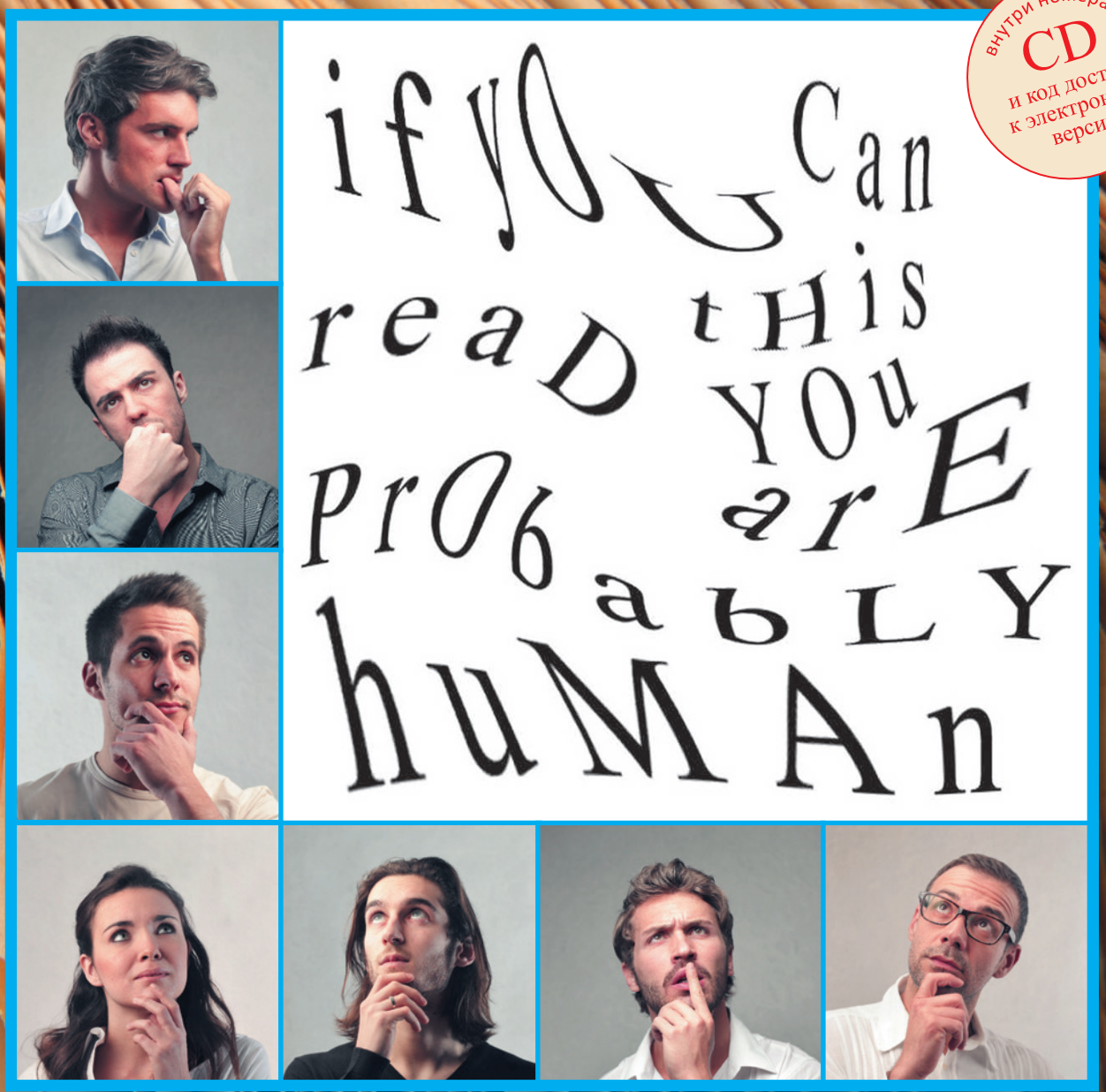
№12 Издание основано в 1995 г. inf.1september.ru
УЧЕБНО-МЕТОДИЧЕСКИЙ ЖУРНАЛ ДЛЯ УЧИТЕЛЕЙ ИНФОРМАТИКИ

ИНФОРМАТИКА



внутри номера CD и код доступа к электронной версии

издательский дом Первое сентября | декабрь 2013
1september.ru
ИНФОРМАТИКА Подписка по каталогу «Почта России»: 79066 — бумажная версия, 12684 — электронная версия

НА ОБЛОЖКЕ

Нет, наверное, никого, кто любил бы “капчи”. Согласитесь — жутко раздражает. Но и такое “зло для пользователей” можно применять с пользой. Так, что даже ярые ненавистники будут с гордостью говорить: используя “капчу”, я помогаю благому делу. На это благое дело работает проект reCAPTCHA. Суть идеи в том, что каждый раз пользователю приходится распознавать и вводить не одно, а два (!) слова. Ну, ничего себе благое дело, скажете вы! Но идея в том, что второе слово (его, кстати, вводить не обязательно) используется для помощи компьютерным системам распознавания текстов, которые работают над оцифровкой различных бумажных архивов. Людям предлагается помочь программам справиться с особо сложными случаями. Согласитесь — отличная идея!

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Презентации к статьям номера
- Исходные файлы, программы, раздаточные материалы

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ
по каталогу “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 23 138 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
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 – О.Волкова
История – А.Савельев
Классное руководство и воспитание школьников – М.Битянова
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
ОБЖ – А.Митрофанов
Русский язык – Л.Гончар
Спорт в школе – О.Леонтьева
Технология – А.Митрофанов
Управление школой – Е.Рачевский
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школа для родителей – Д.Тюттерин
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 13.11.2013,
фактически 13.11.2013
Заказ №
Отпечатано в ОАО “Первая Образцовая типография”
Филиал “Чеховский Печатный Двор”
ул. Полиграфистов, д. 1,
Московская область,
г. Чехов, 142300
Сайт: www.chpd.ru
E-mail: sales@chpk.ru
Факс: 8 (495) 988-63-76

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

Интерпретаторы

Ни слова о Бейсике ☺

Хочу рассказать одну историю из своей педагогической практики, которая до сих пор является для меня надежным средством испортить себе настроение. Вроде бы ничего страшного — мало ли, бывает. Но сейчас вот настроение опять испортилось.

Дело было несколько лет назад, мой педстаж к тому времени приближался ко второму десятку. Конец четверти (у нас в школе на самом деле тогда были триместры — не суть важно), "подбиваем" итоги, один из главных — набор обязательных задач, которые нужно было решить за четверть. Оценка за задачи сильно влияла на четвертную (что логично). Один из учеников принес оставшиеся до отличной оценки задачи на флешке. Но, как это нередко бывает, данные на флешке "побились". Ну, сказал я, знаешь — сам виноват. Не нужно было дотягивать до последнего урока. (Кстати, действительно не нужно было!) Но ученик сильно расстроился, мне тоже было жалко и его, и заслуженной (как я знал) пятерки. Решили так — он в тот же день пришлет мне задачи из дома. На том и договорились. Вспомнил о договоре я на следующее утро, когда выставлял оценки в журнал. Чуть разозлившись, я без тени сомнения влепил за задачи тройку (сколько-то все же было сдано ранее), в четверти — четверку. И забыл об этом.

Много позже этот ученик, который отлично закончил школу, выбрал программирование своей профессией, успешно работает и часто при случае тепло и искренне благодарит меня за науку, рассказал мне, что случилось в тот вечер. В тот вечер у него умерла бабушка. Они с родителями еще успели к ней приехать. Потом ее увезли в больницу, потом ее не стало. Было поздно, и они остались на ночь в квартире бабушки. (Потому и не мог он мне отправить задачи, что просто не доехал до дома.)

Это история грустная. Дальше — веселая. Грустную историю с оценкой я в очередной раз вспомнил, прочитав один из интернет-хитов последнего времени — историю про игуану. Ходит эта история в различных версиях, но смысл один. Вот как выглядит один из вариантов рассказа от первого лица:

"На день рождения подарили игуану. Обрадовался, играл с ней, кормил с рук. А потом она меня за палец цапнула. Я на нее обиделся. Потом смотрю: а она так на меня преданно и грустно смотрит, таскается за мной по всему дому, в глаза заглядывает: прости, мол, хозяин. С утра просыпаюсь, а игуана рядом сидит, смотрит грустно. Ну я аж расчувствовался. Но рука-то у меня опухла. Я — к доктору. Там и выяснилось, что этот вид игуан ядовит, только яд у них очень слабый, поэтому они сначала кусают, а потом таскаются за жертвой, ждут, пока подохнет".

Если бы можно было у игуаны спросить прямо — чего, дескать, таскаешься за мной, жалеешь?

Вот, настроение поднялось ☺

Сергей Островский,
главный редактор (*so@1september.ru*)

Веб-дизайн: практикум

К.Ю. Поляков,
д. т. н., Санкт-Петербург

Введение

В наше время веб-дизайн — модное и востребованное направление. Можно с достаточно высокой вероятностью предполагать, что многие выпускники школы, которые выберут профессии в области информационных технологий, будут работать в этой области или по крайней мере непосредственно контактировать с теми, кто там работает. Поэтому чрезвычайно важно показать школьникам основные идеи и направления современного подхода к разработке веб-страниц.

Конечно, многие профессиональные сайты с динамическим содержанием строятся на основе готовых систем управления содержанием (англ. *CMS* = = ***C****ontent* ***m****anagement* ***s****ystem*), на которые “натягивается” шаблон (стиль оформления). Однако веб-разработчику любого уровня не только полезно, но и необходимо знать, “что там внутри”, как строится веб-страница, которую генерируют скрипты. Только такой подход позволяет решать реальные задачи, выходящие за пределы методики “щелканья по кнопкам”.

Поскольку изучение серверных языков типа PHP и устройства CMS никак не укладывается в программу (даже углубленного!) школьного курса информатики, остается использовать отпущенные учебные часы для изучения языка HTML и создания статических веб-страниц. В то же время знакомиться только с HTML, оставаясь на уровне, который был еще как-то приемлем 10–15 лет назад, сейчас нельзя. Не говорить о CSS и Javascript уже невозможно, хотя некоторые авторы учебников умудряются это делать. Среди доступной литературы для школьников хочется отметить замечательную серию книг А.А. Дуванова [1–2], которая была издана уже довольно давно, но не потеряла актуальности.

Достаточно серьезная проблема — поиск практических работ по созданию веб-сайтов, которые можно использовать на уроках в школе. В свое время автору не удалось найти готовые работы по этой теме, отвечающие современным требованиям, поэтому он вынужден был разработать свои. Эти работы вошли в практикум к учебникам информатики углуб-

ленного уровня [3–4], который вышел в 2013 году в издательстве “Бином”. Практикум можно бесплатно загрузить с сайта поддержки учебника *kpolyakov.spb.ru/school/probook.htm*. В настоящей статье представлено описание этих работ и краткий теоретический материал, необходимый для их выполнения. Для более подробного ознакомления с вопросом читатели могут использовать учебники [3–4] и разнообразные материалы по веб-дизайну в Сети.

Все работы можно выполнять в любой операционной системе, для этого нужны только текстовый редактор (желательно с подсветкой синтаксиса HTML) и браузер. В качестве одного из возможных решений для пользователей *Windows* можно рекомендовать бесплатный редактор *HEFS* (*kpolyakov.spb.ru/prog/hefs.htm*), который позволяет сразу просматривать результат работы браузера *Internet Explorer* в окне программы.

Текстовые веб-страницы

Веб-страницы — это текстовые файлы (в формате “только текст”), в которых содержатся ссылки на дополнительные файлы (файлы описания стилей, скрипты, рисунки, звуки, видео). Поэтому для работы с ними достаточно простого текстового редактора типа *Блокнота* в *Windows* или *Vim* (*www.vim.org*).

Учащиеся должны понимать, что HTML — это не язык программирования, а язык разметки. Для этой цели используются специальные команды — *тэги*, заключенные в угловые скобки. Тэги бывают парные и непарные. Область действия парных тэгов (*контейнеров*) определяется открывающим и закрывающим тэгами; закрывающий тэг начинается с символа “/” (“слэш”).

Вся страница заключается в контейнер **html**, внутри которого размещены контейнеры **head** (“голова”, служебная информация) и **body** (“тело”, основная часть страницы). Из тэгов, размещаемых в контейнере **head**, на первых порах наибольший интерес представляет контейнер **title** — название страницы, который показывается в заголовке окна браузера и служит важнейшей информацией для поисковых систем.

