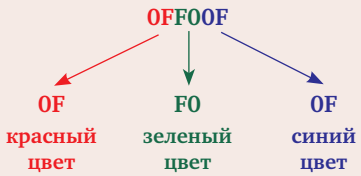
Тег: `<body bgcolor="#0FF00F">`.

Варианты ответа:

1) красный
2) синий
3) белый
4) зеленый

Решение

Как и в предыдущей задаче, начинаем с "разложения на основные составляющие RGB-спектра" заданной кодовой записи цвета:



А теперь надо будет вспомнить принципы записи чисел в позиционной системе счисления (к которым относится и шестнадцатеричная): одна и та же цифра, стоящая правее, имеет меньший "вес" (кратно основанию системы счисления), чем та же самая цифра, стоящая левее. Поэтому в нашем случае для красного и синего цветов яркости заданы сравнительно небольшие ($0F_{16} = 00001111_2 = 15_{10}$), а вот яркость зеленого цвета почти максимальная

($F0_{16} = 11110000_2 = 240_{10}$). В такой ситуации результирующий цвет будет конечно же ближе всего к чистому зеленому:

Ответ: зеленый цвет (вариант 4).

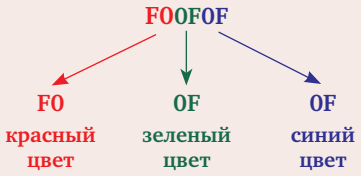
2010/11_T1_B2_A14.

Тег: `<body bgcolor="#F00F0F">`.

Варианты ответов:

1) красный
2) синий
3) белый
4) зеленый

Решение



Здесь ситуация в целом аналогична предыдущей задаче: яркость зеленого и синего цветов почти минимальна ($0F_{16} = 00001111_2 = 15_{10}$), а яркость красного — достаточно велика ($F0_{16} = 11110000_2 = 240_{10}$). Значит, получаемый цвет ближе всего к красному:

Ответ: красный цвет (вариант 1).

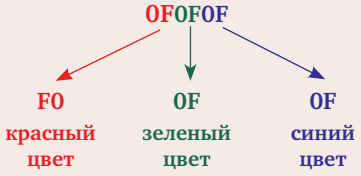
2010/11_T2_B1_A14.

Тег: `<body bgcolor="#0F0F0F">`.

Варианты ответов:

1) оранжевый
2) белый
3) коричневый
4) черный

Решение



Здесь, как нетрудно видеть, во-первых, яркости всех трех основных цветов одинаковы, — значит, получится монохромный оттенок, — а во-вторых, эти яркости очень малы ($0F_{16} = 00001111_2 = 15_{10}$). Значит, получаемый цвет будет ближе всего к черному:

Ответ: черный цвет (вариант 4).

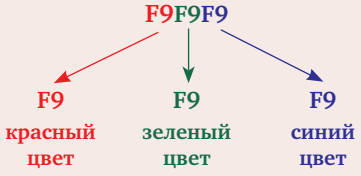
2010/11_T2_B2_A14.

Тег: `<body bgcolor="#F9F9F9">`.

Варианты ответов:

1) белый
2) оранжевый
3) коричневый
4) черный

Решение



Здесь, как и в предыдущем случае, яркости всех трех основных цветов равны, т.е. тоже получится монохромный оттенок. Значения же этих яркостей равны $F9_{16} = 11111001_2 = 15 * 16 + 9 = 249_{10}$. Это очень высокая яркость: ведь максимально возможное ее значение (255) отличается от заданного всего на 6 единиц! Поэтому получаемый цвет будет ближе всего к белому:

Ответ: белый цвет (вариант 1).

2010/11_T3_B1_A14.

Тег: `<body bgcolor="#9FF9F9">`.

Варианты ответов:

1) белый
2) голубой
3) желтый
4) малиновый

Решение

В этом случае яркость красного цвета равна $9F_{16} = 10011111_2 = 9 * 15 + 15 = 159_{10}$, а яркости зеленого и синего цветов составляют $F9_{16} = 11111001_2 = 15 * 16 + 9 = 249_{10}$. Как нетрудно понять, яркости 159 и 249 довольно близки, но все же можно сделать вывод, что в получаемом оттенке больше зеленого и синего цветов, а красного — меньше. Значит, из названных вариантов к получаемому оттенку будет ближе всего голубой цвет (смесь синего и зеленого), а "примесь" красного цвета сделает "чисто" голубой цвет более светлым, ненасыщенным:

Ответ: голубой цвет (вариант 2).

2010/11_T3_B2_A14.

Тег: `<body bgcolor="#F90FF9">`.

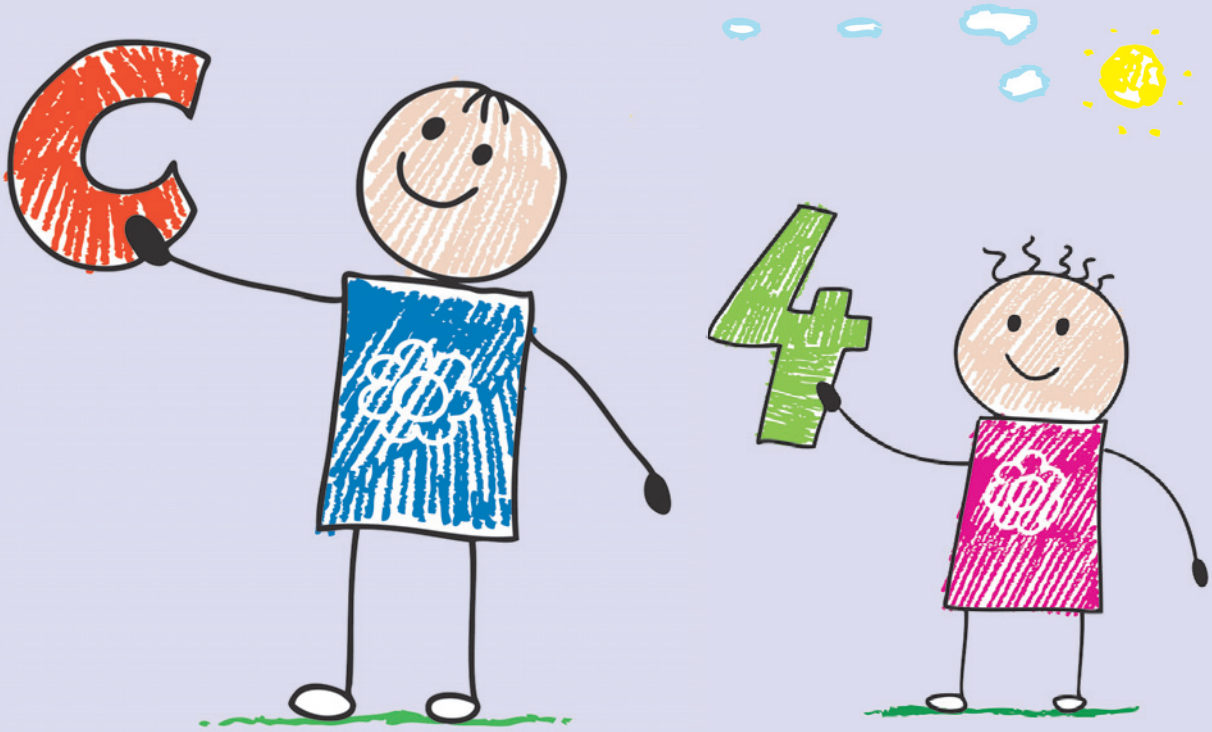
Варианты ответов:

1) белый
2) голубой
3) желтый
4) малиновый

Решение

Здесь яркости красного и синего цветов достаточно велики: $F9_{16} = 11111001_2 = 15 * 16 + 9 = 249_{10}$, а вот яркость зеленого — очень мала: $0F_{16} = 00001111_2 = 15_{10}$. Такую небольшую "примесь" зеленого цвета мы можем просто проигнорировать и считать, что искомый оттенок получается как смесь достаточно ярких красного и синего цветов. А мы знаем, что такая смесь дает фиолетовый (либо, в других русскоязычных определениях английского *magenta*, лиловый или малиновый) оттенок:

Ответ: малиновый цвет (вариант 4).



Методика решения задачи С4 из демонстрационного варианта ЕГЭ по информатике 2012 года

Д.М. Златопольский,
Москва

Условие задачи

В командных соревнованиях по программированию для решения предлагается не больше 11 задач. Команда может решать предложенные задачи в любом порядке. Подготовленные решения команда посылает в единую проверяющую систему соревнований. Вам предлагается написать эффективную, в том числе и по используемой памяти, программу, которая будет статистически обрабатывать пришедшие запросы, чтобы определить наиболее популярные задачи. Следует учитывать, что количество запросов в списке может быть очень велико, так как многие соревнования проходят с использованием Интернета.

Перед текстом программы кратко опишите используемый вами алгоритм решения задачи.

На вход программе в первой строке подается количество пришедших запросов *N*. В каждой из последующих *N* строк записано название задачи в виде текстовой строки. Длина строки не превосходит 100 символов, название может содержать буквы, цифры, пробелы и знаки препинания.

Пример входных данных:

```
6
A+B
Крестики-нолики
Прямоугольник
Простой делитель
A+B
Простой делитель
```

Программа должна вывести список из трех наиболее популярных задач с указанием количества запросов по ним. Если в запросах упоминаются менее трех задач, то выведите информацию об имеющихся задачах. Если несколько задач имеют ту же встречаемость, что и третья по частоте встречаемости задача, их тоже нужно вывести.

Пример выходных данных для приведенного примера входных данных:

A+B 2
Простой делитель 2
Крестики-нолики 1
Прямоугольник 1

Прежде всего обратим внимание на требование кратко описать используемый алгоритм решения, впервые представленное в задаче С4. Кроме того, заметим, что, по нашему мнению, использование термина "запрос" в условии является не совсем правильным (запрос — это обращение к базе данных или т.п., в то время как, согласно условию, в программу поступают как бы ответы на запрос об очередной решенной задаче).

Анализ решения

В программе на школьном алгоритмическом языке применим два массива из 11 элементов в каждом:

1) с именем *задачи* — для хранения названий задач;

2) с именем *кол_задач* — для хранения общего числа упоминания каждой из задач в результатах запросов.

Используем также величину *всего_разных_задач* — количество различных задач в результатах запросов.

Общая схема решения обсуждаемой задачи такая:

```
1. Ввести количество запросов N
2. Для каждого результата запроса
  2.1. Прочитать название задачи
  2.2. Определить, есть ли такая задача
       в списке ранее введенных задач
       (в массиве задачи)
        если есть
         то
          увеличить счетчик количества
          соответствующей задачи
          в массиве кол_задач
         иначе
          - увеличить на 1 количество
            различных задач
            всего_разных_задач;
          - записать новую задачу
            в массив задачи;
          - соответствующему ей значению
            в массиве кол_задач
            присвоить 1
        все
```

```
3. Сравнить значение всего_разных_задач
   с 3:
   если всего_разных_задач < 3
    то
      вывести из массива с названиями
         задачи с индексами от 1 до
         всего_разных_задач
         и для каждой такой задачи -
         число ее вхождений
         из массива кол_задач
    иначе
      вывести из массива с названиями
      задачи, количество которых
      в массиве кол_задач
      не меньше, чем у задачи,
      занимающей третье место
      по частоте встречаемости,
      и для каждой такой задачи - ее
      встречаемость
      из массива кол_задач
   все
```

Для выполнения, так сказать, "«ветви» **иначе**" этапа 3 целесообразно отсортировать массив *кол_задач* в порядке невозрастания значений его элементов (и, соответственно, изменить структуру массива *задачи*)[1].

Из проведенного анализа вытекает, что для решения обсуждаемой задачи необходимо уметь решать ряд частных задач. Обсудим их.

1. Определение того факта, что некоторая решенная задача уже имеется в списке ранее введенных задач (в массиве *задачи*)

Более подробное условие задачи: "В массиве *задачи* записаны названия некоторого известного числа задач. Определить, имеется ли в массиве заданная задача. Если имеется, то определить также ее индекс. Известно, что общее число различных названий — не больше 11".

Решение

В дополнение к массивам *задачи* и *кол_задач* и величине *всего_разных_задач* (см. выше) используем в программе также величину *задача* — название очередной задачи.

С целью упрощения программы и ее отладки целесообразно ввести условные названия решенных задач в виде "1", "2", … и задать некоторое значение величины *всего_разных_задач*.