В основной части страницы выделяют заголовки (контейнеры **h1**, **h2**, **h3**, **h4**, **h5** и **h6**, от англ. *header* — заголовок) и абзацы текста (контейнеры **p**, от англ. *paragraph*). Для этих элементов можно установить режим выравнивания с помощью атрибута (свойства) **align**, например, так:

```
<h1 align='center'>...</h1>
```

Тэг **
** служит для перехода на новую строку в пределах одного абзаца.

Нужно обратить внимание на специальные символы, которых нет на клавиатуре. Например, для того чтобы вставить знаки параграфа (§) и авторского права (©), используются последовательности **§** и **©**. По традиции в русских текстах применяются французские кавычки (“елочки”), которые кодируются как **«** и **»**. Важную роль играет неразрывный пробел (** **), с помощью которого отделяются инициалы человека от его фамилии (А.Пушкин), числа от единиц измерения (25 кг) и тире (**—**) от предыдущего слова (для того чтобы новая строка не начиналась с тире).

Практическая работа № 1

1. Скопируйте на свой компьютер файл[1] **dogs.htm**. Посмотрите, как будет выглядеть этот документ в браузере.

2. Добавьте тэги, необходимые для правильного HTML-документа. В заголовке страницы напишите название документа и фамилию автора, например:

Собаки: работа Василия Пупкина

3. Выделите с помощью тэгов **h1** и **h2** заголовок документа и заголовки разделов. Перед номером раздела добавьте знак “§” (после этого знака ставится неразрывный пробел).

4. Оформите стихотворение, используя команду перехода на новую строку **br**. Перед фамилией автора поставьте знак авторского права “©”.

5. Выделите абзацы текста с помощью тэга **p**, установите выравнивание по ширине.

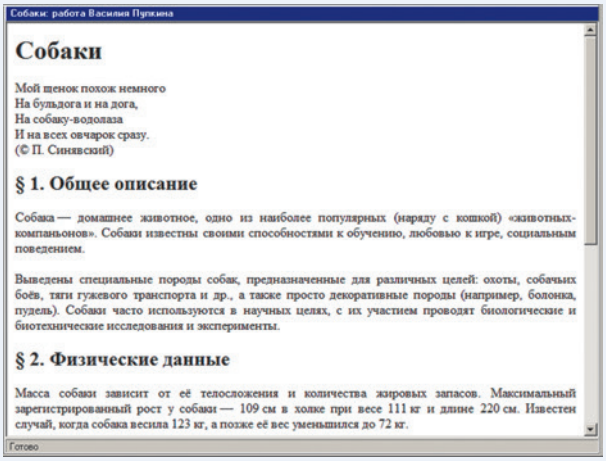
6. Замените знаки “минус” на тире, между тире и предыдущим словом поставьте неразрывный пробел.

7. Замените верхние кавычки на “елочки”.

8. Поставьте неразрывные пробелы между числами и единицами измерения, например, **25 кг**.

9. В начале последнего абзаца поставьте знак авторского права:

© Википедия, 2011



Списки

Списки — важный элемент логической разметки страницы. Маркированный список заключается в контейнер **ul** (от англ. ***u**nordered **l**ist* — неупорядо-

[1] Все файлы, используемые в практических работах, находятся на диске в приложении к этому номеру, а также на сайте поддержки учебника (*kpolyakov.spb.ru/school/probook/prakt.htm*).

ченный список), а нумерованный (с автоматической поддержкой нумерации) — в контейнер **ol** (от англ. *ordered list* — упорядоченный список). Атрибуты для маркированного списка позволяют задать вид маркера:

```
<ul type='circle'>...</ul>
```

а для нумерованного — стиль нумерации и начальное значение номера:

```
<ol type='I' start='5'>...</ol>
```

Каждый элемент списка заключается в контейнер **li**.

Практическая работа № 2

1. Скопируйте на свой компьютер файл **lists.htm**. Посмотрите, как выглядит эта веб-страница в браузере.

2. Оформите веб-страницу про устройство компьютера с помощью маркированных и нумерованных списков:

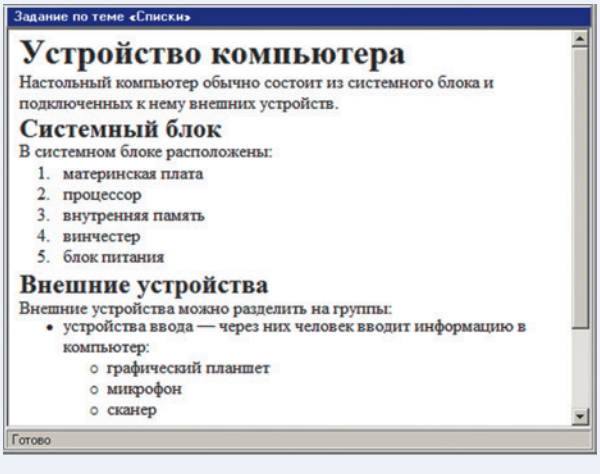
а) Добавьте заголовок документа “**Устройство компьютера**”.

б) Добавьте заголовки разделов “**Системный блок**” и “**Внешние устройства**”.

в) Оформите состав системного блока как нумерованный список.

г) Оформите перечень внешних устройств как многоуровневый маркированный список (первый уровень — группа устройств, второй — перечень устройств этой группы).

д) Исправьте фактические ошибки, которые допущены при наборе.



Гиперссылки

Для добавления гиперссылок на веб-страницу используют тэг **a** (англ. *anchor* — якорь) с атрибутом **href**, который указывает на целевой документ, открывающийся после щелчка на гиперссылке. Рассказывая о гиперссылках, нужно подчеркнуть, что на веб-страницах допускается использование нескольких типов гиперссылок:

• относительные гиперссылки на страницы и документы этого же сайта:

```
<a href='sections.htm'>
Список разделов</a>
<a href='book/sections.htm'>
Список разделов</a>
<a href='../sections.htm'>
Список разделов</a>
<a href='/sections.htm'>
Список разделов</a>
```

• внешние ссылки (на ресурсы других сайтов):

```
<a href='http://lenta.ru/rubrics/culture'>
Культура</a>
```

• ссылки, приводящие к запуску почтовой программы (**mailto**) :

```
<a href='mailto:inf@1september.ru'>
Информатика </a>
```

Никаких абсолютных ссылок типа “**C:\SITE\mypage.htm**” быть не должно, потому что

• при перемещении такого сайта абсолютные ссылки становятся недействительными;

• в Unix-подобных операционных системах в качестве символа-разделителя можно использовать только прямой слэш “/”.

В больших документах полезно использовать внутренние ссылки на странице

```
<a href='#chap5'>Глава 5</a>
```

которые позволяют быстро перейти на метки, отмеченные в HTML-документе с помощью тэга **a** с атрибутом **name**:

```
<a name='chap5'></a><h2>Глава 5</h2>
```

В практической работе используются только ссылки на внешние ресурсы и ссылка, запускающая почтовую программу. Предполагается, что учащиеся способны найти в Интернете нужную информацию с помощью поисковых систем *Google* и *Яндекс*.

Практическая работа № 3

1. Скопируйте на свой компьютер файл **anchor.htm**. Посмотрите, как выглядит файл в браузере.

2. Добавьте название страницы (**title**), укажите там название документа и фамилию автора, например:

Песни из мультфильмов: работа Василия Пупкина

3. Найдите и выделите нужным стилем заголовок документа.

4. Замените верхние кавычки на “елочки”.

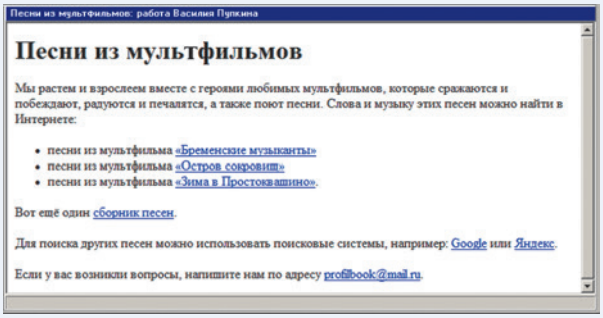
5. Найдите в Интернете тексты указанных песен и сделайте ссылки на эти страницы в элементах списка.

6. Сделайте ссылку на указанный в тексте архив так, чтобы его можно было скачать по этой ссылке. Текст должен быть примерно такой:

Вот еще один сборник песен.

7. Сделайте названия поисковых систем *Google* и *Яндекс* ссылками. При щелчке по этим ссылкам должны сразу появляться результаты поиска по запросу “песни из мультфильмов”.

8. Сделайте активной ссылку на адрес электронной почты в конце документа (при щелчке по ней должна запускаться почтовая программа).



Оформление текста

В идеале текст HTML-страницы должен содержать только *логическую* разметку документа и никаких данных о *физической* разметке, то есть о том, как именно будет представлена информация на устройстве вывода. В предыдущих работах уже использовались приемы логической разметки — выделялись заголовки, абзацы, элементы списков, гиперссылки.

К элементам логической разметки можно отнести также тэги **em** (от англ. *emphasize* — выделение), **strong** (сильное выделение), **cite** (цитата), **dfn** (от англ. *definition* — определение), **abbr** (от англ. *abbreviation* — сокращение). Программный код внутри абзаца выделяется с помощью контейнера **code**, а листинги программ заключаются в контейнер **pre** (от англ. *preformatted* — заранее отформатированный).

Для добавления верхних и нижних индексов, например в математических или химических формулах, применяют тэги **sup** (от англ. *superscript* — верхний индекс) и **sub** (от англ. *subscript* — нижний индекс).

В этой практической работе внешний вид элементов (гарнитура и размер шрифта, цвет символов и фона и т.п.) никак не уточняется — это будет темой следующей работы.

Практическая работа № 4

1. Скопируйте на свой компьютер файл **text.htm**. Посмотрите, как выглядит страница в браузере.

2. Добавьте название страницы (**title**), укажите там название документа и фамилию автора, например:

Примеры оформления текста: работа Василия Пупкина

3. Найдите и выделите соответствующими тэгами заголовок документа и заголовки разделов.

4. Разбейте текст на абзацы.

5. Замените верхние кавычки на “елочки”.

6. Замените везде знаки “минус” на тире, сделайте так, чтобы тире не отрывались от предыдущих слов.

7. Выделите слова “сервер” и “клиент” в первом абзаце с помощью тэга **em**.

8. В следующих двух абзацах выделите определяемые слова (“сервер” и “клиент”) с помощью тэга **dfn**, а все английские слова — с помощью тэга **em**.

9. Оформите математические и химические формулы, используя верхние и нижние индексы. Формула должна занимать отдельную строку и быть выровнена по центру. Точку или запятую, которая следует за формулой, нельзя отрывать от формулы. Исправьте фактические ошибки в формулах, допущенные при наборе.

10. Выделите имена переменных в тексте с помощью тэга **em**.

11. Оформите программу на языке Паскаль так, чтобы сохранилось все форматирование.

Использование CSS

Задача следующих двух практических работ — научиться использовать каскадные таблицы стилей для оформления документа. Согласно принципу разделения содержания и оформления, все установки, задающие внешний вид документа, должны быть вынесены в отдельный файл, который, как правило, имеет расширение **css**. Чтобы подключить стилевой файл **test.css** к веб-странице, используют непарный тэг **link** (связь), который помещают в контейнер **head**:

```
<link rel='stylesheet' type='text/css'
 href='test.css'>
```

В стилевом файле записывают *селекторы* (обозначения элементов, к которым применяется оформление) и после них в фигурных скобках — пары “свойство – значение”, например:

```
p {
  color: green;
  background: white;
}
```

Для того чтобы применить оформление к некоторым (а не ко всем!) однотипным элементам, вводятся *классы*:

```
<p class='error'>...</p>
```

Специальное стилевое оформление для них задается с помощью селектора **p.error**:

```
p.error {
   color:red;
}
```

Если не указано название тэга, селектор выбирает все элементы, имеющие данный класс:

```
.error {
   color:red;
}
```

Для того чтобы применить одинаковое оформление к нескольким типам элементов, их перечисляют через запятую:

```
h1, h2, h3 {
  font-family: Arial, Helvetica,
               sans-serif;
}
```

Если обозначения разделены пробелом, оформление применяется к вложенному элементу, например, к элементу **em** внутри абзаца **p**:

```
p em {
  font-style: italic;
}
```

В двух практических работах по CSS используются следующие свойства, с которыми нужно предварительно познакомить учащихся (можно также стимулировать их самостоятельную работу ссылкой на сайт *css.manual.ru*):

color
background
font-family
font-weight
font-style
text-align
border
margin
padding

Практическая работа № 5

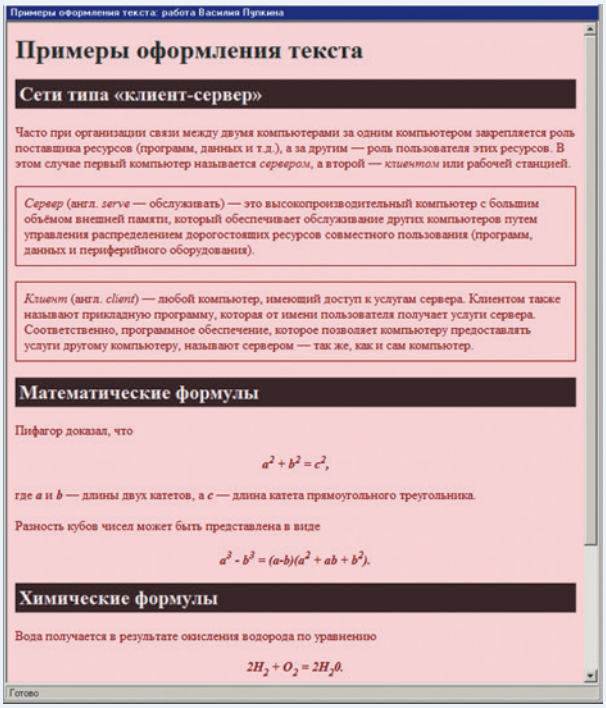
1. Откройте файл **text.htm**, полученный в результате выполнения работы № 4.