Наиболее рациональный вариант решения обсуждаемой задачи — применение оператора цикла с предусловием (условие продолжения работы оператора — сложное):

```
алг Вспомогательная_задача_1
нач лит таб задачи[1:11], лит задача,
    цел всего_разных_задач, i
. всего_разных_задач := 5
. задачи[1] := "1"
```

[1] Это не помешает выполнению и "«ветви» **то**".

```
. задачи[2] := "2"
. задачи[3] := "3"
. задачи[4] := "4"
. задачи[5] := "5"
. ввод задача
. |Проверяем названия имеющихся задач
. i := 1
. нц пока i <= всего_разных_задач
          и задачи[i] <> задача
. . i := i + 1
. кц
. если i <= всего_разных_задач
. . то |Такая задача уже в массиве есть
. . . вывод "Такая задача есть.
             Ее номер: ", i
. . иначе
. . . вывод "Такой задачи нет"
. все
кон
```

Примечание. При отладке целесообразно проверить работу программы для случаев, когда значение величины *всего_разных_задач* равно 0 и 11.

2. Заполнение массива *задачи* неповторяющимися значениями

Более подробное условие задачи: “Известны названия *N* задач, среди которых есть повторяющиеся названия. Заполнить массив *задачи* неповторяющимися названиями. Известно, что общее число различных задач — не больше 11”.

Решение

Здесь для решения сначала для каждого очередного названия необходимо определить (как в предыдущей задаче), имеется ли оно в массиве. Если нет — то:

1) увеличить общее число различных названий *всего_разных_задач* на 1;

2) записать это название в массив (в “новый” элемент).

Соответствующая программа:

```
алг Вспомогательная_задача_2
нач цел N, всего_разных_задач, i, j,
    лит таб задачи[1:11], лит задача
. |Ввод значения N
. ввод N
. всего_разных_задач := 0
. |Вводим N названий и заполняем
. |массив задачи различными названиями
. нц для i от 1 до N
. . ввод задача
. . |Определяем, имеется ли такая
  |задача в массиве
. . j := 1
. . нц пока j <= всего_разных_задач и
            задачи[j] <> задача
. . . j := j + 1
. . кц
. . если j > всего_разных_задач
. . . то |Встретилась новая задача
. . . . |Увеличиваем значение
         |всего_разных_задач
. . . . всего_разных_задач :=
         всего_разных_задач + 1
. . . . |Записываем новую задачу
         |в массив
. . . . задачи[всего_разных_задач] :=
         задача
. . все
. кц
. |Выводим названия задач из массива
. нц для i от 1 до всего_разных_задач
. . вывод задачи[i], " "
. кц
кон
```

Новое название можно также записать в массив следующим образом:

```
задачи[j] := задача
```

3. Заполнение массива *задачи* неповторяющимися значениями и определение “встречаемости” (количества вхождений) каждой задачи

Более подробное условие задачи: “Известны названия *N* задач, среди которых есть повторяющиеся названия. Заполнить массив *задачи* неповторяющимися названиями, а также определить частоту встречаемости каждого названия. Известно, что общее число различных задач — не больше 11”.

Решение

В дополнение к задаче 2 необходимо использовать массив *кол_задач* и для “новой” задачи записать в соответствующий элемент этого массива 1, а для имеющейся — увеличить ее счетчик на 1:

```
алг Вспомогательная_задача_3
нач цел N, всего_разных_задач, i, j,
    лит таб задачи[1:11],
    цел таб кол_задач[1:11], лит задача
. |Обнуление элементов массива
  |кол_задач
. нц для i от 1 до 11
. . кол_задач[i] := 0
. кц
. |Ввод значения N
. …
. |Вводим N названий
. |и заполняем массив задачи,
. |подсчитывая также число
  |вхождений каждой задачи
. нц для i от 1 до N
. . ввод задача
. . |Определяем, имеется ли такая
  |задача в массиве задачи
. …
. . если j <= всего_разных_задач
. . . то |Такая задача уже
         |в массиве есть
. . . . |Увеличиваем ее счетчик
```

```
. . . . кол_задач[j] := кол_задач[j] + 1
. . . иначе |Встретилась новая задача
. . . . |Увеличиваем значение
        |всего_разных_задач
. . . . всего_разных_задач :=
                    всего_разных_задач + 1
. . . . |Записываем новую задачу
        |в массив задачи
. . . . задачи[всего_разных_задач] :=
                                  задача
. . . . |Число вхождений этой задачи
        |пока равно 1
. . . . кол_задач[всего_разных_задач] := 1
. . все
. кц
. |Выводим названия задач из массива
  |задачи и соответствующее число
  |вхождений из массива кол_задач
. нц для i от 1 до всего_разных_задач
. . вывод задачи[i], " ", кол_задач[i]
. кц
кон
```

4. Сортировка массива *кол_задач* в порядке невозрастания (и, соответственно ей, изменение массива *задачи*)

Применим для решения этой задачи один из методов сортировки — сортировку обменом (“пузырьковую” сортировку). Напомним, что при сортировке обменом все соседние элементы массива попарно сравниваются друг с другом и меняются местами в том случае, если предшествующий элемент меньше последующего. Этот процесс повторяется (n – 1) раз, где n — общее число элементов массива.

Если индексы “левых” элементов в паре сравниваемых обозначить *лев*, то фрагмент программы, реализующий описанные действия применительно к массиву *a* на одном проходе, выглядит так:

```
нц для лев от 1 до n - 1
. |Если "левый" элемент в паре
  |сравниваемых
. |меньше "правого"
. если a[лев] < a[лев + 1]
. . то |Проводим их обмен
. . . всп := a[лев]
. . . a[лев] := a[лев + 1]
. . . a[лев + 1] := всп
. все
кц
```

— где *всп* — вспомогательная переменная, а вся процедура сортировки оформляется следующим образом:

```
|Проходим по массиву n - 1 раз,
нц для номер_прохода от 1 до n - 1
  |сравнивая пары элементов
  нц для лев от 1 до n - 1
    если a[лев] < a[лев + 1]
      то |Проводим обмен
       всп := a[лев]
       a[лев] := a[лев + 1]
       a[лев + 1] := всп
    все
  кц
кц
```

Можно усовершенствовать программу, учитывая следующее обстоятельство. В ходе первого прохода минимальный элемент постепенно смещается вправо и в конце концов занимает свое (которое он должен занимать в упорядоченном массиве — крайнее правое) место в массиве. После этого его можно исключить из дальнейшей обработки. Затем процесс повторяется, и свое место занимает второй по величине элемент, который также исключается из дальнейшего рассмотрения. Так продолжается до тех пор, пока весь массив не будет упорядочен.

Выпишем пары индексов элементов, сравниваемых на каждом проходе с учетом сказанного только что, в виде таблицы

Номер прохода по массиву	**Индексы**				
1-й	1 – 2	2 – 3	…	(n – 2) – (n – 1)	(**n – 1**) – n
2-й	1 – 2	2 – 3		(**n – 2**) – (n – 1)	
…					
(n – 1)-й	**1** – 2				

Если индекс “левого” элемента в последней паре сравниваемых на каждом проходе чисел обозначить *посл_лев* (соответствующие значения в таблице выделены жирным начертанием), то можно так оформить фрагмент программы, осуществляющей сортировку:

```
нц для посл_лев от n - 1 до 1 шаг -1
  нц для лев от 1 до посл_лев
    если a[лев] < a[лев + 1]
      то |Проводим обмен
       …
    все
  кц
кц
```

Заметим, что в программах, приведенных в демонстрационном варианте, применен именно такой вариант оформления сортировки обменом.

Можно также в качестве параметра “наружного” оператора цикла использовать номер прохода, а конечное значение параметра “внутреннего” оператора цикла связать с номером прохода согласно таблице (оно равно n – *номер_прохода*):

```
нц для номер_прохода от 1 до n - 1
  нц для лев от 1 до n - номер_прохода
    если a[лев] < a[лев + 1]
```

```
        то |Проводим обмен
           …
      все
    кц
  кц
```

По нашему мнению, такой вариант более “запоминаем”, и именно его мы применим при сортировке массивов *кол_задач* и *задачи*:

```
нц для i от 1 до всего_разных_задач - 1
  нц для лев от 1 до всего_разных_задач - i
    если кол_задач[лев] < кол_задач[лев + 1]
      то |Проводим обмен элементов
         |массива кол_задач
        всп := кол_задач[лев]
        кол_задач[лев] := кол_задач[лев + 1]
        кол_задач[лев + 1] := всп
        |и соответствующих элементов
        |массива задачи
        всп_задача := задача[лев]
        задача[лев] := задача[лев + 1]
        задача[лев + 1] := всп_задача
    все
  кц
кц
```

— где *всп_задача* — вспомогательная переменная строкового типа.

Можно также прекратить проходы по массиву, как только он станет отсортированным (в общем случае это может произойти менее чем за $n - 1$ проходов). Признак, по которому целесообразно установить факт отсортированности массива, — на каком-то проходе обменов элементов местами не было. Однако при этом программа усложняется.

После решения частных задач обсудим также вопрос о выводе требуемых в условии “основной” задачи результатов.

Итак, мы отсортировали массив *кол_задач* в порядке невозрастания и изменили соответственно структуру массива *задачи*. После этого, согласно требованиям условия, вывод результатов может быть оформлен в виде:

```
если всего_разных_задач < 3
. то
. . |Выводим все задачи и их количество
. . нц для i от 1 до всего_разных_задач
. . . вывод нс, задачи[i], " ",
                кол_задач[i]
. . кц
. иначе
. . |Выводим две задачи с наибольшим
    |числом вхождений
. . нц для i от 1 до 2
. . . вывод нс, задачи[i], " ",
                кол_задач[i]
. . кц
. . |и задачи с той же частотой
    |встречаемости, что и третья
    |по частоте задача
. . i := 3
. . нц пока i <= всего_разных_задач и
              кол_задач[i] = кол_задач[3]
. . . вывод нс, задачи[i], " ",
                кол_задач[i]
. . . i := i + 1
. . кц
все
```

Обратим внимание на сложное условие в последнем операторе цикла.

Можно упростить фрагмент, связанный с выводом результатов, следующим образом:

```
если всего_разных_задач < 3
. то
. . j := всего_разных_задач
. иначе
. . j := 3
все
i := 1
нц пока i <= всего_разных_задач и
        кол_задач[i] >= кол_задач[j]
. . . вывод нс, задачи[i], " ",
                кол_задач[i]
. . . i := i + 1
кц
```

Именно так оформлен этот фрагмент в решении задачи в демонстрационном варианте экзамена. Его особенности предлагаем читателям проанализировать самостоятельно.

Полностью программа приведена на диске к данному номеру.

Здесь же перечислим ее основные части:

1) обнуление элементов массива *кол_задач* (в школьном алгоритмическом языке это является обязательным);

2) ввод значения *N*;

3) ввод *N* названий задач и заполнение ими массива *задачи* с подсчетом числа вхождений каждой задачи;

4) сортировка массива *кол_задач* в порядке неубывания и, соответственно ей, изменение массива задачи;

5) вывод требуемых результатов.

На диске к данному номеру представлена также программа на языке Паскаль из демонстрационного варианта ЕГЭ. Заметим, что в ней величина *s* (название очередной задачи) и массив *Names* (с названием задач) описаны как:

```
s : string;
Names : array [1..11] of string;
```

— в то время как согласно условию длина строки с названием задачи не превышает 100 символов.

Кроме того, при упорядочивании массивов в качестве величины, вспомогательной для обмена элементов массива *Names*, используется величина *s*.

Lorem Ipsum, чай и французские булочки ☺

Д.Ю. Усенков,
ст. н. с. Института информатизации образования Российской академии образования, Москва

Как известно, создать файл документа одного из приложений Microsoft Office можно разными путями. Можно открыть соответствующее приложение, создать в нем наполнение документа, а затем сохранить его в файл на диске. А можно сразу щелкнуть в желаемой папке правой кнопкой мыши, выбрать в появившемся контекстном меню пункт **Создать**, а затем в раскрывшемся подменю выбрать требуемое приложение (*рис.* 1).

При этом в папке создается пустой файл документа (с “типовым” именем и соответствующим расширением имени). Далее можно привычным двойным щелчком мыши раскрыть его на редактирование и создавать нужное вам наполнение.