2. Создайте в той же папке стилевой файл **mystyle.css**, подключите его к веб-странице и определите стили оформления, перечисленные в следующих пунктах. Для получения дополнительной справочной информации по CSS используйте Интернет, например, сайт *css.manual.ru*.

3. Задайте свои цвета фона и текста. Текст должен хорошо читаться на выбранном фоне.

4. Определите цвет заголовков (**h1**, **h2**), отличающийся от цвета основного текста. Заголовки **h2** должны также выделяться фоном, причем фон должен быть темный, а буквы — светлые. Добавьте внутренний отступ (**padding**) в пять пикселей.

5. Определите класс **formula**, применимый как к абзацам (**p**), так и к отдельным словам (**span**), и задайте для него *выравнивание по центру; жирный шрифт; курсив*. Используя этот класс, выделите абзацы с формулами и имена переменных в тексте.



6. Определите класс **definition**, применимый только к абзацам (**p**), и задайте для него: *рамку толщиной в один пиксель; внутренний отступ в десять пикселей*.

Используя этот класс, выделите два абзаца с определениями терминов.

Практическая работа № 6

1. Скопируйте на свой компьютер каталог **CSS**. Откройте файл **cats.htm**. Все последующие задания нужно выполнять с помощью CSS.

2. Установите для тела страницы бледно-розовый фон (цвет #FFEEEE) и шрифт *Georgia*, а если его нет — любой шрифт с засечками (*serif*).

3. Для заголовков **h1** и **h2** определите шрифт *Arial*, если его нет — *Helvetica*, а если нет их обоих, то любой рубленый шрифт (*sans-serif*).

4. Для заголовка **h1** задайте фоновый рисунок **cat.png** из подкаталога **images** (без повторения).

5. Для заголовков **h2** определите темно-красный цвет (#800000).

6. Создайте класс оформления **latin** для выделения слов и выражений на латинском языке: курсив, темно-красный цвет. Выделите в тексте все латинские слова с помощью этого стиля.

7. Установите для всех абзацев отступ (красную строку) 30 пикселей.

8. Создайте новый стиль оформления абзацев — **author**: курсивный шрифт, без абзацного отступа. Выделите этим стилем имена и фамилии авторов цитат.

9. Создайте стиль оформления абзацев с цитатами с именем **quote**:

а) фон — светло-желтый (#FEFEE2);

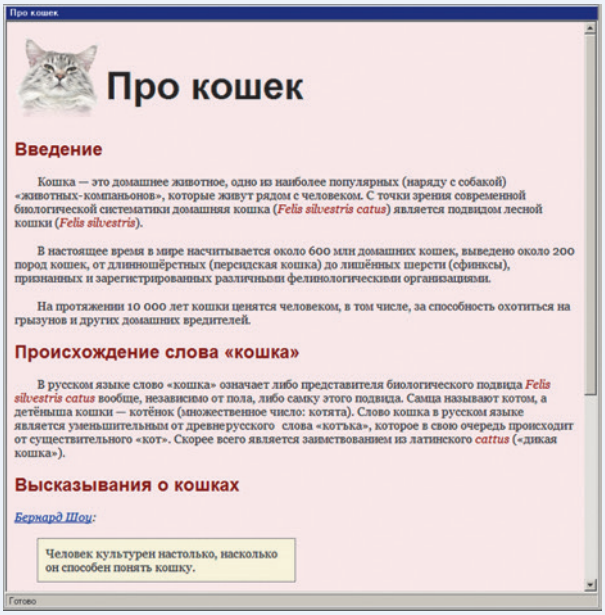
б) абзацного отступа нет;

в) ширина 50% от ширины окна браузера;

г) внешние отступы: сверху — 0, справа и слева по 30 пикселей, снизу — 10 пикселей;

д) рамка шириной один пиксель, точечная (*dotted*), черного цвета;

е) внутренние отступы 10 пикселей.



10. Найдите в Интернете информацию про авторов цитат и сделайте их имена ссылками на соответствующие страницы.

11. Установите для ссылок, которые находятся внутри абзаца стиля **author**, красный цвет при наведении мыши.

Вставка рисунков в документ

Для добавления рисунков на веб-страницу можно использовать непарный тэг **img** с единственным обязательным атрибутом **src** (от англ. *Source* — источник):

```
<img src='/images/vasya.jpg'
 width='100' height='150'>
```

На уроке нужно объяснить, почему считается хорошим стилем указывать размеры рисунка, альтернативный текст (атрибут **alt**) и отступы.

Заранее известные размеры рисунка позволяют браузеру сверстать страницу так, что она не “поедет”, когда рисунок будет фактически загружен с сервера.

Атрибут **alt** дает возможность получить какую-то информацию об изображении, когда оно не загружено (например, если в браузере отключен вывод рисунков). Кроме того, он полезен для незрячих и слабовидящих людей, для которых веб-страницы читают специальные программы (*скринридеры*).

Практическая работа № 7

1. Скопируйте на свой компьютер каталог **img**. Откройте файл **img.htm** в редакторе.

Ваша задача — оформить веб-страницу про некоторые направления в музыке XX века. Для оформления нужно использовать рисунки из подкаталога **images**.

2. Добавьте заголовок страницы (**title**), укажите там название документа и фамилию автора, например:

Веб-страница с рисунками: работа Василия Пупкина

3. Найдите и выделите соответствующими тэгами заголовок документа и заголовки разделов.

4. Разбейте текст на абзацы.

5. Замените верхние кавычки на “елочки”.

6. Замените везде знаки “минус” на тире, сделайте так, чтобы тире не отрывались от предыдущих слов.

7. Создайте стилевой файл **mystyle.css** и подключите его к веб-странице. С помощью стилевого файла:

а) установите в качестве фона страницы рисунок **back.jpg**;

б) выберите соответствующий цвет текста;

в) создайте класс **name**, определите для него жирный шрифт, курсив и свой цвет, и выделите имена и фамилии, названия ансамблей, фильмов, песен (тэг **span**).

8. Добавьте на страницу рисунки, расположенные в подкаталоге **images**.

9. Установите для рисунков выравнивание (с обтеканием текстом) и всплывающие подсказки. Рисунки лучше ставить в шахматном порядке (один налево, следующий — направо и т.д.).

10. С помощью стилевого файла установите для рисунков внешний отступ (**margin**) 10 пикселей.

Вставка звука и видео в документ

В этой работе учащимся предстоит добавить на веб-страницу звуковые файлы в разных форматах (WAV, MP3, OGG), флэш-ролики и видео. К сожалению, нужно быть готовым к тому, что браузер пользователя поддерживает не все запрошенные возможности, и некоторых звуков мы не услышим, а некоторых видеороликов — не увидим.

Для вставки мультимедийных объектов используется тэг **embed**, а также новые тэги **audio** и **video**, введенные в HTML5.

Практическая работа № 8

1. Скопируйте на свой компьютер каталог **MEDIA**. Откройте файл **media.htm** в редакторе. Ваша задача — научиться добавлять на веб-страницу звуковые файлы, флэш-анимацию и видеоролики. Для этого нужно использовать мультимедийные данные из подкаталогов **audio**, **flash** и **video**. Места вставки медиафайлов вы легко определите из текста файла **media.htm**.

2. Добавьте на веб-страницу звуковой файл **hcopter.wav** из каталога **audio** (шум вертолета). Используйте тэг **embed**, отключите автозапуск проигрывания, установите ширину проигрывателя 300 пикселей и высоту 20 пикселей.

3. Добавьте на веб-страницу звуковой файл **nature.mp3** из каталога **audio** (звуки природы). Используйте тэг **embed**, укажите те же атрибуты, что и в предыдущем случае.

4. Добавьте на веб-страницу звуковой файл **nature.ogg** из каталога **audio** (звуки природы). Используйте тэг **audio**, укажите его атрибуты:
src='адрес файла' — ссылка на файл-источник
controls='controls' — показывать элементы управления
width='300' — ширина проигрывателя в пикселях
height='20' — высота проигрывателя в пикселях

Откройте веб-страницу в браузере и выясните, поддерживает ли ваш браузер тэг **audio**.

5. С помощью тэга **embed** добавьте на страницу флэш-ролик **cube.swf** (вращающийся кубик) из каталога **flash**. Установите размеры 257 на 200 пикселей.

6. С помощью тэга **embed** добавьте на страницу флэш-ролик **lightbot.swf** (игра *Lightbot*) из каталога **flash**. Установите размеры 640 на 480 пикселей. Откройте веб-страницу в браузере и посмотрите, как работает флэш-анимация.

7. Зайдите на сервис **youtube.com**, с помощью поиска по сайту найдите ролик по теме "HTML5" и вставьте на веб-страницу. Для этого нужно щелкнуть по кнопке *Поделиться* под роликом, затем — по строчке *HTML-код*. После этого остается просто скопировать код, появившийся в текстовом окне, на веб-страницу.

8. С помощью тэга **video** добавьте на страницу видеофайл **finish.mp4** из каталога **video** следующим образом:

```
<video width='ширина' height='высота'
 controls='controls'>
<source src='адрес файла'
        type='video/mp4;
        codecs='avc1.42E01E,
        mp4a.40.2''>
</video>
```

Установите размеры окна 320 на 240 пикселей.

9. С помощью тэга **video** добавьте видеофайл **finish.webm** из каталога **video** следующим образом:

```
<video width='ширина' height='высота'
 controls='controls'>
<source src='адрес файла'
        type='video/webm;
        codecs='vp8, vorbis''>
</video>
```

Установите размеры окна 320 на 240 пикселей.

10. Обновите страницу в браузере и выясните, поддерживает ли ваш браузер тэг **video**, и с какими форматами он умеет работать.

Табличная верстка

Следующая тема курса — таблицы, которые строятся с помощью тэгов **table** (таблица), **tr** (англ. *table row* — строка таблицы), **th** (англ. *table header* — заголовок таблицы) и **td** (англ. *table data* — данные таблицы).

Хотя табличная верстка считается "нехорошим" стилем среди веб-разработчиков, во многих случаях именно такой подход позволяет надежно обеспечить кросс-браузерность, то есть одинаковый вид страницы в разных браузерах.

В этой работе таблицы используются как по прямому назначению (для оформления табличных данных), так и для верстки — расположения материала в задуманном виде. Оформление таблиц выполняется через стилевой файл.

Практическая работа № 9

1. Скопируйте на свой компьютер каталог **TABLE.** Откройте файл **table.htm** в редакторе. Ваша задача — оформить веб-страницу так, как показано на образце. Для этого нужно использовать рисунки из подкаталога **images** и стилевой файл.

2. Добавьте заголовок страницы (**title**), укажите название документа и фамилию автора, например:
Таблицы: работа Василия Пупкина.

3. С помощью стилевого файла установите для всех таблиц вертикальный отступ (**margin**) 10 пикселей.

4. Присвойте первой таблице класс **sport** для того, чтобы можно было управлять оформлением ее ячеек независимо от других таблиц.