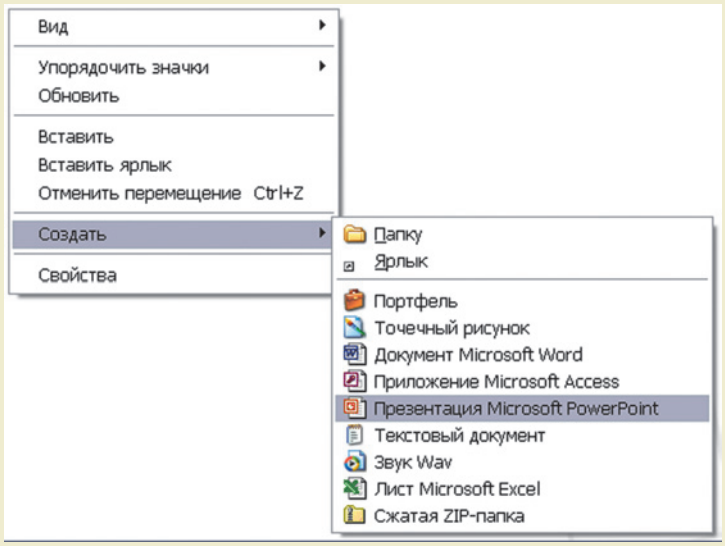


Рис. 1

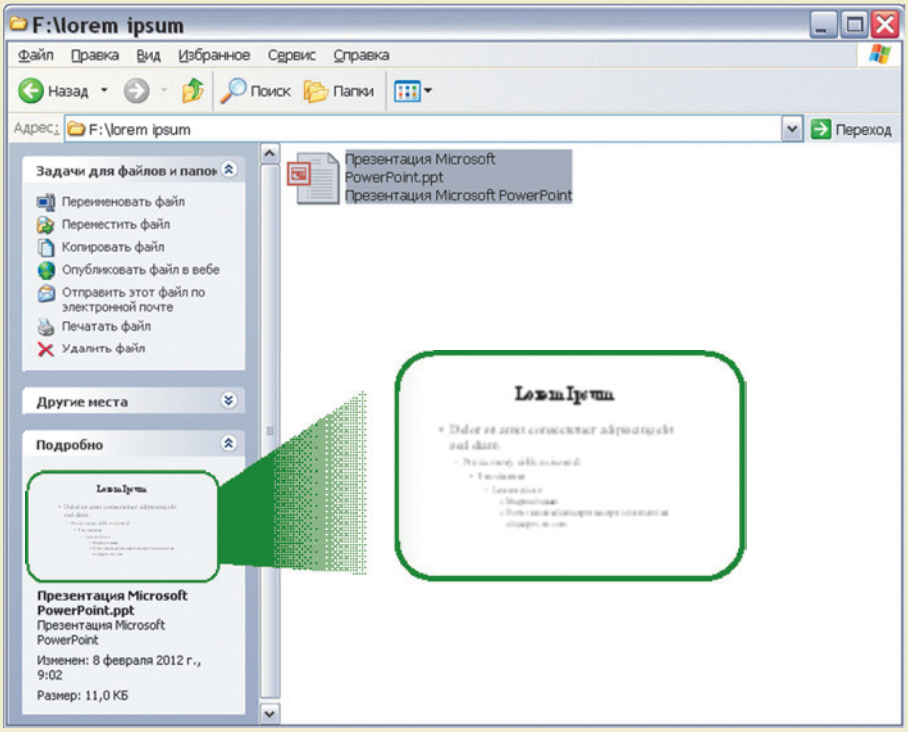


Рис. 2

А теперь — интересный факт, который можно заметить при известной доле внимательности: если создать таким способом пустую презентацию Power Point, а затем щелкнуть мышью на созданной иконке, то слева в области просмотра появляется миниатюра (как это обычно и делается для презентаций, рисунков и цифрового видео), на которой, хотя и очень нечетко (из-за малого размера надписи и ее крупнозернистости), проступают два таинственных слова: “Lorem Ipsum” (*рис.* 2). То же самое можно увидеть и непосредственно в самом окне папки, если установить для нее вид иконок **Эскизы страниц**. Но если раскрыть созданную презентацию на редактирование, то там никаких таких слов нет (и даже нет ни одного слайда)!

Что же это за странные слова и почему они всегда появляются на миниатюре созданной вышеуказанным способом пустой презентации? Давайте разберемся.

Введем словосочетание “Lorem Ipsum” в любом поисковом сервере. Порывшись на найденных сайтах, нетрудно найти, что это — название классического текста — “рыбы”[1], который нередко используют дизайнеры (разработчики web-страниц, верстальщики и другие специалисты, чей труд связан с оформлением текстов). Такой текст может быть и вполне бессмысленным, а цель его использования — просто продемонстрировать используемые эффекты оформления: шрифты, выравнивания, цвета и пр.

Текст “Lorem Ipsum” для этой цели был впервые применен кем-то из книгопечатников в XVI в. для набора шрифтовых образцов [3] и представляет собой сильно искаженный отрывок из философского трактата Цицерона “О пределах добра и зла”, написанного на латинском языке в 45 г. до н. э. Обычно этот “рыбный” текст имеет вид:

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fu-

[1] Аналогично, “рыбой” часто называют различные бланки, образцы, заготовки, которые кто-либо, не знающий, как сделать тот или иной документ, просит у более сведущих коллег (в разных странах такие заготовки документов называют по-разному, например, в Украине их часто называют “козой”). Почему такой “черновой” текст называют “рыбой” и откуда пошло это название — вопрос особый. По некоторым предположениям, рыбий скелетик с хребтом и ребрами чем-то напоминает по смыслу структуру будущего документа, на которую потом “наращивается мясо” в виде нужного содержания. Но в Интернете (например, на форуме *http://forum.lingvo.ru/actualthread.aspx?tid=118149*) можно найти и указание, что термин пришел когда-то от музыкантов, у которых “рыбой” называлось “мычание” сочиняемой мелодии “без голоса”. Например, композитор Б.А. Александров при создании гимна СССР (мелодия которого теперь “унаследована” и нынешним гимном России) давал поэту С.В. Михалкову следующее задание на стихотворный текст: “Нужны стихи с таким размером:

Карась, краснопёрка, карась, краснопёрка,
Карась, краснопёрка, уклейка, карась...”,

и по этой “рыбе” Михалков писал:

“*Союз нерушимый республик свободных*
Сплотила навеки великая Русь...”

giat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Такой “рыбный” текст намеренно делается непонятным для читателя: тогда в нем ничего не отвлекает от визуальной оценки используемых шрифтов, распределения текста на странице (макета) и различных дизайнерских изысков. И вместе с тем, текст кажется достаточно реалистичным, создавая впечатление финальной публикации [1].

Знатоки латинского языка могут при желании заняться переводом приведенного выше фрагмента, — хотя считается, что он по содержанию полностью бессмысленнен, в нем все же (как указывают некоторые авторы заметок на сайтах, например, [4]) может встретиться и что-то любопытное, ибо за многие годы модификации этого текста шутники, пользуясь тем, что большинство потребителей их продукции латыни не знают, чего только туда не повписывали… ☺

Итак, о том, что значат эти слова в миниатюре презентации, мы узнали. Но на этом сама “рыбная” проблема, которую мы ненароком “подцепили на крючок”, не исчерпывается.

Одна из проблем, возникающих при использовании текста Lorem Ipsum в качестве “рыбы”, заключается в том, что в нем частота использования букв различна, а заглавных букв очень мало (только в начале предложений), поэтому данный текст не дает полного представления о внешнем виде всех символов используемого шрифта во всех их возможных сочетаниях (парах расположенных рядом букв). Поэтому в качестве хорошего “рыбного” текста лучше выбирать текст, содержащий по возможности все буквы и типичные для используемого языка специальные знаки.

Такой текст, в котором содержатся все буквы алфавита, называют *панграммой*, а самый популярный пример такой панграммы для русского языка — это следующая строчка:

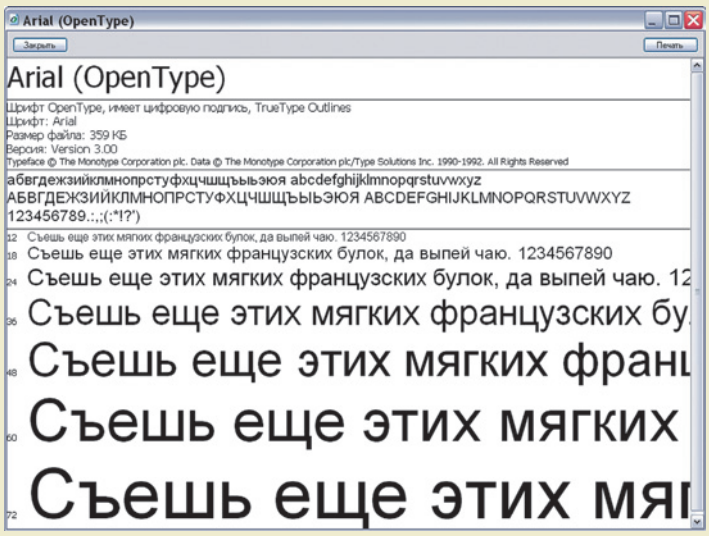


Рис. 3

съешь же еще этих мягких французских булок, да выпей чаю

Эта фразочка знакома, наверное, любому пользователю компьютера: ведь именно ее (правда, в усеченном виде: “*съешь еще этих мягких французских булок, да выпей чаю*”, т.е. с потерей буквы “*ж*”) фирма Microsoft использовала в своей утилите для отображения шрифтов в ОС Windows (*рис.* 3).

Интересно, что “генератор” панграммы про чай и булочки встроен в Microsoft Word: достаточно просто в нужном месте текстового документа напечатать **`=rand()`** и нажать клавишу Enter, и в этом месте будет вставлено три абзаца (каждый — из пяти предложений) этой “рыбы”. Можно задать количество требуемых фраз и явно: **`=rand(n,m)`**, где числа *n* и *m* указывают, соответственно, количество абзацев и количество предложений в каждом абзаце. А вот в версии Microsoft Office 2007 есть аналогичная функция **`=lorem(n,m)`**, которая генерирует тот самый “рыбный” текст Lorem Ipsum (значения *n* и *m* здесь указывают, какую часть текста напечатать и как объединять предложения в абзацы: *m* — количество предложений в абзаце, *n* — количество таких абзацев).

Вот, оказывается, как далеко нас завел поиск ответа на, казалось бы, простую загадку “таинственных слов” в PowerPoint. А тем из вас, кого сказанное особенно заинтересовало, предлагаем посмотреть и другие панграммы на различных языках, которые приведены на сайте Артемия Лебедева [0], а потом попробовать придумать свои панграммы. Главные требования: текст должен содержать *все* без исключения буквы алфавита рассматриваемого языка и быть как можно более коротким. А если это будет осмысленная и легкозапоминающаяся фраза, то еще лучше! Свои творения присылайте в редакцию и, — кто знает? — может быть, именно ваша панграмма появится где-нибудь в будущих версиях компьютерных программ… ☺

Литература

1. Что такое текст-“рыба”? [электронный ресурс]. URL: *http://www.blindtextgenerator.com/ru/about-lorem-ipsum*

2. Форум Abbyy Lingvo [электронный ресурс]. URL: *http://forum.lingvo.ru/actualthread.aspx?tid=118149*

3. Lorem Ipsum // Википедия [электронный ресурс]. URL: *http://ru.wikipedia.org/wiki/Lorem_ipsum*

4. *Лебедев А.* Ководство. § 67. Lorem ipsum [электронный ресурс]. URL: *http://www.artlebedev.ru/kovodstvo/sections/67*

5. *Лебедев А.* Ководство. § 33. Панграммы [электронный ресурс]. URL: *http://www.artlebedev.ru/kovodstvo/sections/33*

Педагогический университет
«ПЕРВОЕ СЕНТЯБРЯ»
предлагает

Лицензия Департамента образования
г. Москвы 77 № 000349,
рег. № 027477 от 15.09.2010

ДЛЯ УЧИТЕЛЯ ИНФОРМАТИКИ

ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ
ВНЕ ЗАВИСИМОСТИ ОТ МЕСТА ПРОЖИВАНИЯ
(обучение с 1 сентября 2012 по 31 мая 2013 года)

КОД	ПРОФИЛЬНЫЕ КУРСЫ
07-001	*И.Г. Семакин.* **Информационные системы в базовом и профильном курсах информатики**
07-008	*А.Г. Гейн.* **Математические основы информатики**
DVD 07-009	*С.Л. Островский.* **Основы web-программирования для школьного «сайтостроительства»**
07-010	*А.Г. Кушниренко, А.Г. Леонов.* **Методика преподавания основ алгоритмизации на базе системы «Кумир»**

КОД	ОБЩЕПЕДАГОГИЧЕСКИЕ КУРСЫ
21-001	*С.С. Степанов.* **Теория и практика педагогического общения**
21-002	*Н.У. Заиченко.* **Методы профилактики и разрешения конфликтных ситуаций в образовательной среде**
21-003	*С.Н. Чистякова, Н.Ф. Родичев.* **Образовательно-профессиональное самоопределение школьников в предпрофильной подготовке и профильном обучении**
21-004	*М.Ю. Чибисова.* **Психолого-педагогическая подготовка школьников к сдаче выпускных экзаменов в традиционной форме и в форме ЕГЭ**
DVD 21-005	*М.А. Ступницкая.* **Новые педагогические технологии: организация и содержание проектной деятельности учащихся**
DVD 21-007	*А.Г. Гейн.* **Информационно-методическое обеспечение профессиональной деятельности педагога, педагога-психолога, работника школьной библиотеки**
21-008	*А.Н. Майоров.* **Основы теории и практики разработки тестов для оценки знаний школьников**
21-009	*В.Д. Шадриков, И.В. Кузнецова, М.Д. Кузнецова.* **Формирование и оценка профессиональных качеств современного педагога**

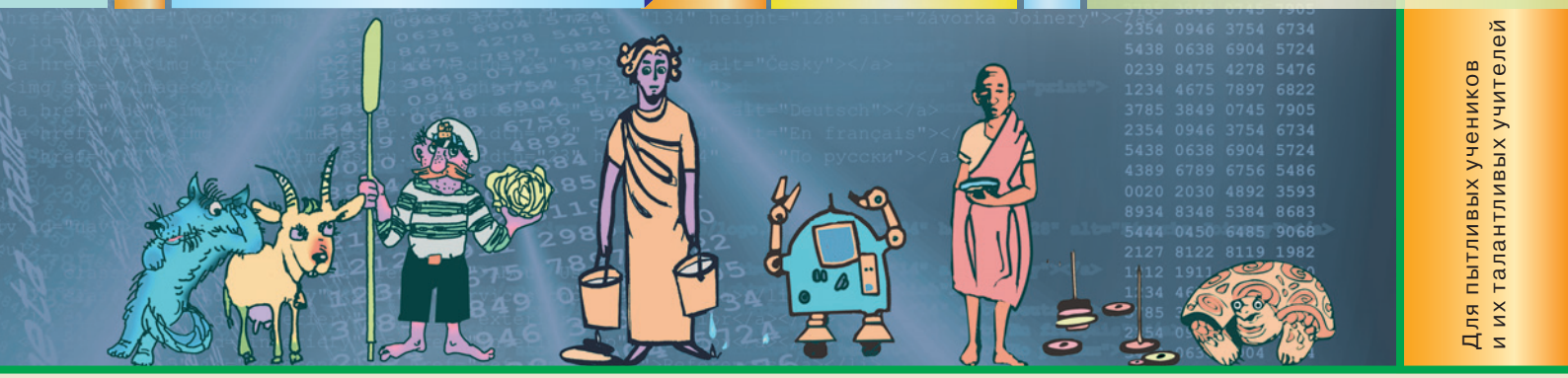
Имеются два варианта учебных материалов дистанционных курсов: брошюры и брошюры+DVD.
Курсы, включающие видеолекции (DVD), помечены значком DVD.
Нормативный срок освоения каждого курса – 72 часа. Дополнительная информация – на сайте **edu.1september.ru**
Окончившие дистанционные курсы получают удостоверение установленного образца.
Базовая стоимость курса (без учета скидок) составляет 2190 руб. для курсов без видеоподдержки и 2390 руб. – для курсов с видеоподдержкой.

ОЧНЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ
ДЛЯ ЖИТЕЛЕЙ МОСКВЫ И МОСКОВСКОЙ ОБЛАСТИ
(обучение с 1 октября по 30 декабря 2012 года)

Я.Н. Зайдельман. **Алгоритмизация и программирование: от первых шагов до подготовки к ЕГЭ**

Нормативный срок освоения каждого курса – 72 часа.
Дополнительная информация – на сайте **edu.1september.ru** и по телефону (499) 240-02-24 (звонки принимаются с 15.00 до 19.00).
Окончившие очные курсы получают удостоверение государственного образца.
Базовая стоимость курса (без учета скидки) – 5900 руб.

Электронную заявку можно в режиме online подать на сайте edu.1september.ru. Это удобно и просто!



ЛИЧНОСТИ

“Осязательная арифметика” Николаса Сондерсона

В.В. Шилов, Москва

Сегодня компьютеры все шире используются для помощи людям с ограниченными возможностями. Например, речевой интерфейс позволяет пользователю давать команды голосом и, наоборот, получать информацию в устной форме (компьютер может, в частности, прочитать текст из введенного в него файла или же произнести результаты вычислений). Микропроцессоры, управляющие сложными манипуляторами, возвращают людям утраченную подвижность и т.д.

Однако мало кто помнит, что попытки создать вычислительные приборы, которыми могли бы пользоваться люди с дефектами слуха и зрения, имеют весьма давнюю историю. В частности, можно вспомнить, что еще в середине XVIII в. французский педагог Хакоб Родригес Перейра (1715–1780) изобрел арифметическую машину, с помощью которой успешно обучал глухонемых детей четырем действиям арифметики. Методы Перейры удостоились похвального отзыва парижской Академии наук, а сам он был избран членом Лондонского королевского общества. К сожалению, о машине Перейры известно мало, нет даже ни одного ее изображения.

Х.Р. Перейра

Но еще раньше в Англии Николасом Сондерсоном был создан совершенно уникальный счетный инструмент.

Николас Сондерсон родился в январе (точная дата неизвестна) 1682 г. в деревушке Турлстон (графство Йоркшир), расположенной километрах в 20 к северо-западу от Шеффилда. Его отец служил акцизным чиновником, но, помимо жалованья, имел и другие источники дохода, так что семья, в которой Николас стал первенцем, была вполне обеспеченной. Однако наслаждаться безмятежным детством ребенку не пришлось. В возрасте одного года Николас перенес оспу, которая не только изуродовала его лицо, но и лишила зрения.

Специальной системы образования для слепых в то время не существовало, поэтому единственной возможностью получить знания для ребенка было систематическое чтение вслух. Николасу повезло — он был окружен вниманием и заботой родителей и многочисленных друзей семьи, так что недостатка в желающих помочь не было. Позднее его стали водить в бесплатную школу в соседнем городке, и мальчик впитывал в себя все услышанное на уроках. В это же время отец, служба которого требовала обширных вычислений, начал приобщать его к арифметике. Он обучил сына правилам счета, и уже очень скоро тот мог производить в уме длительные и сложные вычисления. При этом Николас вырабатывал свои собственные приемы счета, зачастую более эффективные, чем те, что предлагали ему учителя. В школе его особенно интересовала математика, хотя не менее блестяще он освоил древние и иностранные языки (так что сочинения Евклида, Архимеда и Диофанта он читал в подлиннике).

Однако о поступлении в университет даже речи не могло быть, и после завершения школьного курса Сондерсон продолжил самообразование. Друзья читали ему математические работы, но скорее всего он так и остался бы в науке дилетантом, если бы не встреча с математиком-любителем Ричардом Вестом, под руководством которого юноша приступил к серьезному изучению алгебры и геометрии. Спустя несколько лет

его познания в математике уже были столь обширными, что многие советовали ему обратиться в Кембридж. В 1707 г. один из друзей Николаса, студент, предложил ему разделить свою комнату в университетском общежитии, и Сондерсон переехал в Кембридж. Здесь он получил разрешение посещать лекции и пользоваться библиотекой, но, чтобы официально стать студентом, семейных средств было недостаточно. И в это время решающую роль в судьбе Николаса сыграл тогдашний люкасовский профессор[1] Уильям Уистон, занявший кафедру после И.Ньютона. В беседе с ним Сондерсон выразил желание стать преподавателем, и тот, впечатленный обширными знаниями и блестящими способностями собеседника, предоставил ему такую возможность. С 1707 г. Сондерсон начал вести занятия, причем, поскольку по тогдашней традиции преподаватель должен был являться универсалом, он читал не только математику, но и гидростатику, механику, оптику, акустику, астрономию и другие науки. Он оказался блестящим лектором, послушать которого стекались толпы студентов — случалось, что аудитория не могла вместить всех желающих, и лекцию приходилось повторять.

Николас Сондерсон

Его авторитет быстро рос, и когда 10 октября 1710 г. Уистон был снят с должности (причиной этого послужили его неортодоксальные взгляды на природу Троицы), единственным достойным его преемником в глазах большей части кембриджской общественности являлся Николас Сондерсон. Существовала и оппозиция, но она была не слишком велика. Однако назначению препятствовало то, что он не имел ученой степени и более того, даже никогда не был студентом! Обойти традицию было невозможно, и главы кембриджских колледжей подали прошение канцлеру университета герцогу Сомерсетскому о присуждении Сондерсону ученой степени. О том же просила и королева Анна. 19 ноября 1711 г. Сондерсон был произведен в магистры искусств. Уже на следующий день он был назначен люкасовским профессором математики, четвертым по счету, а 21 января 1712 г. прочитал на блестящей латыни вступительную лекцию, которая, по словам одного из присутствовавших, соответствовала лучшим образцам римского ораторского искусства и была достойна Цицерона.

Вся дальнейшая жизнь Сондерсона была связана с университетом. Он продолжал читать лекции, вел обширную переписку с ведущими математиками страны. Хотя Сондерсон не публиковал научных трудов, его авторитет продолжал расти, а известность простиралась далеко за пределы университета. В 1719 г. его избрали членом Королевского общества. В 1728 г. посетивший Кембридж король Георг II встретился с необычным преподавателем и, пораженный не только умом, но и мужеством собеседника, пожаловал Сондерсону степень доктора права. Однако выпадавшая на него нагрузка была слишком велика даже для этого от природы крепкого человека. Обычно Сондерсон читал лекции в течение семи или восьми часов в день, все остальное время он посвящал общению с учениками. Малоподвижный образ жизни также сказывался, и в 1733 г. Сондерсон впервые тяжело заболел. После этого здоровье Сондерсона так и не восстановилось, зимой 1739 г. у него началась гангрена, и 19 апреля ученый скончался.

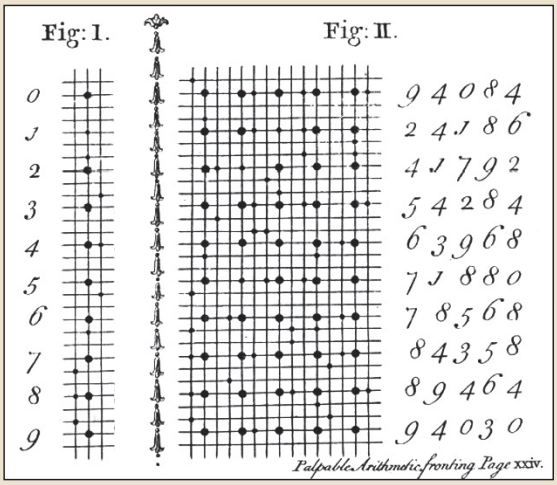
Через два года после смерти Сондерсона впервые были изданы его сочинения. Книге был предпослан очерк, содержавший описание изобретенного им уникального счетного инструмента. Очерк был написан известным математиком Джоном Колсоном, другом Сондерсона, который после его смерти стал пятым люкасовским профессором. О счетном приборе Сондерсона вспоминали многие современники, однако очерк Колсона — единственное подробное свидетельство о нем.

Понятно, что выполнение в уме сложных вычислений представляет немалую трудность, а ведь научные занятия Сондерсона требовали производить их в большом количестве. На помощь лишенному зрения ученому, как и во многих других ситуациях, приходили другие органы чувств. Известно, что Сондерсон обладал музыкальным слухом, хорошо играл на флейте. Острота его слуха была такова, что он мог оценивать размеры комнаты, расстояние до стен. Столь же чувствительным было и его осязание: вспоминают, что он на ощупь определял среди античных монет подделки, которые не сумели распознать даже эксперты.

Именно осязание помогло Сондерсону решить проблему вычислений и построить свою систему счета, которую Колсон назвал “осязательной арифметикой” (*palpable arithmetic*). Его счетный прибор представлял собой гладкую тонкую доску размером чуть более 30×30 см, по краям окаймленную рамкой. По вертикали и горизонтали доска была расчерчена множеством параллельных линий, про-

[1] Эта кафедра была учреждена в 1663 г. на средства члена парламента Генри Люкаса и традиционно возглавлялась видными математиками. Так, в 1669–1701 гг. ее вторым профессором был Исаак Ньютон. Избрание на эту должность является знаком признания весомых научных заслуг.