5. Введите класс **left** для ячеек первой таблицы (селектор **table.sport td.left**), установите для них выравнивание по левой границе. Для остальных ячеек таблицы (селектор **table.sport td**) установите (с помощью стилевого файла) выравнивание по центру. Ячейки, содержащие заголовки и места, занятые командами, оформите с помощью тэга **th**.

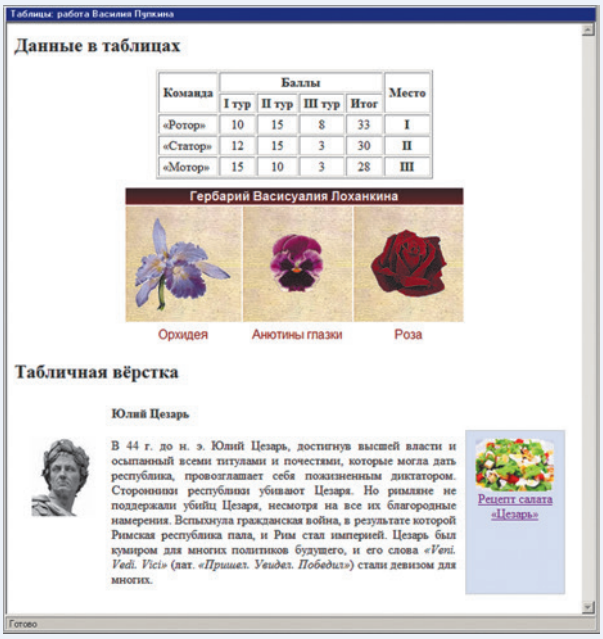
6. Для всех ячеек первой таблицы установите внутренний отступ два пикселя по вертикали и пять пикселей по горизонтали.

7. Для второй таблицы используйте три класса ячеек:
header — фоновый рисунок **back.jpg** из каталога **images**; белый цвет символов, шрифт без засечек (**sans-serif**), жирный, выравнивание по центру;
picture — выравнивание по центру, фоновый рисунок **paper.jpg** из каталога **images**; внутренний отступ (**padding**) 10 пикселей, ширина ячейки (**width**) 120 пикселей;
note — шрифт "Arial", "Helvetica" или любой шрифт без засечек (**sans-serif**), цвет символов #800000, выравнивание по центру, внутренний отступ (**padding**) 5 пикселей сверху и снизу, и 10 пикселей по бокам, шрифт без засечек.

8. Замените верхние кавычки на "елочки".

9. Выделите латинские слова и их перевод с помощью тэга **em**.

10. Рекламный блок в третьей таблице расположен в отдельной ячейке, которой присвоен класс **ad**: рамка цвета #CCCCCC толщиной один пиксель, фоновый цвет **rgb(230,230,255)**, выравнивание по центру. Текст должен быть ссылкой на веб-страницу в Интернете с рецептом салата "Цезарь".



Блочная верстка

Блочная верстка на современных сайтах практически вытеснила табличную. Она позволяет сделать более логичным содержание HTML-файла, очистив его от тэгов физического оформления, роль которых при табличной верстке выполняют тэги: **table**, **tr**, **th** и **td**.

Блоки — это элементы **div**, которые не имеют определенного смыслового содержания, то есть их можно наполнять чем угодно. Для их оформления используются стандартные свойства: **border**, **margin**, **padding**. Но более интересно свойство **float** (обтекание другими элементами), с помощью которого можно выполнять достаточно сложную верстку материала, не используя таблицы. Установка свойства **float:left** задает обтекание, при котором элемент прижимается к левой границе, а при **float:right** — к правой.

Для блоков часто используют идентификаторы (атрибут **id**):

```
<div id='menu'>...</div>
```

Идентификатор — это уникальная характеристика, в том смысле, что на странице не должно быть двух элементов с одинаковым идентификатором. Чтобы указать на блок с идентификатором, в стилевом файле перед его именем ставится знак “**#**”:

```
#menu {
  font-style: italic;
  float: left;
}
```

В этой работе учащиеся освоят трехколоночную верстку без использования таблиц.

Практическая работа № 10

1. Скопируйте на свой компьютер каталог **DIV**. Откройте файл **baikal.htm** в редакторе. Ваша задача — оформить веб-страницу так, как показано на образце. Для этого нужно использовать рисунки из подкаталога **images** и стилевой файл.

2. Добавьте заголовок страницы (**title**), укажите название документа и фамилию автора, например:

Блочная верстка: работа Василия Пупкина.

3. Создайте стилевой файл **baikal.css** и подключите его к веб-странице. Установите для всех элементов страницы (они обозначаются знаком “*****”) нулевые внутренние и внешние отступы:

```
* {
  margin:0px;
  padding:0px;
}
```

4. Добавьте в блок с идентификатором **header** заголовок документа “Озеро Байкал”. В стилевом файле установите для этого блока высоту 80 пикселей и фоновый рисунок **header.jpg** из каталога **images** (без повторения). Добавьте для заголовка отступы слева и сверху (как на образце).

5. Добавьте в блок **pogoda** два скрипта, которые записаны в файле **informer.txt**. Они выводят на страницу информацию о погоде в двух поселках на берегу Байкала — в Листвянке и Хужире.

6. Добавьте в блок **photo** четыре фотографии с именами **baikal1.jpg**, **baikal2.jpg**, **baikal3.jpg**, **baikal4.jpg** из подкаталога **images**.

7. Добавьте в блок **content** текст, записанный в файле **text.txt**. Оформите абзацы с помощью тэга **p**. В стилевом файле для тэгов **p** установите отступы со всех сторон 5 пикселей и абзацный отступ (красную строку) 20 пикселей.

8. В тексте замените, где нужно, знаки “минус” на тире, перед тире поставьте неразрывные пробелы. Поставьте неразрывные пробелы между числами и единицами измерения.

Добавьте в конец текста абзац со словами

По материалам Википедии.

Выровняйте абзац по правой границе и выделите этот текст с помощью тэга **em**.

Слово “Википедии” должно быть ссылкой на страницу Википедии, посвященную Байкалу.

9. Добавьте в блок **footer** текст, содержащий вашу фамилию, имя и класс, например,

Работу выполнил Василий Пупкин, 11-й “А” класс

10. Посмотрите, как выглядит страница в браузере. Теперь остается с помощью стилей оформить блоки так, чтобы они заняли правильное место на странице.

11. В стилевом файле для блока **pogoda** установите ширину 150 пикселей, отступ слева 10 пикселей и обтекание слева (**float:left;**). Посмотрите, как теперь выглядит страница.

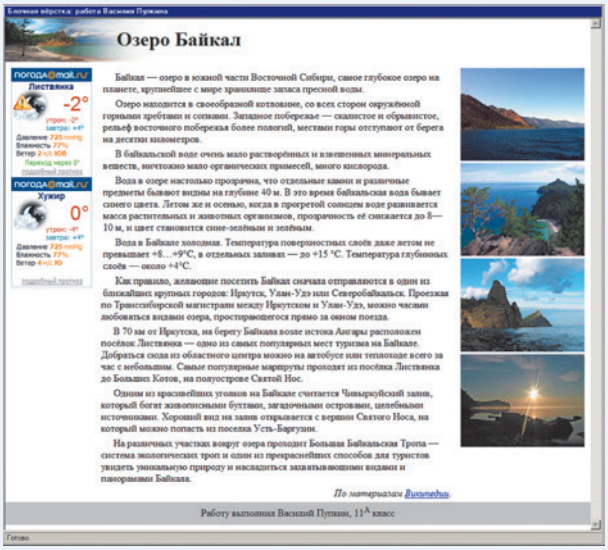
12. В стилевом файле для блока **photo** установите ширину 220 пикселей, отступ слева 10 пикселей и обтекание справа (**float:right;**). Посмотрите, как теперь выглядит страница.

13. Теперь сделаем так, чтобы центральная колонка (блок **content**) не залезала на левую и правую. В стилевом файле для блока **content** установите отступ слева 150 пикселей и отступ справа 220 пикселей. Посмотрите, как теперь выглядит страница.

14. Остается оформить нижний блок (“подвал”). В стилевом файле установите для блока **footer** цвет фона **#CCCCCC**, внутренние отступы — сверху 5 пикселей, снизу — 10 пикселей, выравнивание текста по центру. Для того чтобы этот блок не обтекался другими ни слева, ни справа, добавьте свойство

```
clear:both;
```

15. Посмотрите на окончательный результат в браузере:



Использование Javascript

Современные веб-страницы редко обходятся без использования языка программирования *Javascript* для автоматизации каких-то операций или придания странице динамичности и интерактивности. Понятно, что изучить *Javascript* за короткое время не удастся, задача состоит в том, чтобы дать почувствовать его достоинства при решении часто встречающихся задач и направить заинтересованных учащихся в нужном направлении. Среди таких задач выбраны три:

• эффект изменения рисунка при наведении мыши (“ролл-овер”), который реализуется с помощью обработчиков событий **onMouseOver** и **onMouseOut**:

```
<img src='box_closed.gif'
     onMouseOver='this.src='box_opened.gif''
     onMouseOut='this.src='box_closed.gif''>
```

• появление на экране скрытого блока, к которому сначала применяется стиль **display:none**, а после щелчка мышкой на каком-то элементе (например, на слове “Показать”) вызывается функция **show**, которая меняет стиль элемента, делая его видимым:

```
function show ( name )
{
 var elem=document.getElementById(name);
  if ( elem )
    elem.style.display = 'block';
}
```

Эта функция принимает единственный аргумент — идентификатор блока, ссылка на который ищется в дереве DOM (англ. ***D**ocument **O**bject **M**odel* — объектная модель документа) с помощью метода **getElementById**. Если элемент найден, выполняется изменение стиля.

• обработка данных простейшей формы:

```
<form name='calc'>
  <input name='answer'>
  <input type='button' value='Ответить'
onClick='check();'>
</form>
```

После щелчка по кнопке “Ответить” срабатывает обработчик события **onClick**, в котором вызывается функция **check**, проверяющая введенный ответ:

```
function check()
{
if ( calc.answer.value == 'Правильный
ответ' )
      alert('Правильно!');
else alert('Неправильно!');
}
```

Веб-страница, которую учащиеся создают в результате этой работы, использует три файла: HTML-страницу, стилевой файл и файл, содержащий скрипты.

Практическая работа № 11

1. Скопируйте на свой компьютер каталог **SCRIPT.** Откройте в редакторе файл **valaam.htm** и посмотрите, как он выглядит в браузере.

Ваша задача — оформить документ так, как показано на образце. В нем будет один скрытый блок. Для оформления используйте рисунки из каталога **images**.

2. Добавьте тэги, необходимые для правильного HTML-документа. Добавьте заголовок страницы (**title**), укажите название документа и фамилию автора, например:

Валаам: работа Василия Пупкина.

3. Выделите заголовок документа стилем **h1**. Справа от заголовка добавьте рисунок **valaam_gray.gif**, который будет ссылкой на сайт **valaam.ru**. При наведении мыши рисунок должен меняться на **valaam.gif**.

4. Выделите абзацы в тексте с помощью тэга **p**.

5. Создайте стилевой файл **valaam.css** и файл для скриптов **valaam.js**, подключите эти файлы к документу. Все оформление должно быть сделано с помощью CSS, все скрипты “убраны” в файл **valaam.js**.

6. Замените верхние кавычки на “елочки”.

7. Замените везде, где нужно, обычные пробелы на неразрывные и знаки “минус” — на тире; сделайте так, чтобы тире не отрывались от предыдущих слов.

8. Сделайте плавающий блок, содержащий фотографию Валаама и подпись “Фото А.Колыбалова (*www.rg.ru*)”. Адрес сайта должен быть ссылкой на этот сайт.

Подпись должна быть набрана шрифтом без засечек (**sans-serif**), курсивом, размер 80% от размера шрифта основного текста, внешних отступов (**margin**) нет.

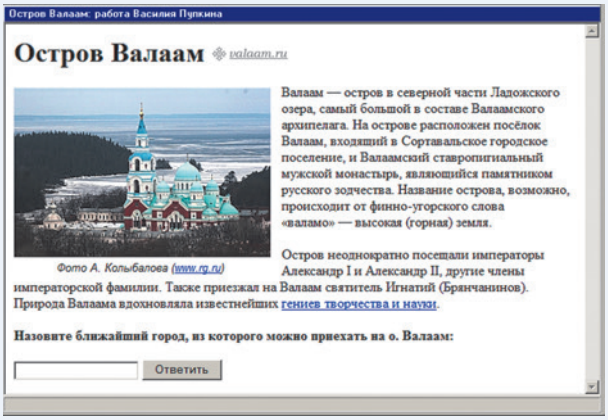
9. Оформите скрытый блок, включающий ту часть текста, которой нет на экране в краткой версии (см. образец далее). Присвойте этому блоку идентификатор (**id**), установите следующее оформление: цвет фона #EEEEFF; рамка синяя, сплошная, толщиной один пиксель; отступы по бокам 10 пикселей.