веденных на равном расстоянии друг от друга. По краям доски с интервалом около половины дюйма (1,25 см) были сделаны зарубки (между парой зарубок помещались 5 параллельных линий). Таким образом, каждый квадратный дюйм содержал 100 маленьких квадратов.



Абак Сондерсона

В точках пересечения линий были просверлены тонкие отверстия, в которые можно было втыкать булавки двух видов — с большой и маленькой головками. Для представления одной цифры использовались 4 смежных квадрата. Каждая цифра (кроме 1) задавалась с помощью большой булавки, воткнутой в точке соединения всех четырех квадратов, а также маленькой булавки, воткнутой в одно из отверстий по периметру указанных квадратов. Число 1 представлялось посредством маленькой булавки в центре (см. левую часть рисунка).

Таким образом, большие булавки позволяли отличить одну цифру от другой. Все цифры находились на равном расстоянии друг от друга, что давало возможность дополнительно контролировать правильность записи, длину числа и т.д. Вероятно, зарубки по краям также служили для контроля. Цифры Сондерсон различал на ощупь, легко касаясь булавок кончиками пальцев.

Обычно перед началом работы кто-либо помогал Сондерсону, подготавливая доску — “заполняя” ее нулями (т.е. помещая большие булавки во всех позициях, через две линии на третьей). Теперь для начала вычислений Сондерсону требовалось только дополнить каждую цифру, поместив в нужной позиции маленькую булавку (а в случае единицы — заменить большую булавку маленькой). Множество булавок обоих видов в специальных коробочках всегда находились у него под рукой. По свидетельству восхищенных очевидцев, Сондерсон управлялся с булавками с невероятной быстротой и легкостью. Он упорядочивал числа на счетном столе, складывал, вычитал, умножал и т.д., причем мог прервать процесс вычислений и возобновить его спустя какое-то время.

Колсон писал, что, кроме описанной счетной доски (которую он называет также *счетным столом* и *абаком*), в распоряжении Сондерсона были еще несколько отдельных досок с записанными на них специальными, предварительно вычисленными, математическими таблицами (Колсон говорит о четырех таких досках). Их назначение Колсон определить не сумел, но предположил, что они имели какое-то отношение к тригонометрическим вычислениям. Каждая из таблиц представляла собой прямоугольный брусок размером приблизительно 28×13×1,25 см. Обе большие противоположные грани брусков были, как и на счетной доске, разбиты вертикальными и горизонтальными линиями на квадраты, однако отверстия в них были просверлены только в одной из позиций, и в каждое из них была воткнута булавка. На одной грани, таким образом, помещалось по 9 маленьких таблиц из десяти пятизначных целых чисел (одна из таких таблиц изображена в правой части рисунка).

Интересно, что, по мнению Колсона, тот же абак служил Сондерсону и для занятий геометрией. С помощью булавок и натянутых между ними нитей он мог представлять не только прямые и ломаные линии, но и различные фигуры. Сондерсон часто сожалел, что в детстве не успел научиться писать, хотя нет никаких сомнений, что при необходимости он мог бы сделать это и в зрелом возрасте. Он не только знал буквы, но и различал их на ощупь, например, мог читать надписи, высеченные на надгробиях. Поэтому Колсон полагал, что Сондерсон мог иметь в своем распоряжении специальные булавки-литеры, соответствующие буквам и математическим знакам, и использовать их для обозначения углов, вершин геометрических фигур, а также для решения алгебраических задач. В любом случае достоверно известно, что Сондерсон умел решать различные уравнения и производить сложные вычисления. Но Колсон все-таки уточнил, что не берется судить, когда именно тот использовал для этого свои механические приспособления, а когда “полагался на силу мысли”…

Литература

1. *Colson, John*. Dr. Saunderson’s Palpable Arithmetic Decypher’d / The Elements of Algebra, in Ten Books: By Nicholas Saunderson. Volume The First. Cambridge. Printed at the University-Press: MDCCXLI. Pp. XX–XXVI.

ПОИСК ИНФОРМАЦИИ

Три города и страна

1. Какой город заменил Лондон на съемках нашего сериала о Шерлоке Холмсе?

2. С какой столицей связан провал Второго крестового похода?

3. В исполком какого города как-то нагрянули сразу трое “детей лейтенанта Шмидта”?

4. В какой единственной европейской стране нет своих источников пресной воды?

Ответы на эти вопросы найдите в Интернете или по другим источникам информации. Определите также, в каком городе установлен памятник, изображенный справа[2], и какое отношение он имеет к перечисленным вопросам?

Ответы присылайте в редакцию (можно отвечать не на все вопросы).

Уважаемые коллеги!

Для поощрения самых активных участников конкурсов, проводимых в разделе “В мир информатики”, редакция может направить вам электронный вариант диплома.

Заявку на диплом просьба прислать в апреле–мае 2012 года в адрес редакции (*vmi@1september.ru*). В школы, не имеющие доступа в Интернет, могут быть направлены дипломы учащимся согласно заявке обычной почтой.

ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА

Счетные бирки

Александр Слипенчук,
ученик гимназии № 1530 г. Москвы

Из книги [1] я узнал, что когда-то в России применялись так называемые “счетные бирки”, и заинтересовался, как же на них проводились расчеты. К сожалению, информации о бирках очень мало. Кроме книги [1], имеется также статья [2]. Поэтому большой удачей оказался тот факт, что в Российской государственной библиотеке для молодежи имеются счетные бирки, и благодаря заведующему сектором изучения электронных книг Исследовательского центра “Библиотека. Чтение. Интернет” этой библиотеки Александру Владиленовичу Пурнику у меня появилась возможность внимательно ознакомиться с ними.

Бирки представляют собой бруски, объединенные в связки.

На гранях бирок имеются зарубки разной формы, и каждая зарубка имеет свое значение. Можно выделить зарубки следующего вида:

1) | — обозначение единицы;

2) / и \ — обозначение числа 5;

3) X — обозначение числа 10;

4) ˈ — соответствует значению 0,5.

Кроме зарубок разной формы, имеются глубокие зарубки и в виде неглубоких царапин. Знак в виде глубокой зарубки имеет значение в 100 раз большее, чем тот же знак в виде царапины.

Значения на бирках получаются в виде суммы значений отдельных зарубок, причем зарубки одного и того же вида могут повторяться в разных “местах” числа. Например:

||XX /||||/

Все связки бирок можно разделить на две группы.

[2] Изображение в увеличенном размере представлено на диске к данному номеру журнала.

В первой группе зарубки выполнены на трех сторонах бирки. Можно выявить в обозначенных числах систему. На одной (первой) стороне каждой бирки обозначены числа 1, 1,5, 2, 2,5, 3 и другие, на второй и третьей сторонах — результат умножения чисел на первой стороне на 4,5 и на 19 соответственно (или на 49 и 232 соответственно).

Во второй группе бирок зарубки имеются на всех четырех гранях. Проведенный анализ бирок этой группы не позволил выявить какую-то систему. Возможно, это связано с недостаточным количеством бирок (и эта группа бирок требует дальнейшего анализа с использованием новых экземпляров).

В целом можно сделать вывод, что счетные бирки представляют собой не расчетный прибор, с помощью которого можно проводить вычисления, как, например, палочки Непера [3], а своеобразные таблицы с результатами расчета, которые могли использоваться для определения, например, размера налога.

В заключение выражаю благодарность директору Музея истории вычислительной техники гимназии, в которой я учусь, Дмитрию Михайловичу Златопольскому, за помощь, оказанную при подготовке данной статьи.

Литература

1. *Апокин И.А., Майстров Л.Е.* Развитие вычислительных машин. М.: Наука, 1974.

2. *Куканов М.А.* Мордовская народная математика. *www.edurm.ru/files/File/mordva/mathematic.doc*

3. Палочки Непера. / Информатика, № 3/2012.

От редакции. Дорогие читатели! Присылайте в редакцию результаты работы над исследовательскими проектами по информатике. Лучшие проекты мы опубликуем.

ЗАДАЧНИК

> Задача, которую вы решаете, может быть очень скромной, но если она бросает вызов вашей любознательности и если вы решаете ее собственными силами, то вы сможете испытать ведущее к открытию напряжение ума и насладиться радостью победы.
>
> *Джордж Пойя*

Ответы, решения, разъяснения к заданиям, опубликованным ранее

Задача “Три одноклассника”

Напомним, что необходимо было определить, кто из трех одноклассников — Влад, Тимур и Юра, встретившихся спустя 10 лет после окончания школы, чем любит заниматься в свободное время и у кого какая профессия, если один из них стал врачом, другой физиком, а третий юристом. Кто-то полюбил туризм, кто-то — бег, страсть третьего — регби. Юра сказал, что на туризм ему не хватает времени, хотя его сестра — единственный врач в семье, заядлый турист. Врач сказал, что он разделяет увлечение коллеги. Интересно, что у двоих из друзей в названиях их профессий и увлечений не встречается ни одна буква их имен.

Решение

Здесь исходные данные разбиваются на тройки (имя — профессия — увлечение).

Из слов Юры ясно, что он не увлекается туризмом и он не врач. Из слов врача следует, что он турист:

Имя	Юра		
Профессия		Врач	
Увлечение		Туризм	

Буква “а”, присутствующая в слове “врач”, указывает на то, что Влад тоже не врач, следовательно, врач — Тимур. В его имени есть буквы “т” и “р”, встречающиеся в слове “туризм”, следовательно, второй из друзей, в названиях профессии и увлечения которого не встречается ни одна буква его имени, — Юра. Юра не юрист и не регбист, так как в его имени содержатся буквы “ю” и “р”. Следовательно, окончательно имеем:

Имя	Юра	Тимур	Влад
Профессия	Физик	Врач	Юрист
Увлечение	Бег	Туризм	Регби

Ответ. Влад — юрист и регбист, Тимур — врач и турист, Юра — физик и бегун.

Правильные ответы представили:

— Аветисян Мариам и Сорокина Анна, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Агафонова Вероника, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Андрющенко Александр, Куценко Евгения, Парамонова Анастасия 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.**;

— Воробьева Валерия, Катюрина Александра и Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Глазкова Екатерина, Республика Коми, г. Сыктывкар, МОУ “Лицей народной дипломатии”, учитель **Гранаткина О.М.**;

— Голик Екатерина, Егорова Ксения, Засухина Елена, Миноцкий Ян, Пономарева Татьяна, Синицын Никита, Телегин Дмитрий и Юматова Светлана, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Зюзиков Дмитрий и Шепелев Константин, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Костюнин Александр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Петренко Анастасия, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Царев Иван, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Шадрина Юлия, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Задача “Шесть карточек”

Напомним, что следовало определить, сколько всего существует цепочек из шести карточек с буквами *А*, *Б*, *Е*, *Ж*, *И*, *К*, если каждая цепочка должна состоять из всех шести карточек и при этом должны соблюдаться правила:

1) любая цепочка начинается гласной буквой;

2) после гласной буквы не может снова идти гласная, а после согласной — согласная;

3) буквы в цепочке не должны повторяться.

Решение

1) для первой буквы имеются 3 варианта (*А*, *Е*, *И*);

2) в каждом из трех вариантов второй карточкой может быть карточка с любой из трех согласных букв (то есть всего будет 9 вариантов);

3) для третьей карточки в каждом случае имеется 2 варианта, так как одна гласная уже использована (общее число вариантов цепочек из трех карточек — $9 \times 2 = 18$);

4) для четвертой карточки в каждом случае имеется 2 варианта, так как одна согласная уже использована (общее число вариантов цепочек из четырех карточек — $18 \times 2 = 36$);

5) для пятой и шестой карточек есть единственные варианты (после любой гласной буквы должна стоять единственная оставшаяся согласная, а после любой согласной — единственная оставшаяся гласная). Значит, общее число вариантов не изменится.

Ответ: всего существует 36 таких цепочек.

Ответы прислали:

— Аветисян Мариам и Сорокина Анна, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Агафонова Вероника, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Алексеева Алена, Голик Екатерина, Егорова Ксения, Нигматуллина Юлия, Пономарева Татьяна, Синицын Никита и Юматова Светлана, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Антипов Анатолий, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Глазкова Екатерина, Республика Коми, г. Сыктывкар, МОУ “Лицей народной дипломатии”, учитель **Гранаткина О.М.**;

— Зюзиков Дмитрий и Шепелев Константин, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Катюрина Александра и Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Костюнин Александр и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Петренко Анастасия, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Селин Влад, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Яковенко Антон, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Задача “Наша любимая двойка”

Напомним, что необходимо было, используя пять раз цифру 2, знаки арифметических действий и скобки, записать выражение, значение которого будет равно:

1) 11;

2) 15;

3) 12 321.