10. В нижнюю часть скрытого блока добавьте внутренний плавающий блок, содержащий фотографию резьбы по дереву с подписью *'К.Гоголев. "На пристани" (резьба по дереву)'*. Оформление подписи должно быть такое же, как и для первого плавающего блока.

11. Сделайте так, чтобы скрытый блок показывался при щелчке по словам "гениев творчества и науки".

12. В конце скрытого блока должна быть ссылка с текстом "Свернуть", при щелчке по которой блок скрывается.

13. В конце документа добавьте форму с вопросом "Назовите ближайший город, из которого можно приехать на о. Валаам" (правильный ответ — Сортавала). Выделите вопрос с помощью тэга **strong**. Сделайте так, чтобы после щелчка по кнопке *Ответить* на экран выдавалось сообщение "*Правильно*" или "*Неправильно*".



Полная версия (развернутый вид):

Заключение

В статье представлены практические работы по теме "Создание веб-сайтов", входящие в практикум к учебнику углубленного уровня для 10–11-х классов [3–4]. В отличие от многих существующих аналогов они представляют собой введение в методы разработки современных веб-сайтов на основе HTML (включая элементы HTML5), каскадных таблиц стилей (CSS) и языка программирования *Javascript* . "За бортом" рассмотренного материала остались различные *Javascript*-библиотеки (*JQuery* и т.п.), а также серверные языки (например, PHP), которые заинтересованные школьники могут изучать факультативно, на элективных курсах или самостоятельно.

Литература

1. *Дуванов А.А.* Web-конструирование. HTML. СПб.: BHV-Санкт-Петербург, 2003.
2. *Дуванов А.А.* Web-конструирование. DHTML. СПб.: BHV-Санкт-Петербург, 2003.
3. *Поляков К.Ю., Еремин Е.А.* Информатика. 10-й класс. Углубленный уровень. В двух частях. М.: Бином, 2013.
4. *Поляков К.Ю., Еремин Е.А.* Информатика. 11-й класс. Углубленный уровень. В двух частях. М.: Бином, 2013.

Одна программа для всех интерактивных устройств в вашем классе

MimioStudio

Простое и удобное программное обеспечение для интерактивных досок любых производителей и множества интерактивных устройств

MimioStudio 11 распространяется не только с интерактивным оборудованием Mimio. Вы можете приобрести MimioStudio для того, чтобы создавать и использовать уроки на интерактивной доске любого производителя.

Создавайте новые уроки и используйте свои старые наработки, ведь MimioStudio позволяет импортировать уроки из большинства распространенных форматов.

Вся мощь и удобство MimioStudio 11 теперь доступны любому учителю!

Есть как минимум пять весомых причин приобрести MimioStudio отдельно или получить в комплекте с интерактивным оборудованием семейства MimioClassroom:

1. Одна и та же программа позволяет управлять интерактивными досками практически любых производителей (не только Mimio): создавать и демонстрировать уроки, осуществлять групповую работу с доской и многое другое. Поддерживается использование нескольких стилусов, функций single touch, multi-touch и жестов. *Если у вас есть доска, которой вы не пользуетесь по причине несовершенного или отсутствующего ПО, MimioStudio — это отличный шанс дать вашей доске вторую жизнь! Хорошее оборудование не должно пылиться без дела.*
2. Импорт файлов из форматов SMART, Promethean, IWB, PowerPoint и даже Word позволяет использовать уже имеющиеся у вас и ваших коллег содержательные материалы.
3. Модуль ActivityWizard помогает за считаные минуты создавать педагогичные и увлекательные учебные занятия, используя встроенный банк знаний.
4. Встроенный журнал успеваемости, который автоматически фиксирует ответы учащихся при использовании системы тестирования MimioVote избавляет преподавателей от необходимости проверки тестов вручную.
5. Вы можете использовать MimioStudio на любом компьютере: полная совместимость с операционными системами Windows от XP до Windows 8, Mac OS 10.6, 10.7 и 10.8, Linux Fedora и Ubuntu.

Вы можете бесплатно протестировать MimioStudio 11. Свяжитесь с нами, чтобы узнать подробности:

http://www.mimioclass.ru

8 (800) 5555-33-0

Звонок по России бесплатный

mimio
a better way to learn

ООО «Рене» — генеральный дистрибьютор Mimio в России

журнал

Информатика – Первое сентября

Дополнительные материалы – в Личном кабинете

Вниманию подписчиков бумажной версии журнала!

В новом году вы сможете получать дополнительные материалы для практического использования (презентации, раздаточные материалы, образовательное видео) в своем Личном кабинете. Код доступа будет приходить вам вместе с очередным номером журнала.

Откройте Личный кабинет на сайте www.1september.ru:

– введите код доступа;

– скачивайте материалы, распечатывайте;

– используйте прямо на уроке!

Подписка на бумажную версию («Почта России»)

– бумажная версия журнала (доставка по почте)
– индивидуальный код доступа к электронной версии журнала и дополнительным материалам в Личном кабинете на сайте www.1september.ru

Подписка на CD-версию («Почта России»)

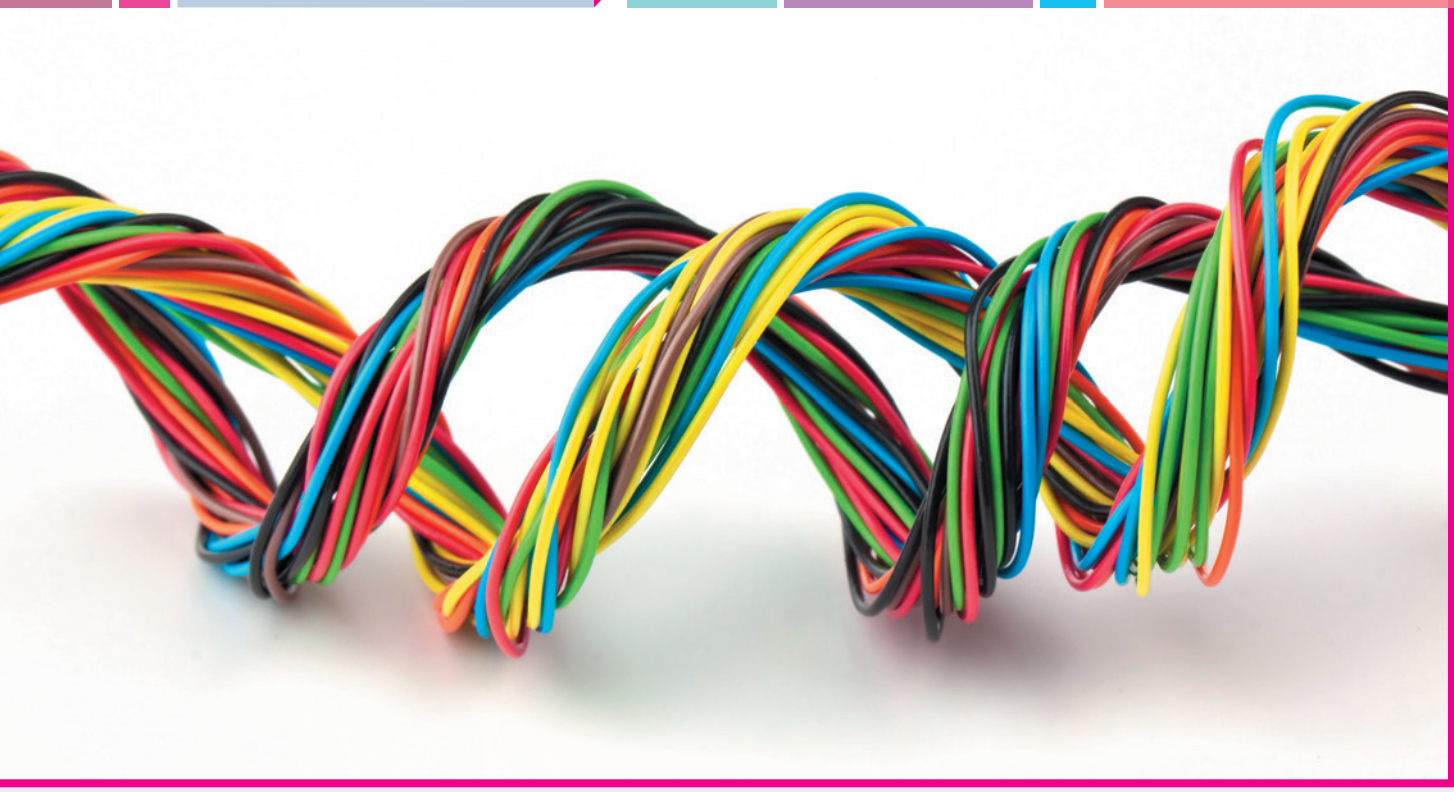
– электронная версия журнала на CD (доставка по почте)
– индивидуальный код доступа к электронной версии журнала и дополнительным материалам в Личном кабинете на сайте www.1september.ru

Подписка на электронную версию (www.1september.ru)
Участие в проекте «Школа цифрового века» (digital.1september.ru)

– электронная версия журнала и дополнительные материалы в Личном кабинете на сайте www.1september.ru

При оформлении подписки на сайте www.1september.ru оплата производится по квитанции в отделении банка или электронными платежами on-line

VISA MasterCard QIWI WebMoney



Методика решения задачи С4 из демонстрационного варианта ЕГЭ по информатике и ИКТ 2014 года

Д.М. Златопольский,
Москва

Условие

По каналу связи передается последовательность положительных целых чисел, все числа не превышают 1000. Количество чисел известно, но может быть очень велико. Затем передается контрольное значение последовательности — наибольшее число *R*, удовлетворяющее следующим условиям:

1) *R* — произведение двух различных переданных элементов последовательности (“различные” означает, что не рассматриваются квадраты переданных чисел; допускаются произведения различных элементов последовательности, равных по величине);

2) *R* делится на 21.

Если такого числа *R* нет, то контрольное значение полагается равным 0.

В результате помех при передаче как сами числа, так и контрольное значение могут быть искажены.

Напишите эффективную, в том числе по используемой памяти, программу (укажите используемую версию языка программирования, например, Borland Pascal 7.0), которая будет проверять правильность контрольного значения.

Программа должна напечатать отчет по следующей форме:

```
Вычисленное контрольное
          значение: …
Контроль пройден (или Контроль
                     не пройден)
```

Перед текстом программы кратко опишите используемый вами алгоритм решения.

На вход программе в первой строке подается количество чисел *N*. В каждой из последующих *N* строк записано одно натуральное число, не превышающее 1000. В последней строке записано контрольное значение.

Пример входных данных:

```
6
70
21
997
7
9
300
21000
```

Пример выходных данных для приведенного выше примера входных данных:

```
Вычисленное контрольное значение: 21000
Контроль пройден
```

Решение

Прежде всего заметим, что требование в условии, связанное с эффективностью программы по используемой памяти (особенно с учетом указания, что количество чисел исследуемой последовательности может быть очень велико), исключает возможность применения в программе массива для хранения чисел последовательности. Нужно обрабатывать каждое число сразу после его ввода в программу. Эта особенность уже стала традиционной в задачах С4 ЕГЭ по информатике и ИКТ последних лет.

Далее обсудим, какие числа последовательности могут повлиять на результат.

Произведение двух чисел делится на 21 в двух случаях:

1) если один из сомножителей делится на 21 (второй может быть любым, в том числе кратным 21, кратным 7 или кратным 3);

2) если ни один из сомножителей не делится на 21, причем один из сомножителей делится на 7, а другой — на 3.

Значит, в ходе обработки чисел последовательности нужно искать и запоминать значения четырех величин:

1) самого большого числа, кратного 3, но не кратного 7 (в программе на школьном алгоритмическом языке имя величины — *макс*_3);

2) самого большого числа, кратного 7, но не кратного 3 (*макс*_7);

3) самого большого числа, кратного 21 (*макс*_21);

4) *макс* — самого большого числа последовательности, отличного от *макс*_21 (если число *макс*_21 встретилось более одного раза и оно же является максимальным среди всех чисел, то *макс* = *макс*_21).

В программе используем также (кроме количества чисел *N* и контрольного значения *R*) величины:

— *очер* — очередное число обрабатываемой последовательности;

— *контр* — контрольное значение, рассчитанное в программе по результатам обработки полученной последовательности.