Ответы:

1) $22/2 + 2 - 2 = 11$;

2) $(2 + 2)^2 - 2/2 = 15$ или $(2 \times 2)^2 - 2/2 = 15$ или $2^{2+2} - 2/2 = 15$ или $22/2 + 2 \times 2 = 15$ или $22/2 + 2^2 = 15$;

3) $(222/2)^2 = 12\,321$.

Ответы прислали:

— Агафонова Вероника, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Андрющенко Александр, Куценко Евгения, Парамонова Анастасия 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.**;

— Голик Екатерина, Егорова Ксения, Засухина Елена, Нигматуллина Юлия и Пономарева Татьяна, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Махонина Ирина, Караклинская средняя школа, Чувашская Республика, Канашский р-н, учитель **Макарова Л.Ф.**;

— Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Рочева Зоя, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Селин Влад, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Троекуров Василий, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Царев Иван, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Задача "Проложить маршрут"

Напомним, что необходимо было перечислить все маршруты, ведущиеся на шахматной доске из клетки c3 в клетку *a*1, соблюдая следующее правило: каждый ход делается либо на одну клетку влево, либо на одну клетку вниз, либо на одну клетку вниз и на одну клетку влево.

Ответ — возможны 13 маршрутов:

1) ЛЛНН;
2) ЛДН;
3) ЛНЛН;
4) ЛНД;
5) ЛННЛ;
6) ДЛН;
7) ДД;
8) ДНЛ;
9) НЛЛН;
10) НЛД;
11) НЛНЛ;
12) НДЛ;
13) ННЛЛ.

Ответы прислали:

— Аветисян Мариам и Сорокина Анна, Совхозная средняя школа, Московская обл., Серебряно-Прудский р-н, поселок Успенский, учитель **Жарикова Е.Н.**;

— Агафонова Вероника, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Аксенов Василий, Демьянова Елена, Костюнин Александр и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Алексеева Алена, Голик Екатерина, Нигматуллина Юлия и Пономарева Татьяна, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Грибанов Владлен, Дукач Светлана и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Диков Андрей, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Скакалин Игорь, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Хромченко Владимир, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Два "географических" числовых ребуса

Напомним, что необходимо было решить два числовых ребуса:

1. **ГАВ** = **АН**$^{\text{А}}$.
2. **СССР** = **Р**$^{\text{Ф}}$.

Решение ребуса № 1

Видно, что **А** < 3, так как уже 30^3 — пятизначное число. Но при этом **А** не может быть равно 1, значит, **А** = 2.

Проанализировав[3] квадраты чисел третьего десятка (20, 21, …, 29), можно обнаружить, что среди них есть три числа, вторая цифра которых равна 2:

$23^2 = 529$;
$25^2 = 625$;
$27^2 = 729$.

Второй вариант не подходит, т.к. в нем **Н** = **В**, третий — также (в нем **Н** = **Г**). Первый вариант подходит.

Итак, решение ребуса: $23^2 = 529$.

Решение ребуса № 2

Ясно, что **А** > 1. При этом **А** ≠ 2, так как даже 2^9 — трехзначное число. Остальные возможные значения цифры **А** можно исследовать, использовав электронную таблицу Microsoft Excel или другую. Целесообразно оформить лист для расчета значений А, равных 3, 4, …, 9 в степени 4, 5, …, 9 (см. с. 55).

Примечание. Куб числа 9 — трехзначное число, поэтому 2-ю и 3-ю степени можно не рассчитывать.

Анализ полученных результатов показывает, что решением ребуса является $6^5 = 7776$.

[3] Для этого можно использовать электронную таблицу Microsoft Excel или др.

	A	B	C	D	E	F	G	H
1			Степени					
2			*4*	*5*	*6*	*7*	*8*	*9*
3	**A=**	**3**	81	243	729	2187	6561	19683
4	**A=**	**4**	256	1024	4096	16384	65536	262144
5	**A=**	**5**	625	3125	15625	78125	390625	1953125
6	**A=**	**6**	1296	7776	46656	279936	1679616	10077696
7	**A=**	**7**	2401	16807	117649	823543	5764801	40353607
8	**A=**	**8**	4096	32768	262144	2097152	16777216	134217728
9	**A=**	**9**	6561	59049	531441	4782969	43046721	387420489

Правильные ответы представили:

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Андрющенко Александр, Куценко Евгения, Парамонова Анастасия 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.**;

— Голик Екатерина и Миноцкий Ян, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Катюрина Александра и Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Манохина Татьяна, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Решетников Виталий, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**;

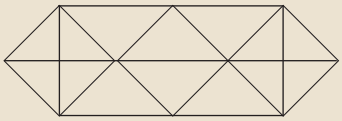
— Рочева Зоя, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Хромченко Владимир, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Головоломка “Сколько треугольников на рисунке?”

Напомним, что необходимо было определить число треугольников на следующем рисунке:



Правильный ответ: 28.

Ответы прислали:

— Алексеева Алена, Голик Екатерина, Миноцкий Ян, Пономарева Татьяна, Синицын Никита, Тананаева Анастасия и Юматова Светлана, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Ахмедьянова Алия, Галеева Эльвира, Докукина Снежана, Катюрина Александра, Костьянов Дмитрий и Кремнева Валерия, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Божко Юлия, Бондарев Вадим и Корчагин Александр, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Глазкова Екатерина, Республика Коми, г. Сыктывкар, МОУ “Лицей народной дипломатии”, учитель **Гранаткина О.М.**;

— Козлов Глеб, г. Ярославль, школа № 33, учитель **Голубина Л.И.**;

— Новиков Филипп и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Рочева Зоя, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Скакалин Игорь, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Селин Влад, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Царев Иван и Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Шадрина Юлия, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Задача “Бидоны с молоком”

Напомним, что следовало определить, сколько 50-литровых бидонов заполняет надоенное молоко, при условии, что если его разлить в 40-литровые бидоны, то понадобится на 5 бидонов больше, и один из них останется неполным, а если же это молоко разлить в 70-литровые бидоны, то понадобится на 4 бидона меньше, и один из них тоже останется не-

	A	B	C	D	E	F
1	Число 50 л бидонов	Общий объем молока	Сколько бидонов по 40 л	На сколько больше	Сколько бидонов по 70 л	На сколько меньше
2	1	50	1.25	0.25	0.71	0.29
3	2	100	2.5	0.5	1.43	0.57
4	3	150	3.75	0.75	2.14	0.86
…	…	…	…	…	…	…
26	25	1250	31.25	7	17.86	7.16

полным. Предлагалось решить задачу, используя электронную таблицу Microsoft Excel (допускалось и аналитическое решение).

Решение

Оформим лист электронной таблицы, как показано выше (необходимые формулы определите самостоятельно).

Из условия следует, что на листе разность P1 между числом 40-литровых бидонов и 50-литровых бидонов должна находиться в диапазоне $4 < P1 < 5$, а разность P2 между числом 50-литровых и 70-литровых бидонов — аналогично: $4 < P2 < 5$ (убедитесь в этом!).

Анализ таблицы показывает, что указанным ограничениям соответствует строка для числа 50-литровых бидонов, равного 17.

Ответ: 17.

Аналитическое решение

Обозначим искомое число 50-литровых бидонов — x. Тогда общий объем молока $V = 50x$.

Далее составим табличку:

Бидоны	Их число	Объем молока в этом числе бидонов
40 л	$x + 5$	$40(x + 5) = 40x + 200$
70 л	$x - 4$	$70(x - 4) = 70x - 280$

Можем записать $40x + 200 > V$, так как на самом деле один из $(x + 5)$ бидонов неполный. Отсюда: $x < 20$.

Для 70-литровых бидонов: $70x - 280 > V$, поскольку один из $(x - 4)$ бидонов неполный. Решив неравенство, получим $x > 14$.

Итак, искомое значение — одно из чисел 15, 16, 17, 18, 19. Анализ показывает, что подходит число 50-литровых бидонов, равное 17.

Ответы представили:

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

— Ахмедьянова Алия, Галеева Эльвира, Докукина Снежана, Костьянов Дмитрий и Кремнева Валерия, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Володина Алина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Глазкова Екатерина, Республика Коми, г. Сыктывкар, МОУ “Лицей народной дипломатии”, учитель **Гранаткина О.М.**;

— Грибанов Владлен, Дукач Светлана и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Зуйков Денис и Куклев Константин, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Новиков Филипп и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Турубара Владимир, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Денис Зуйков, Екатерина Глазкова, Константин Куклев и читатели из школы № 8 поселка Стрелка, решившие задачу с помощью программы Microsoft Excel, будут награждены дипломами. Поздравляем!

Числовой ребус “Восстановить пример”

Напомним, что необходимо было заменить звездочки недостающими цифрами, чтобы пример на умножение был верен:

```
      * * 7
    × 3 * *
    -------
    * 0 * 3
    * 1 *
  * 5 *
  -------
  * 7 * * 3
```

Любой звездочкой может быть любая цифра.

Решение

Для удобства описания решения заменим звездочки буквами (при этом будем иметь в виду, что разные буквы могут соответствовать одним и тем же цифрам):

```
      A B 7
    × 3 C D
    -------
    E 0 F 3
    G 1 H
  I 5 J
  -------
  K 7 L M 3
```

1. Видно, что **D** может быть равно только 9.
2. **J** = 1 (последняя цифра произведения 3 на 7).
3. Запишем известные цифры в пример:

```
      A B 7
    × 3 C 9
    -------
    E 0 F 3
    G 1 H
  I 5 1
  -------
  K 7 L M 3
```

Теперь видно, что **E** = **G** = 1.

4. Так как **AB**7 × 9 = 10**F**3, то **A** = 1 и **B** = 1.
Окончательный вариант:

	×	1	1	7
		3	1	9
3	7	3	2	3

Правильные ответы представили:

— Аксенов Василий, Демьянова Елена, Костюнин Александр и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Андрющенко Александр, Куценко Евгения, Парамонова Анастасия 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.**;

— Воробьева Валерия, Катюрина Александра и Махмутов Роберт, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Голик Екатерина и Головченко Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Диков Андрей и Филимонова Галина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Рочева Зоя, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

— Царев Иван, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Головоломка «Получить из “СОКОЛ” — “КОЛОС”»

Напомним, что необходимо было, перемещая карточки с буквами слова “СОКОЛ”, получить слово “КОЛОС”.

Ответы прислали:

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Володина Алина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

— Манохина Татьяна, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

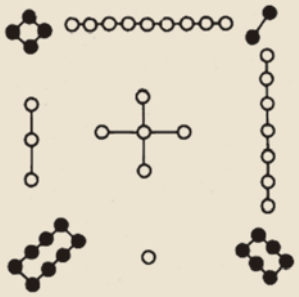
— Толстикова Елена, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

Задача решается за 34 перекладывания (возможны несколько вариантов решения).

Головоломка “Странное изображение”

Напомним, что предлагалось прокомментировать следующее изображение:



Ответ

На рисунке изображен вариант так называемого “магического квадрата” — квадрата из 9 клеток с числами, в котором сумма чисел в каждой строке, в каждом столбце и в каждой диагонали одинаковая:

4	9	2
3	5	7
8	1	6

(Числа на изображении представлены кружочками.)

Ответы прислали:

— Андронова Алина, Батракова Полина, Белоусов Сергей, Дукач Светлана, Грибанов Владлен, Зазнобина Екатерина, Коротнева Ирина, Королева Анна, Ливинская Виктория, Максимова Ксения, Самойлова Дарья, Селенгина Юлия, Соболев Иван, Степанова Мария, Тин-да-лин Анна и Шорохова Полина, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

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

— Володина Алина, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Герасимова Мария, Махонина Ирина и Чебукова Людмила, Караклинская средняя школа, Чувашская Республика, Канашский р-н, учитель **Макарова Л.Ф.**;

— Нигматуллина Юлия и Пономарева Татьяна, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Новиков Филипп и Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**;

— Яковенко Антон, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Задача "Три кучки спичек"

Напомним, что необходимо было из трех кучек спичек (11, 7 и 6), перекладывая спички из одной кучки в другую, сделать так, чтобы в каждой кучке было бы по 8 спичек. При этом требуется соблюдать следующее правило: к любой кучке разрешается дополнять ровно столько спичек, сколько в ней есть.

Решение

Первым ходом к первой кучке добавлять спички нельзя ни из какой другой, а ко второй — из третьей. Значит, возможны варианты:

1) из первой кучки — ко второй;

2) из первой кучки — к третьей;

3) из второй кучки — к третьей.