После того как все данные прочитаны и определены 4 перечисленные величины, искомое контрольное значение *контр* вычисляется как максимальное из произведений *макс*_21 · *макс* и *макс*_3 · *макс*_7.

Итак, схема программы такая:

```
1. Ввод значения N.
2. Задание начальных (нулевых) значений
   величин макс_3, макс_7, макс_21 и макс.
3. Цикл для i от 1 до N
     3.1. Ввод очередного числа
          последовательности очер.
     3.2. Проверка числа очер на предмет
          возможного уточнения
          значения величины макс_3.
     3.3. Проверка числа очер на предмет
          возможного уточнения
          значения величины макс_7.
     3.4. Проверка числа очер на предмет
          возможного уточнения
          значения величин макс_21 и макс.
4. Ввод входного значения R.
5. Определение контрольного значения контр.
6. Вывод результата:
   1) контрольного значения контр;
   2) сообщения о том, пройден контроль
      или нет.
```

Определение значений *макс*_3 и *макс*_7 является достаточно простым:

```
|Проверяем, кратно ли значение очер трем и
не кратно семи
если mod(очер, 3) = 0 и mod(очер, 7) > 0
  то
    |Сравниваем его со значением макс_3
    если очер > макс_3
     то |Встретилось новое значение макс_3
        макс_3 := очер
    все
все
|Проверяем, кратно ли значение очер семи
|и не кратно трем
если mod(очер, 7) = 0 и mod(очер, 3) > 0
  то
    |Сравниваем его со значением макс_7
    если очер > макс_7
     то |Встретилось новое значение макс_7
        макс_7 := очер
    все
все
```

Более сложным является определение значений *макс* и *макс*_21.

Рассмотрим два основных случая:

1) значение *очер* кратно 21 (и варианты для него);

2) значение *очер* не кратно 21.

1. Значение *очер* кратно 21

1.1. И больше *макс*_21 (например, *очер* = 420)

Рассмотрим также возможные варианты значений *макс* и *макс*_21 в какой-то момент в ходе выполнения программы — такие, как в таблице:

макс	180	210	230
*макс*_21	210	210	210

В этом случае новые значения *макс* и *макс*_21 будут, соответственно, следующими (измененные значения выделены цветом):

макс	210	210	230
*макс*_21	420	420	420

В этом варианте:
— значение *макс*_21 меняется безусловно,
— значение *макс* может измениться, если *макс_21* было больше *макс*.

Внимание! Важен порядок изменения значений (см. программу ниже).

1.2. Значение *очер* не больше *макс*_21 (например, *очер* = 210 или *очер* = 42)

1.2.1. Если *очер* = 210 (*очер* = *макс*_21)

При тех же “старых” значениях *макс* и *макс*_21:

макс	180	210	230
*макс*_21	210	210	210

— новые значения будут такими:

макс	210	210	230
*макс*_21	210	210	210

В этом варианте значение *макс* может измениться, если *очер* больше *макс*.

1.2.2. Если *очер* = 42 (*очер* < *макс*_21) и “старые” значения *макс* и *макс*_21 — как в таблице:

макс	21	42	65
*макс*_21	210	210	210

то новые значения *макс* и *макс*_21:

макс	42	42	65
*макс*_21	210	210	210

В этом варианте значение *макс* также может измениться, если *очер* больше *макс*.

2. Значение *очер* не кратно 21

В этом случае может измениться только значение *макс* (если *очер* > *макс*).

Соответствующий фрагмент, в котором реализованы сделанные рассуждения:

```
|Проверяем, кратно ли значение очер
|двадцати одному
если mod(очер, 21) = 0
  то
    |Сравниваем его со значением макс_21
    если очер > макс_21
     то |Встретилось новое значение макс_21
        |Проверяем, надо ли менять
        |значение макс
        если макс_21 > макс
          то |Меняем значение макс
            макс := макс_21
        все
        |А теперь меняем значение макс_21
        макс_21 := очер
     иначе |очер <= макс_21
        |Сравниваем его со значением макс
        если очер > макс
          то |Меняем значение макс
            макс := очер
        все
    все
  иначе |Значение очер не кратно 21
  |Сравниваем его со значением макс
  если очер > макс
    то |Меняем значение макс
      макс := очер
  все
все
```

Можно также оформить программу согласно сделанным рассуждениям по-другому — использовать сложные условия. Анализ показывает, что значение *макс*_21 меняется, а значение *макс* может измениться, если очередное число *очер* кратно 21 и больше “старого” значения *макс*_21. Иначе может измениться только значение *макс* (если *очер* больше “старого” значения *макс*). Соответствующий, более компактный, фрагмент:

```
если mod(очер, 21) = 0 и очер > макс_21
  то
    если макс_21 > макс
      то
        макс := макс_21
    все
    макс_21 := очер
  иначе
    если очер > макс
      то
        макс := очер
    все
все
```

Именно в таком виде оформлен фрагмент, связанный с определением значений *макс* и *макс*_21, в примерах программ на языках Паскаль и Бейсик, приведенных в демонстрационном варианте.

Вся программа решения задачи:

```
алг Задача_С4_демо_2014
нач цел N, R, очер, макс_3, макс_7, макс_21,
макс, контр, i
|Ввод количества чисел последовательности
вывод нс, "Введите количество чисел N "
ввод N
|Задание начальных значений искомых величин
макс_3 := 0
макс_7 := 0
макс_21 := 0
макс := 0
нц для i от 1 до N
  |Ввод и обработка чисел последовательности
  вывод нс, "Введите очередное число
             последовательности "
  ввод очер
  |Проверяем, кратно ли значение очер трем
  |и не кратно семи
  если mod(очер, 3) = 0 и mod(очер, 7) > 0
    … (см. выше)
  |Проверяем, кратно ли значение очер семи
  |и не кратно трем
  если mod(очер, 7) = 0 и mod(очер, 3) > 0
```

```
    … (см. выше)
  |Уточняем (при необходимости) значения
  |макс_21 и макс
    если mod(очер, 21) = 0 и очер > макс_21
    … (см. выше)
кц
|Ввод входного контрольного значения
вывод нс, "Введите контрольное число "
ввод R
|Определяем расчетное контрольное значение
если макс_21 * макс > макс_3 * макс_7
  то
    контр := макс_21 * макс
иначе
    контр := макс_3 * макс_7
все
|Вывод результатов
вывод нс, "Введите контрольное число "
если R = контр
  то
    вывод нс, "Контроль пройден"
  иначе
    вывод нс, "Контроль не пройден"
все
кон
```

Полностью программы на школьном алгоритмическом языке (в двух вариантах фрагмента, связанного с возможным уточнением значений *макс*_21 и *макс*) представлены на диске к данному выпуску журнала.

В заключение заметим, что приведенная в разделе “Система оценивания экзаменационной работы по информатике и ИКТ” демоварианта ЕГЭ фраза о том, что в примере программы на языке Бейсик* используется алгоритм, немного отличающийся от описанного выше, относится только к фрагменту, связанному с определением значений *MAX* (*макс*) и *M21* (*макс*_21):

```
IF DAT MOD 7 = 0 AND DAT > M7 THEN
  M7 = DAT
ELSE
  IF DAT MOD 3 = 0 AND DAT > M3 THEN
    M3 = DAT
  END IF
END IF
```

Видно, что в данном случае применен вложенный условный оператор.

Заметим также, что в программе на языке Паскаль во фрагментах, относящихся к возможному уточнению значений *M7* (*макс*_7) и *M3* (*макс*_3):

```
if ((dat mod 7) = 0)
     and ((dat mod 3) > 0) and (dat > M7)
  then M7 := dat;
if ((dat mod 3) = 0)
     and ((dat mod 7) > 0) and (dat > M3)
  then M3 := dat;
```

— использование скобок в выражениях `(dat mod 7)` и `(dat mod 3)` является излишним.

* Программы на языках Бейсик и Паскаль также представлены на диске.

Министерство образования и науки Пермского края
Пермский государственный национальный исследовательский университет
Учебный центр “ИНФОРМАТИКА”
Компьютерная школа при ПГНИУ
Издательство “БИНОМ. Лаборатория знаний”

ИНФОРМАЦИОННОЕ ПИСЬМО*

6–7 февраля 2014 г. в г. Перми состоится
Всероссийская научно-методическая конференция с международным участием
“Информатика в школе: прошлое, настоящее и будущее”

Уважаемые коллеги!

Приглашаем Вас принять участие в работе Всероссийской научно-методической конференции “Информатика в школе: прошлое, настоящее и будущее”, которая состоится 6–7 февраля 2014 г. в г. Перми. Место проведения конференции — Пермский государственный национальный исследовательский университет.

К участию в конференции приглашаются все специалисты, имеющие отношение к организации обучения информатике в общеобразовательной школе: ученые-методисты, учителя информатики, авторы учебников, издатели учебно-методической литературы по информатике, преподаватели вузов и системы повышения квалификации, занимающиеся подготовкой учителей информатики, студенты педагогических специальностей.

Проведение конференции приурочено также к юбилейной дате: 20-летию деятельности в области школьной информатики пермского авторского коллектива в составе: Семакин И.Г., Хеннер Е.К., Русаков С.В., Залогова Л.А., Шестакова Л.В., Шестаков А.П., Шеина Т.Ю., Плаксин М.А., Русакова О.Л., Иванова Н.Г. и др. Работа пермских авторов происходит в сотрудничестве с московским издательством “БИНОМ. Лаборатория знаний”, вместе с которым выпущено более 25 учебников и учебных пособий по информатике для всех ступеней школьного образования.

Адрес оргкомитета: 614990, г. Пермь, ул. Генкеля, д. 7, ПГНИУ, корп. 2, ауд. 518.

Контактные лица оргкомитета:

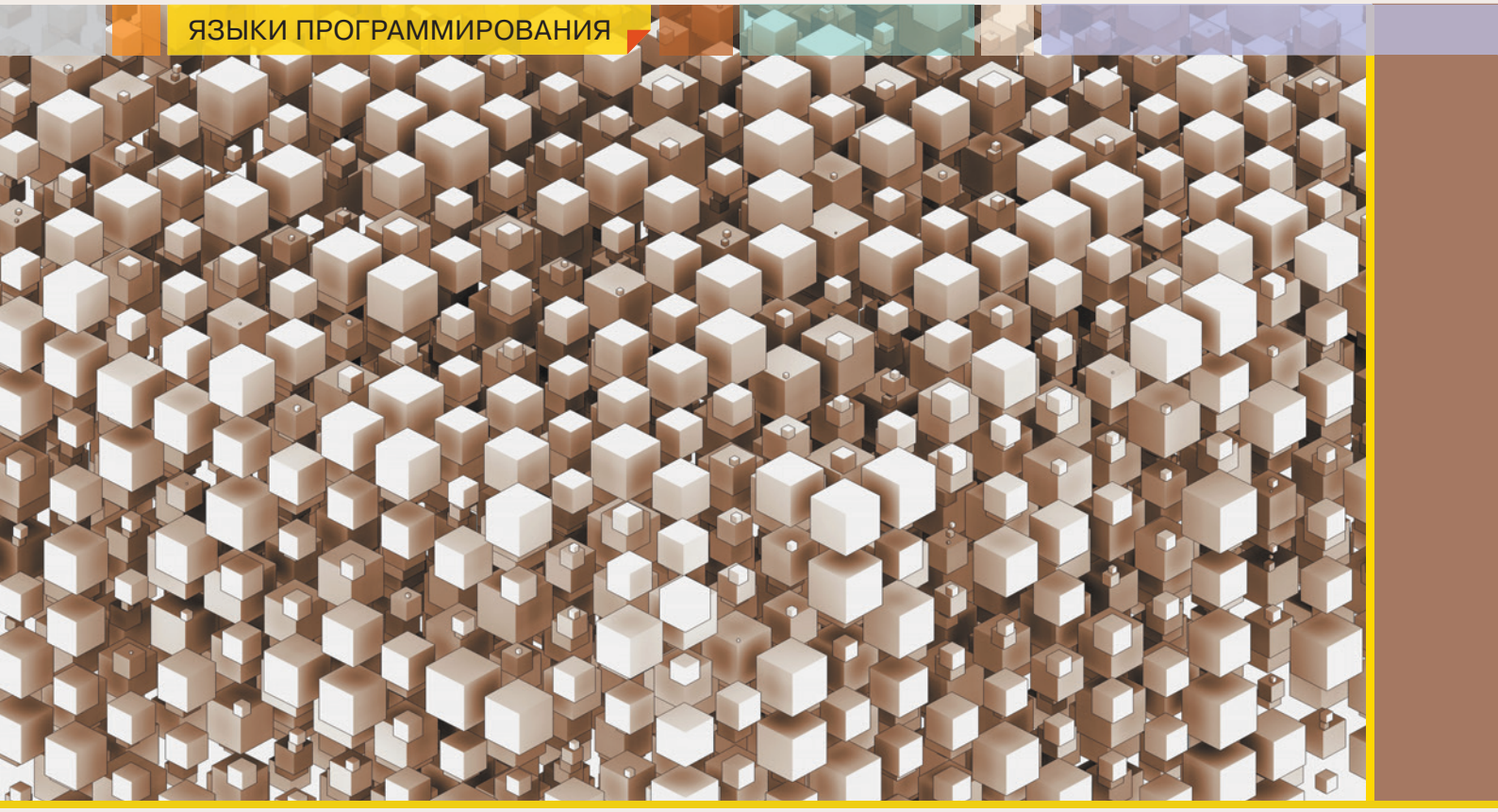
Татьяна Юрьевна Шеина — E-mail: tsheina@yandex.ru, тел. (342) 239-63-84.

Татьяна Николаевна Соловьева — E-mail: solovevatn@ya.ru, тел. (342) 239-61-39.

Предварительная регистрация участников конференции производится на сайте конференции: http://confer.cschool.perm.ru с 25 октября 2013 г. по 20 января 2014 г. или по E-mail: tsheina@yandex.ru.

Информация на сайте конференции будет доступна до 28 февраля 2014 г.

* Печатается в сокращении. Полная версия письма и другая информация — на сайте конференции.



A, J, K

И.А. Сукин,
г. Переславль-Залесский

Введение

В этой статье, лаконично озаглавленной всего тремя буквами латинского алфавита, речь будет идти о нескольких столь же лаконичных языках программирования. Искушенный читатель сразу может понять, что это APL и его родственники: J, K и многие другие. Эти языки принято относить к классу векторных[1], поскольку программисты на них имеют дело с данными, структурированными в виде многомерных прямоугольных однородных таблиц, которые называются массивами[2]. Вы можете сказать: “Что же в этом особенного? В своем любимом языке программирования я тоже могу свободно пользоваться массивами и определять для них операции и функции”. Принципиальная разница состоит в том, что программирование на векторных языках практически полностью состоит из различных вычислений над массивами и векторами, вследствие чего программист начинает думать о прикладных задачах несколько иначе, учитывая эти “массивно-ориентированные” особенности.

История векторных языков программирования начинается в конце пятидесятых годов двадцатого века (в 1957 г.), когда Кеннет Айверсон начал работы над созданием новой нотации для записи математических выражений. Айверсону нужен был инструмент для ясной и лаконичной записи задач, связанных с обработкой больших объемов данных. Результатом работы Айверсона стал язык, названный им просто “Язык программирования” (“***A Programming Language***”), первая буква английского названия которого вскоре перестала быть просто артиклем и получила собственный смысл, а само название было сокращено до APL. Впервые APL был описан в одноименной книге 1962 года издания. Айверсон в то время уже работал в IBM, которая стала колыбелью APL и является ей до сегодняшнего времени. APL использовал особый набор символов, что послужило причиной создания специальных APL-клавиатур и APL-терминалов. Долгое время язык варился в собственном соку, в него добавлялись небольшие по объему правки, выходили такие же небольшие стандарты (самым крупным из них был APL2), а пользователи и разработчики сформировали тесное сообщество, не слишком дружелюбное по отношению к внешнему

[1] Это не совсем верный термин, поскольку они “массивно-ориентированные” (array), но в русской литературе устоялось именно прилагательное “векторный”. Может быть, полезно было бы использовать слово “тензорный”.

[2] При этом любой массив можно рассматривать как вектор, о чем будет сказано позже.

миру, но, если можно так выразиться, уютное изнутри. Важной чертой APL-сообщества всегда было преобладание в нем высококвалифицированных (и высокооплачиваемых) разработчиков.

Небольшой бум в сфере векторных языков произошел в 90-е годы XX века, когда ученик Айверсона, Роджер Хуэй, вместе со своим учителем создал J — вариант APL, ориентированный на работу с набором ASCII-символов и призванный исправить все то, что за полвека показалось APL-программистам неправильным. С тех пор сообщество было поделено на две части: профессионалы и индустрия продолжали использовать оригинальный APL, ученые же стали все больше и больше присматриваться к J и экспериментировать с ним. В 1993 году еще один известный в среде APL-программистов специалист, Артур Уитни, для собственной компании Kx Systems разработал язык K, вобравший в себя как плюс мира векторных языков, так и становившиеся в то время все более популярными функциональные средства разработки, такие, как лямбда-выражения. Для K также была разработана база данных kdb, позволяющая легко и эффективно хранить огромные массивы числовых данных. K получил свою достаточно узкую нишу в сфере финансов, среди его пользователей крупнейшие банки мира: Goldman Sachs, Morgan Stanley, Deutsche Bank, Bank of America. Уже в 2003 году K получил преемника в виде языка Q, в котором Уитни упростил синтаксис, сделав его более понятным для тех, кто не слишком близок к APL-сообществу.

В настоящее время появилось немало энтузиастов, экспериментирующих и создающих свои векторные языки, смешивая парадигму, ориентированную на работу с массивами, с концепциями стека, ООП, другими математическими теориями.

Многообразие векторных языков

Как я уже сказал ранее, сейчас ситуация в сфере векторного программирования уже совсем не та, что была во времена создания APL и даже через 10–20 лет после этого. Любители и профессионалы все чаще и чаще изобретают свои векторные языки. В этом обзоре я не буду пытаться охватить все возможные реализации всех возможных языков. Перечислю лишь те из них, на которые действительно стоит обратить внимание:

- APL, оригинальная версия до сих пор нисколько не потеряла актуальности и широко используется профессионалами;
- A+, несколько расширенная версия APL от Артура Уитни;
- NARS2000, реализация APL, предоставляющая некоторые новые возможности, такие, как поддержка вложенных массивов (она есть, однако, и в APL2), массивов из разнородных данных, комплексных чисел;
- J, наиболее перспективный в настоящее время векторный язык программирования;
- K, язык, объединяющий в себе преимущества APL, J и функционального программирования;
- Nial, векторный язык, в котором для именования операций вместо одиночных и двойных символов используются полноценные слова. Интерпретатор Nial является компьютерной реализацией так называемой “теории массивов” Тренчарда Мора и Майка Дженкинса;
- Eli, еще одна вольная интерпретация оригинального APL со своими расширенными возможностями;
- Lang5, смелая попытка объединения стековой парадигмы (Forth, Factor, Joy) с векторной, один из самых новых языков в этой семье.

APL

APL изначально был создан как мощный язык программирования для тех, кто является специалистом в своей прикладной области, но знает не так уж и много об общении с компьютером и внутреннем устройстве вычислительной машины. Мы последуем за мыслью Кеннета Айверсона и не мешкая приступим к рассмотрению собственно конструкций языка и приемов разработки на нем.

В качестве тестового интерпретатора можно использовать учебную версию Dyalog APL или же созданный Dyalog веб-интерпретатор, расположенный по адресу *http://tryapl.org*, также можно воспользоваться другим онлайн-интерпретатором[3] — *http://ngn.github.io/apl/web/index.html*, для простых примеров может также подойти NARS2000, однако в нем не реализованы некоторые операции, которые могут нам впоследствии понадобиться[4].

При записи кода на APL (и других векторных языках) вводимые строки будут сдвинуты на несколько символов от начала, а ответ интерпретатора останется несдвинутым. Как и в случае работы с REPL интерпретатора любого другого языка программирования, после ввода каждой строки кода необходимо нажимать клавишу Enter для получения ответа системы.

Работа внутри APL-системы происходит в рамках так называемых “рабочих пространств” (workspaces). Выражения в каждом рабочем пространстве имеют соответствующий этому пространству набор объявленных функций и переменных. Рабочие пространства обычно можно сохранять и загружать (онлайн-интерпретаторы, приведенные выше, имеют только одно рабочее пространство, без возможности сохранения).

Первые шаги

Первое, что можно попытаться сделать, — это использовать APL в качестве продвинутого калькулятора:

```
      27 + 53
80
      1271 - 708
562
      59 × 8
475
      86 ÷ 4
21.5
```

Как вы уже можете заметить, умножение и деление представляются в APL особыми символами “×” и “÷”, вместо привычных многим астериска-звездочки “*” и слэша “/”. Тем не менее это не самое необычное и страшное, с чем вам предстоит столкнуться в мире APL.

Все вычисления в APL происходят справа налево, а не так, как вы привыкли в математике, это позволяет упростить интерпретатор и понимание порядка выполнения даже довольно сложных операций.

[3] Или скачать его исходный код на Javascript и запустить у себя в браузере или на Node.js.

[4] Особенности ввода APL-символов в каждой реализации обычно наглядно описаны в файлах помощи. Большинство реализаций поддерживает экранную клавиатуру APL.

Для создания переменных и присваивания используется символ стрелки влево, слева от которой находится имя переменной, а справа — данные, которые необходимо присвоить. К примеру:

```
VAT ← 0.18 ⍝ Читается как: Присвоить
           значение 0.18 переменной VAT
Years ← 1952 1943 1986 2005
```

Вторая переменная содержит вектор — одномерный массив данных. Для того чтобы посмотреть содержимое переменной, достаточно лишь ввести ее имя и нажать Enter:

```
      Years
1952 1943 1986 2005
```

Символ каждой операции в APL имеет свое, иногда необычное, иногда даже, если можно сказать, "романтическое" имя на английском языке (попытка перевода этих имен не приведет ни к чему хорошему). Так, символы "×", "÷" и "←" называются "Times", "ColonBar" и "LeftArrow" соответственно, а символ, отмечающий начало однострочного комментария "⍝", носит имя "Lamp". В дальнейшем я при каждом удобном случае буду называть такие мнемонические имена вводимых символов и операций.

Глобальный подход

Одной из важнейших особенностей APL является возможность одновременного применения операций к двум наборам данных, имеющим одинаковую "форму". Введем, к примеру, две векторные переменные из шести элементов, первая будет содержать цены, а вторая -— соответствующие количества купленного товара:

```
      Price ← 5.2  11.5  3.6  4  8.45 6.4
      Qty   ← 2     1    3    6  2    7
```

Если мы попробуем их перемножить, то операция умножения будет применена поэлементно, а результатом станет массив, имеющий ту же форму, что и два исходных:

```
      Costs ← Price × Qty
      Costs
10.4 11.5 10.8 24 16.9 44.8
```

Такой глобальный подход к обработке данных делает ненужным применение циклических операторов в подавляющем большинстве случаев. Все вышесказанное остается верным и для многомерных массивов более сложной формы, чем векторы.

К примеру, финансист может сделать прогноз по продажам четырех продуктов, производимых фирмой на следующие шесть месяцев, и внести соответствующие значения в переменную Forecast. В конце срока он может сгруппировать реальные данные о продажах и сохранить их в переменной Actual.

```
      Forecast                    Actual
150 200 100  80  80  80     141 188 111  87  82  74
300 330 360 400 500 520     321 306 352 403 497 507
100 250 350 380 400 450     118 283 397 424 411 409
 50 120 220 300 320 350      43  91 187 306 318 363
```

После этого он может захотеть посчитать разницу между реальными и предсказанными данными, что будет легко сделать, набрав в интерпретаторе APL:

```
      Actual-Forecast
¯9 ¯12 ¯11  7  2  ¯6
21 ¯24  ¯8  3 ¯3 ¯13
18  33  47 44 11 ¯41
¯7 ¯29 ¯33  6 ¯2  13
```

Здесь нужно отметить, что для представления отрицательных чисел в APL используется символ "¯", имеющий очевидное название "Overbar". Он является частью числового литерала, и его следует отличать от унарного минуса "-".

В традиционных языках программирования для реализации подобного вам потребовалось бы два вложенных цикла, как, например, в C:

```
for (i = 0; i < 4; i++) {
  for (j = 0; j < 6; j++) {
    diff[i][j] = actual[i][j] - forecast[i][j];
  }
}
```

При этом задача управления массивами памяти сложной формы со сложной адресацией ложится на плечи программиста. Многие согласятся, что это не слишком разумная плата за простейшую операцию.