В последнем варианте после первого хода во второй кучке останется всего одна спичка, что не позволит получить требуемый результат за два оставшихся хода.

Анализ двух первых вариантов показывает, что решением является представленное в таблице:

Кучка	Начальное состояние	Первый ход	Второй ход	Третий ход
Первая	11	11 – 7 = 4	4	4 + 4 = 8
Вторая	7	7 + 7 = 14	14 – 6 = 8	8
Третья	6	6	6 + 6 = 12	12 – 4 = 8

Ответы представили:

— Алексеева Алена, Голик Екатерина, Синицын Никита и Телегин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Ахмедьянова Алия, Галеева Эльвира, Докукина Снежана, Катюрина Александра, Костьянов Дмитрий и Кремнева Валерия, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Грибанов Владлен, Дукач Светлана и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

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

— Божко Юлия, Бондарев Вадим и Корчагин Александр, основная школа поселка Михинский, Воронежская обл., Таловский р-н, учитель **Удалова А.А.**;

— Дмитриева В., Тобурдановская средняя школа, Чувашская Республика, Канашский р-н, учитель **Макарова Л.Ф.**;

— Зюзиков Дмитрий и Шепелев Константин, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Чернова Ксения, Республика Карелия, поселок Надвоицы, школа № 1, учитель **Каликина Т.В.**

В ряде ответов задача решалась за 4 хода.

Задача "Гастролер"

Напомним, что необходимо было определить, какого числа и какого месяца некий гастролер был в каждом из городов, если первый вторник месяца провел в Иркутске, первый вторник после первого понедельника — в Новосибирске, а в следующем месяце он первый вторник провел в Воронеже, а первый понедельник — в Москве?

Решение

Первый вторник месяца, который гастролер провел в Иркутске, не может быть 2-, 3-, ..., 7-го числа, так как при этом первый понедельник будет раньше на один день (то есть 1-, 2-, ..., 6-го числа), но тогда вторник, который он провел в Новосибирске, совпадет с датой, когда он был в Иркутске. Значит, в последнем городе он находился 1-го числа.

Для следующего месяца рассуждения — аналогичные. Но так, чтобы первые числа двух подряд идущих месяцев попадали на один и тот же день недели, может быть только в високосном году для февраля и марта (убедитесь в этом!).

Значит, ответ такой: гастролер находился 1 февраля в Иркутске, 8 февраля — в Новосибирске, 1 марта — в Воронеже и 8 марта — в Москве.

Ответы представили:

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

— Ахмедьянова Алия, Галеева Эльвира, Докукина Снежана, Костьянов Дмитрий и Кремнева Валерия, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

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

— Глазкова Екатерина, Республика Коми, г. Сыктывкар, МОУ "Лицей народной дипломатии", учитель **Гранаткина О.М.**;

— Голик Екатерина и Тананаева Анастасия, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Согомонян Серине, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

— Тюрникова Елена и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Шадрина Юлия, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

ШКОЛА ПРОГРАММИРОВАНИЯ

Одинаковые ли цифры?

Рассмотрим такую задачу: “Даны два двузначных числа. Определить, состоят ли они из одних и тех же цифр. Например, для чисел 51 и 15 ответ положительный, для чисел 22 и 72 — отрицательный”.

Такую задачу решить несложно. Нужно выделить цифры каждого числа, а затем сравнить их. Соответствующая программа на школьном языке программирования имеет вид:

```
алг Проверка
нач цел a, b, a1, a2, b1, b2
. |Вводим исходные числа
. вывод нс, "Введите два двузначных числа"
. ввод a, b
. |Выделяем цифры в каждом из них
. a1 := div(a, 10)
. a2 := mod(a, 10)
. b1 := div(b, 10)
. b2 := mod(b, 10)
. |Проверяем, совпадают ли цифры
. если a1 = b1 и a2 = b2
       или a1 = b2 и a2 = b1
. . то
. . . вывод нс, "Цифры одинаковые"
. . иначе
. . . вывод нс, "Есть отличающиеся цифры"
. все
кон
```

А если числа — трехзначные? В этом случае задача решается во многом аналогично рассмотренной. Также нужно выделить цифры (теперь — три) каждого числа, а затем также сравнить их. Конечно, вариантов для сравнения будет больше. Их можно сформировать так. Пусть цифры первого числа — $a1$, $a2$ и $a3$, второго — $b1$, $b2$ и $b3$. Тогда можно сравнивать цифры $a1$, $a2$, $a3$ с цифрами следующих сочетаний:

$b1, b2, b3$
$b1, b3, b2$
$b2, b1, b3$
$b2, b3, b1$
$b3, b1, b2$
$b3, b2, b1$

Соответствующая программа:

```
алг Проверка
нач цел a, b, a1, a2, a3, b1, b2, b3
  |Вводим исходные числа
  вывод нс, "Введите два
             3-значных числа"
  ввод a, b
  |Выделяем цифры в каждом из них
  a1 := div(a, 100)
  a2 := div(mod(a, 100), 10)
  a3 := mod(a,10)
  b1 := div(b,100)
  b2 := div(mod(b,100),10)
  b3 := mod(b,10)
  |Проверяем, совпадают ли цифры
  если a1 = b1 и a2 = b2 и a3 = b3 или
       a1 = b1 и a2 = b3 и a3 = b2 или
       a1 = b2 и a2 = b1 и a3 = b3 или
       a1 = b2 и a2 = b3 и a3 = b1 или
       a1 = b3 и a2 = b1 и a3 = b2 или
       a1 = b3 и a2 = b2 и a3 = b1
  . . то
  . . . вывод нс, "Цифры одинаковые"
  . . иначе
  . . . вывод нс, "Есть отличающиеся
                   цифры"
  . все
кон
```

Конечно, уже при 4-значных числах решение значительно усложняется, точнее, становится трудоемким — придется проверить 24 сложных условия, каждое из которых состоит из четырех простых! А если числа 10-значные?

Идея решения для таких случаев заключается в следующем. Нужно цифры каждого из проверяемых чисел записать в два массива, отсортировать элементы массивов (скажем, в порядке невозрастания), после чего проверить, совпадают ли элементы двух массивов, стоящие на одних и тех же местах. Как вам идея? Давайте реализуем ее на примере 5-значных чисел.

Заполнить массив *da* из пяти элементов цифрами 5-значного числа *a* можно так:

```
нц для k от 5 до 1 шаг -1
  |Последнюю цифру числа a
  |принимаем в качестве
  |k-го элемента массива
  da[k] := mod(a, 10)
  |Отбрасываем последнюю цифру
  a := div(a, 10)
кц
```

Для сортировки элементов полученного массива в порядке невозрастания можно применить так называемый “метод пузырька” (метод обмена [1]):

```
нц для i от 1 до 4
  нц для j от 1 до 4
    если da[j] < da[j + 1]
     то |Меняем соседние элементы
        всп := da[j]
        da[j] := da[j + 1]
        da[j + 1] := всп
    все
  кц
кц
```

Проверить, совпадают ли элементы двух массивов *da* и *db*, стоящие на одних и тех же местах,

можно, подсчитав количество таких совпадающих элементов:

```
к_совпад := 0
нц для i от 1 до 5
  если da[i] = db[i]
   то
     к_совпад := к_совпад + 1
  все
кц
```

После этого по значению величины *к_совпад* можно получить требуемый ответ.

Для массивов из большого количества элементов целесообразно использовать оператор цикла с условием (проводить сравнения элементов до момента появления двух разных цифр).

Всю программу “соберите” самостоятельно.

Задания для самостоятельной работы

Разработайте программы решения двух задач.

1. Даны два слова, состоящие из одинакового количества букв (количество букв известно заранее). Определить, можно ли из букв одного слова, используя каждую букву столько раз, сколько она встречается в нем, составить второе. Например, из слова “логика” можно составить слово “иголка”, из слова “австралопитек” можно составить “ватерполистка”, из слова “сумма” можно составить слово “самум”. Регистр букв не должен учитываться (то есть оба слова должны быть заданы в одном и том же регистре).

2. Даны два списка фамилий, в каждом из которых указаны по 10 человек, из одних и тех же фамилий. Все фамилии, естественно, начинаются с прописных букв. Проверить, состоят ли списки из одних и тех же фамилий.

Программы, пожалуйста, присылайте в редакцию.

Литература

1. *Златопольский Д.М.* Программирование: типовые задачи, алгоритмы, методы. М.: Бином. Лаборатория знаний, 2007.

“ЛОМАЕМ” ГОЛОВУ

Числовой ребус «“БЛОКНОТ” из трех “ЛИСТКОВ”»

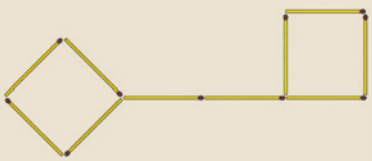
Решите, пожалуйста, числовой ребус:

	Л	И	С	Т	О	К
+	Л	И	С	Т	О	К
	Л	И	С	Т	О	К
	Л	И	С	Т	О	К
Б	Л	О	К	Н	О	Т

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

Переставить четыре спички

Задание предназначено для учащихся начальной школы и учеников 5–7-х классов.



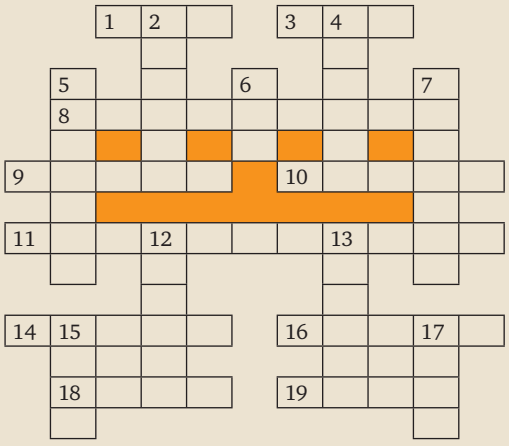
Переставьте четыре спички так, чтобы из изображения ключа получилось три квадрата.

61 монета, две фальшивые

Есть 61 монета одинакового внешнего вида. Известно, что две из них — фальшивые. Все настоящие одинакового веса, обе фальшивые — тоже одинакового веса, отличающегося от веса настоящих монет.

Как можно узнать с помощью трех взвешиваний на чашечных весах без гирь, легче или тяжелее настоящих фальшивые монеты? (Определять фальшивые монеты не требуется.)

Кроссворд



По горизонтали

1. … координат.
2. Характеристика файла, папки или переменной величины.
8. Число, определяющее систему счисления.
9. Знак, обозначающий число.
10. Реакция объекта на воздействие или запрос.
11. Операция, проводимая над фрагментом текста, файлом или папкой.
14. Место хранения резервных копий файлов.
16. Инструмент графического редактора Microsoft Paint.
18. Непрошенное рекламное сообщение, сетевой мусор.

19. Поименованная совокупность данных на носителе.

По вертикали

2. Устройство ввода информации в компьютер.

4. Предмет, создающий вокруг себя особое поле — область, в которой другие предметы подвергаются действию определенной силы.

5. Устройство для вывода информации в персональном компьютере.

6. Цифра десятичной системы счисления.

7. Арифметическая операция.

12. Значение переменной величины или константы логического типа.

13. Часть света, в которой будут выпускаться компьютеры "черной сборки".

15. Размер программиста по высоте.

17. Часть оператора цикла (другие операторы, повторяемые при выполнении цикла).

ВНИМАНИЕ! КОНКУРС

Конкурс "Графические редакторы" (повторно)

Такой конкурс (№ 89) был объявлен в октябрьском номере журнала "Информатика" за 2011 год. Благодаря Юлию Маслову, Сергея Морозова и Анастасию Царевскую, Ардатовское ПУ-104, поселок Ардатов Нижегородской обл., преподаватель **Зудин В.П.**, представивших ответы (Анастасия, Сергей и Юлия будут награждены дипломами), предлагаем читателям еще раз подумать над его вопросами. Напомним их.

1. В векторном редакторе нарисован квадрат со стороной в 10 пикселей. Изображение было сохранено в файле, размер которого составил *N* Кб. Разрешающая способность монитора при этом была равна 1024 на 768 пикселей. Чему будет равен размер графического файла, если сторону квадрата увеличить в 2 раза?

2. Файл в формате bmp содержал рисунок закрашенного квадрата со стороной в 10 пикселей. Рисунок загрузили в редактор и увеличили сторону квадрата в 5 раз. Как изменится размер графического файла? В обоих случаях изображение записывалось с глубиной цвета 24 бит на пиксель.

3. В растровом графическом редакторе было создано черно-белое прямоугольное изображение, которое сохранили в файле формата bmp. Размер файла составил 350 Кб. Разрешающая способность монитора при этом была равна 1024 на 768 пикселей. После этого она была изменена на 800 на 600 пикселей. Созданный файл был сохранен вновь. Чему будет равен его размер?

4. Какой графический редактор, растровый или векторный, точнее построит:

1) окружность радиусом в 100 пикселей;

2) окружность радиусом в 200 пикселей;

3) прямую линию с наклоном в 30 градусов?

5. В векторном редакторе было создано изображение с размерами на экране 150 × 300 пикселей. При этом разрешение экрана составляло 96 dpi. Затем изображение экспортировали в формат bmp с разрешением 300 dpi. Каковы будут размеры изображения (в пикселях) при выводе его на экран в этом формате? А при выводе на бумагу?

Обратим внимание на то, что вопросы, аналогичные приведенным, встречаются в заданиях ЕГЭ по информатике и ИКТ.

Ответы с обоснованием, пожалуйста, отправьте в редакцию. Срок представления — 1 октября. Можно отвечать не на все вопросы.

Итоги конкурса № 90

Напомним, что был предложен ряд заданий, связанных с логическими и сдвиговыми операциями над числами.

Ответы представили:

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

— Бородин Виктор, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Голик Екатерина и Тананаева Анастасия, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Довгань Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Измайлов Андрей, г. Орел, лицей № 4, учитель **Чапкевич И.М.**;

— Касатонова Мария, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Мартынов Андрей, Воронежская обл., поселок Каменка, средняя школа № 1 им. Героя Советского Союза В.П. Захарченко, учитель **Старикова М.Е.**;

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

— Проклов Виталий, г. Красноярск, лицей № 67, учитель **Кузнецов А.А.**;

— Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Учитывая, что тема конкурса является важной для предмета "информатика", приведем развернутые ответы на приведенные в условии конкурса задания и вопросы.

1.

а) 1101101 **AND** 10101:

1101101	1	1	0	1	1	0	1
10101	0	0	1	0	1	0	1
1101101 AND 10101	0	0	0	0	1	0	1

Ответ: 5_{10}.

б) 1011011 **OR** 11001:

1011011	1	0	1	1	0	1	1
11001	0	0	1	1	0	0	1
1011011 OR 11001	1	0	1	1	0	1	1

Ответ: 91_{10}.

в) 1001101 **XOR** 10111:

1001101	1	0	0	1	1	0	1
10111	0	0	1	0	1	1	1
1001101 XOR 10111	1	0	1	1	0	1	0

Ответ: 90_{10}.

2.

1) при нулевой маске результат операции **И** равен нулю;

2) результат операции **ИЛИ**:

а) при маске из всех единиц равен $2^k - 1$, где k — количество двоичных разрядов в записи данного числа:

	1	1	0	1
	1	1	1	1
ИЛИ	1	1	1	1

б) при маске в виде нуля равен данному числу:

	1	0	0	1
	0	0	0	0
ИЛИ	1	0	0	1

3) младший разряд двоичного числа позволяет выделить (получить) операция **И** с маской, равной 1:

	1	1	0	1
				1
И				1

и

	1	1	0	0
				1
И				0

4) если к какому-то числу дважды применить операцию исключающего **ИЛИ** с любой маской, то получится исходное число:

Число	1	1	0	0
Маска	1	0	0	1
XOR	0	1	0	1
Маска	1	0	0	1
XOR	1	1	0	0

5) если операцию исключающего **ИЛИ** применить к двум одинаковым числам, то получится 0;

6) если к какому-то числу применить операцию исключающего **ИЛИ** с маской из всех единиц, то получится результат, равный разности маски и заданного числа:

Число	1	1	0	0
Маска	1	1	1	1
XOR	0	0	1	1

3. Определить результат выполнения поразрядной логической операции **NOT** с целым десятичным числом N типа *byte* "в уме" можно — он равен $255 - N$.

Объяснение

Значение **not** N равно так называемому "инвертированному" числу N (обозначим его a). Тогда $N + a = 255$ (сумма 8-разрядного двоичного числа и инвертированного равна 11111111, то есть 255_{10}). Откуда десятичное значение not $N = 255 - N$.

4. Поразрядные логические операции к одному или двум отрицательным операндам применять можно.

5. В результате выполнения следующего фрагмента программы:

```
var a: byte; b: shortint;
BEGIN
a := 11;
writeln(not a);
b := 11;
writeln(not b)
END.
```

на экран будут выведены разные значения: 244 и –12. Первое значение обосновывается в статье.

Для значения b результат приведен в таблице:

	Двоичное представление							
11	0	0	0	0	1	0	1	1
not 11	1	1	1	1	0	1	0	0

Что за число получено? Так как тип данных указан shortint, то первый разряд — знаковый. То есть получено отрицательное число, дополнительный код которого равен 1110100 (7-разрядный).

Получить модуль отрицательного числа по его дополнительному коду можно следующим образом:

Дополнительный код	1	1	1	0	1	0	0
Вычитаем 1	1	1	1	0	0	1	1
Инвертируем	0	0	0	1	1	0	0
Десятичное представление модуля числа	12						

Итак, результат равен –12.

6. Равенство a **and** $-(a + 1) = 0$ (например, 51 **and** $-52 = 0$) имеет место по следующим причинам.

Как известно, целые отрицательные числа представлены в компьютере в так называемом "дополнительном" коде. Можно показать, что такой код отрицательного числа $(a + 1)$ представляет собой обратный код положительного числа a. Например, прямой код числа 51 при восьмиразрядной ячейке: 00110011. Для отрицательного числа $-(51 + 1) = -52$ дополнительный код можно определить так:

Прямой код числа 52	0	0	1	1	0	1	0	0
Обратный код	1	1	0	0	1	0	1	1
Дополнительный код (прибавляем 1)	1	1	0	0	1	1	0	0

Видно, что результат логической операции **and** над числами 00110011 и 11001100 будет равен нулю.

7. В результате выполнения оператора присваивания `a := a shr 1`:

1) при четном a: число a уменьшится в 2 раза;

2) при нечетном a: значение a станет равно a div 2, где div — знак операции целочисленного деления.

Обобщить оба ответа можно в виде: a div 2.

8. Число a, равное 1024, в результате выполнения оператора присваивания:

```
a := a shr k.
```

— уменьшится в 2^k раз.

9. Однозначно утверждать, что при поразрядном сдвиге влево заданное число a в общем случае увеличится или уменьшится, нельзя:

	Двоичное представление				Результат
a	1	0	1	0	10
***a* shl 1**	0	1	0	0	4
***a* shl 1 shl 1**	1	0	0	0	8

10. К отрицательным числам сдвиговые операции применять можно:

–6 shl 1 = –12;

–6 shr 1 = 2 147 483 645.

Приведенные результаты исследуйте самостоятельно.

11. Число a при циклическом сдвиге вправо на 1 разряд изменится следующим образом:

1) когда в крайнем правом разряде записан 0 — оно станет равно a div 2:

a	1	0	1	0	1	1	0	0
***a* shr 1**	0	1	0	1	0	1	1	0

2) когда в крайнем правом разряде записана 1 — оно станет равно a div 2 + 2^{k-1}, где k — количество двоичных разрядов в записи данного числа:

a	1	0	1	0	1	1	0	1
***a* shr 1**	1	1	0	1	0	1	1	0

12. Формула, по которой можно определить, как изменится число a при циклическом сдвиге вправо на b разрядов, в случае:

1) когда в b правых разрядах записаны 0:

$$a = \underbrace{a \text{ div } 2 \text{ div } 2 \ldots}_{b \text{ раз}} = a \text{ div } 2^b$$

На пример, при $b = 2$:

a	1	0	1	0	1	1	0	0
a* shr *b	0	0	1	0	1	0	1	1

2) когда в b правых разрядах записаны 1:

$$a = \underbrace{((a \text{ div } 2 + 2^{k-1}) \text{ div } 2 + 2^{k-1}) \text{ div } 2 + 2^{k-1} \ldots}_{b \text{ раз}}$$

Попробуем получить общую формулу, исследовав, например, вариант для $k = 5$, $2^{k-1} = 16$.

Для части выражения со знаками div общая формула a div 2^b. Такую же формулу можно получить и для "свободного члена" (16, 24, 28, …). Для этого составим табличку:

b				
1	16	31 – 15	31 – (16 – 1)	31 – (2^4 – 1)
2	24	31 – 7	31 – (8 – 1)	31 – (2^3 – 1)
3	28	31 – 3	31 – (4 – 1)	31 – (2^2 – 1)

из которой следует общая формула для "свободного члена": $2^k - 1 - (2^{k-b} - 1) = 2^k - 2^{k-b}$, а вся формула для расчета результата циклического сдвига будет иметь вид: a div $2^b + 2k - 2^{k-b}$ (проверьте ее для случая $k = 5$, $b = 4$).

13. Число a при циклическом сдвиге влево на 1 разряд изменится следующим образом:

1) когда в крайнем левом 8-м разряде записан 0 — число a увеличится в 2 раза:

a	0	1	1	0	1	0	1	1
***a* shl 1**	1	1	0	1	0	1	1	0

Обоснование

Число a без первой двоичной цифры равно остатку от деления a на 2^7 (то есть совпадает с a). После приписывания справа нуля это число увеличивается в 2 раза.

2) когда в крайнем левом 8-м разряде записана 1 — число a будет равно остатку от деления a на 2^7, умноженному на 2, и плюс 1:

1	1	1	0	1	0	1	1
1	1	0	1	0	1	1	1

Если указанный остаток равен 127 (например, $a = 255$), то число a не изменится, если он меньше 127, то число a уменьшится.

14. Число a при циклическом сдвиге влево на b разрядов изменится следующим образом:

1) когда в b левых разрядах записаны 0 — увеличится в 2^b раз;

2) когда в b левых разрядах записаны 1 — число a будет равно остатку от деления a на 2^{8-b}, умноженному на 2^b, и плюс $2^b - 1$.

15. Для числа n, записанного в восьми двоичных разрядах, можно получить цифры следующим образом:

1) цифру в первом (крайнем справа) разряде: n **and** 1;

2) цифру в четвертом разряде (n **shr** 3) **and** 1;

3) цифру в k-м разряде: (n **shr** (k – 1)) **and** 1.

Победителями конкурса (одного из самых сложных за время существования нашего издания) признаны Александр Андрющенко, Андрей Измайлов, Андрей Мартынов, Мария Касатонова и Николай Свистунов. Они будут награждены дипломами. Поздравляем!

b	***a***	***или a***
1	a div 2 + 16	a div 2 + 16
2	(a div 2 + 16) div 2 + 16	a div 2 div 2 + 24
3	((a div 2 + 16) div 2 + 16) div 2 + 16	a div 2 div 2 div 2 + 28

ПОДПИСКА на 2-е полугодие 2012 г.

журнал

Информатика – Первое сентября

ТАРИФНЫЕ ПЛАНЫ

Максимальный – 1254 руб.

бумажная версия (по почте) + CD (по почте) + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:

«Роспечать» – индекс 32291 *(для индивидуальных подписчиков и организаций)*

«Почта России» – индекс 79066 *(для индивидуальных подписчиков и организаций)*

Оптимальный – 594 руб.

электронная версия на CD (по почте) + доступ к электронной версии на сайте

Оформление подписки – на сайте www.1september.ru или на почте по каталогам:

«Роспечать» – индекс 19179 *(для индивидуальных подписчиков и организаций)*

«Почта России» – индекс 12684 *(для индивидуальных подписчиков и организаций)*

Экономичный – 200 руб.

доступ к электронной версии на сайте

Подписаться по данному тарифному плану можно только на сайте www.1september.ru

Бесплатный – 0 руб. ШКОЛА ЦИФРОВОГО ВЕКА

для педагогических работников образовательных учреждений, участвующих в Общероссийском проекте «Школа цифрового века».
Подробности – на digital.1september.ru

Бумажная версия (доставка по почте)

CD с электронной версией журнала и дополнительными материалами для практической работы (доставка по почте)

Электронная версия в Личном кабинете подписчика на сайте www.1september.ru. Дополнительные материалы включены

Пользователям электронной версии высылаются по почте подтверждающие документы

ЭКОНОМИЧНЫЙ тарифный план

ОПТИМАЛЬНЫЙ тарифный план

МАКСИМАЛЬНЫЙ тарифный план

При оформлении подписки на сайте www.1september.ru оплата производится по квитанции в отделении банка или электронными платежами on-line

VISA MasterCard QIWI