Операции, работающие с двумя массивами одинаковой формы, могут также работать с массивом и скаляром. Допустим, в примере выше мы захотим посчитать величину НДС для каждого товара, чья цена входит в массив Price. Вне зависимости от того, напишете вы `VAT × Price` или `же Price × VAT`, результат будет одинаковым:

```
0.936 2.07 0.648 0.72 1.521 1.152
```

Инопланетные символы

Наши возможности в математике (хотелось бы верить) не ограничиваются четырьмя-пятью операциями, представимыми с помощью аналогичного количества символов. Тем не менее большинство привычных традиционных языков программирования долгое время содержало именно столько.

Кеннет Айверсон при создании APl расщедрился и ввел большое число символов для самых разных унарных и бинарных операций. В качестве примера можно рассмотреть операции нахождения максимума (символ "UpStile"):

```
      64.4 ⌈ 94.3
94.3
      23 54 35 29 ⌈ 96 34 43 11
96 54 43 29
```

и минимума (символ "DownStile"):

```
      54 22 57 31 ⌊ 44
44 22 44
```

Всего в APL включено около 70 различных символов и операций, некоторые из которых к тому же имеют разный смысл в зависимости от контекста. Многие из этих символов привычны, некоторые нет, однако все они довольно быстро запоминаются при частой работе с языком.

Двойные смыслы

Как и в математике, некоторые символы в APL имеют двойной смысл. Самым ярким и знакомым всем примером является символ "минус". Бинарный минус в выражении `a = x - y` означает вычитание, унарный же, как в выражении `a = - y`, — отрицание.

В APL бинарные операции называются "диадическими", а унарные — "монадическими". К примеру, для нахождения формы массива применяется символ "⍴" ("Rho", греческая буква "ро"). В монадической форме он находит список измерений массива:

```
⍴ Price    ⍝ Монада: Price состоит из 5 элементов
5
⍴ Forecast ⍝ Монада: Forecast состоит из 4 рядов
             по 6 элементов в каждом
4 6
```

В диадической форме этот же символ обозначает операцию изменения формы массива:

```
      2 3 ⍴ 1 2 3 4 5 6
1 2 3
4 5 6
```

К примеру, для создания таблицы из двух столбцов необходимо сначала указать ее форму: массив 4 2 (четыре строки, два столбца), потом символ соответствующей операции и уже затем сами данные:

```
      Tab ← 4 2 ⍴ 25 60 33 47 11 44 53 28
```

Таким же образом наш финансист мог создать массивы Forecast и Actual из примеров выше.

Терминология

В первом разделе я уже говорил о некоторой путанице как в оригинальной терминологии, так и в ее переводе на русский язык. Сейчас хотелось бы внести большую ясность и уточнить, что "вектором" называется массив с одним измерением, список элементов, такой, как переменные Price и Qty, или строка символов 'Some characters'. Двумерные таблицы типа Forecast и Actual принято называть "матрицами", а одиночные значения вроде числа 32.4 или символа 'R' — "скалярами".

Различные векторные языки по-разному ведут себя с массивами, имеющими несколько измерений, но состоящими из одного элемента. К примеру, возможно создать следующий четырехмерный массив:

```
      1 1 1 1 ⍴ 2
2
```

При этом остается открытым вопрос, будет ли такой массив эквивалентным скаляру 2 или любому другому подобному массиву с другим числом измерений. В большинстве векторных языков эти значения будут эквивалентными, однако есть языки, в которых они отличаются.

Редукция

В предыдущем разделе мы подсчитали стоимость некоторых товаров:

```
10.4 11.5 10.8 24 16.9 44.8
```

Что нам делать в том случае, если мы захотим посчитать суммарную стоимость? В математике для этого давно был введен символ "Σ", обросший кучей индексов сверху и снизу и явно не подходящий для того, чтобы быть набранным на компьютере. В APL операцию суммирования всех элементов вектора можно записать так:

```
      +/ Costs
118.4
```

При этом мы смогли обойтись без указания каких-либо индексов, поскольку они неявно хранятся где-то внутри машины и нам вообще необязательно о них знать.

В среде APL-программистов принято говорить, что мы только что написали "редукцию по сложению" переменной Costs.

Рассмотрим этот процесс подробнее[5]:

- Перепишем наше выражение как `+/ 10.4 11.5 10.8 24 16.9 44.8`
- Оно будет эквивалентно выражению `10.4 + 11.5 + 10.8 + 24 + 16.9 + 44.8`
- Что, в свою очередь, эквивалентно числу `118.4`

В результате мы как бы "вставили" символ "+" между элементами вектора. Можно привести еще несколько таких примеров:

- Возьмем выражение

```
      ×/ 10.4 11.5 10.8 24 16.9 44.8
```

- Это то же самое, что

```
10.4 × 11.5 × 10.8 × 24 × 16.9 × 44.8
```

- Что то же самое, что и число `23470962.28`

или

- Выражение `⌈/ 10.4 11.5 10.8 24 16.9 44.8`
- Эквивалентно выражению

```
10.4 ⌈ 11.5 ⌈ 10.8 ⌈ 24 ⌈ 16.9 ⌈ 44.8
```

- Что эквивалентно числу `44.8`

Редукция принадлежит к особой категории символов, обозначающих "операторы", в то время как другие символы, которые мы уже рассматривали, обозначают "функции". Разница между этими двумя типами синтаксическо-семантических конструкций состоит в том, что аргументами функций являются данные: `Price × Qty`, в то время как левым аргументом операторов являются функции: `+/ Costs`. Как можно видеть, введение операторов еще больше увеличивает глобальность языковых средств, о которой мы говорили ранее.

Ваша первая программа

"Но мы же ее уже написали несколькими разделами ранее!" — скажете вы и будете правы. Однако в APL под понятием "программы" понимается то, что в других языках называется "подпрограммой", или "процедурой", — именованная последовательность языковых конструкций. Тем не менее во избежание путаницы и в силу того, что эти самые "программы" не отличаются по поведению от примитивных функций, часть APL-программистов так и называет их — "функциями". Мы тоже будем придерживаться этой терминологии.

Предположим, что нам необходимо подсчитать среднее значение в массиве:

```
      Val ← 22 37 41 19 54 11 34
```

Для этого мы должны будем поделить результаты вычисления двух выражений:

- выражения `+/Val`, которое вычисляет сумму всех элементов в Val, 218;
- выражения `⍴Val`, вычисляющего число элементов в Val, 7.

Все это может быть описано одним выражением `(+/Val)÷(⍴Val)`. Поскольку возможность вычисления среднего значения может понадобиться нам довольно часто, будет полезно вынести ее в отдельную функцию. Для определения новой функции в текущем рабочем пространстве APL использует символ "∇" ("Del"). Конкретное поведение зависит от реализации языка, но обычно первой строчкой будет:

```
      ∇ R ← Average V
```

При этом после нажатия Enter откроется дополнительное окно[6], в котором можно будет вводить тело функции. Заканчивается тело также символом "∇"[7], при этом содержимое дополнительного окна будет выглядеть примерно так:

```
[0]     R ← Average V
[1]     R ← (+/V)÷(⍴V)
```

Здесь V — аргумент функции. Использовать собственную функцию так же просто, как и встроенные:

[5] Знакомые с функциональным программированием без труда узнают здесь функцию reduce/fold.

[6] Или внутренний интерпретатор.

[7] Это необязательно, а в некоторых реализациях вообще является ошибкой. В NARS2000 достаточно просто закрыть дополнительное окно и нажать "Yes", когда вас спросят, сохранить ли определение.

```
      Average Val
31.1428571428
      Average 12 74 56 23
41.25
```

Она может быть также и частью более сложного выражения:

```
      10×Average 12 74 56 23
412.5
```

Индексация

Индексация массивов в APL синтаксически и семантически близка к подобным операциям в других языках. Имея наш массив

```
      Val← 22 37 41 19 54 11 34
```

мы можем записать Val[4], чтобы получить доступ к его четвертому элементу. Однако в отличие от многих других языков APL позволяет получить сразу несколько элементов за одну операцию:

```
      Val
22 37 41 19 54 11 34
      Val [2 4 7 1 4] ⍝ один из элементов
                      мы берем два раза
37 19 34 22 19
```

Подобные рассуждения справедливы и для присваивания значений элементам массива. При этом массив присваиваемых значений должен быть либо скаляром, либо иметь ту же форму, что и массив, используемый для индексации:

```
      Val[3 5 1] ← 300 77 111
      Val
111 37 300 19 77 11 34
```

Однако если нам понадобится извлечь 500 первых элементов из массива, было бы слишком наивно перечислять их индексы как есть. Это одна из причин, по которой в APL включен символ “⍳” (“iota”, греческая буква “йота”). В монадическом контексте этот символ генерирует массив из первых n целых чисел, где n — аргумент операции ⍳. Таким образом, вместо 1 2 3 4 5 6 7 8 мы можем написать ⍳8, а для индексации большого массива подойдет, например, такое выражение: BigArr[⍳500].

Программирование без подпрограмм

Немалое количество вычислений в APL, как и в системах компьютерной алгебры вроде Matlab, может быть произведено без создания функций-подпрограмм. Рассмотрим следующий пример. Предположим, что зарплаты на предприятии сгруппированы в три категории с номерами 1 2 3, соответственно. Некто (допустим, бухгалтер) создает две переменные, одну для собственно числовых значений зарплаты, другую для категорий:

```
   Salaries ← 42250 16190 37060 22400 20760 13890
39160 39180 49390 27350
   Categories ←   3     1     3     2     2     1
3     3     3     2
```

Перед бухгалтером встает задача обновления значений зарплаты с течением времени (в связи с инфляцией, например). При этом величина (в процентах), на которую увеличивается зарплата, различна для каждой категории: для категории 1 это 8%, для 2 — 5%, для 3 — 2%. Сколько денег нужно для того, чтобы повысить зарплату всем сотрудникам? Создадим переменную, содержащую значения темпов роста зарплаты:

```
      Rates ← 8 5 2 ÷ 100
      Rates
0.08 0.05 0.02
```

Поскольку первый сотрудник относится к третьей категории, его зарплату нужно увеличить на два процента:

```
      Rates[3]
0.02
```

Первые пять сотрудников относятся к категориям 3 1 3 2 2, соответственно, поэтому их зарплаты необходимо увеличить на:

```
      Rates[3 1 3 2 2]
0.02 0.08 0.02 0.05 0.05
```

Чтобы получить темпы роста зарплаты для всех сотрудников, мы можем ввести:

```
      Rates[Categories]
0.02 0.08 0.02 0.05 0.05 0.08 0.02 0.02 0.02 0.05
```

Теперь мы можем посчитать конкретные суммы, на которые необходимо поднять зарплату каждому:

```
      Salaries × Rates[Categories]
  845  1295.2  741.2  1120  1038  1111.2  783.2  783.6
987.8  1367.5
```

Сложив эти значения, мы получим сумму, которую необходимо потратить, чтобы поднять всем сотрудникам зарплату:

```
      +/ Salaries × Rates[Categories]
10072.7
```

При этом можно отметить, что процесс, описанный в этом разделе, обладает следующими свойствами:

- нам не важно количество сотрудников и категорий;
- мы можем получить результат без создания подпрограмм;
- итоговое выражение похоже на фразу на естественном языке: “Найти сумму всех зарплат, умноженных на скорость роста, соответственно категориям”.

Во многих других языках мы не смогли бы так просто придумать такое элегантное и в то же время лаконичное решение.

Наши бинарные друзья

Бинарные операции — диады — являются очень важной частью APL и всех других векторных языков программирования. Особое значение имеют бинарные операции, результатами которых становятся бинарные данные, массивы, чьи элементы могут быть равными только нулю или единице. Самые простые примеры таких операций — это операции сравнения:

```
      Salaries > 3000
1 0 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 0 0 0
      Actual > Forecast
0 0 1 1 1 0
1 0 0 1 0 0
1 1 1 1 1 0
1 0 0 1 0 1
```

В APL имеются символы для всех основных математических операций отношения между числами: < ≤ = ≥ > ≠. В других языках программисты вынуждены использовать комбинации ASCII-символов, от более-менее понятного “<=” до совершенно магического “<>”.

Для работы с бинарными (двоичными) данными в APL используются стандартные булевы логические операции:

- ∧, конъюнкция, логическое И;
- ∨, дизъюнкция, логическое ИЛИ;
- ~, инверсия, логическое НЕ;
- ≠, исключающее ИЛИ, XOR.

Во многих реализациях присутствуют также дополнительные операции:

- ⍱, логическое И с отрицанием;
- ⍲, логическое ИЛИ с отрицанием.

Предположим, что мы (работая в нашем текущем рабочем пространстве) захотим найти тех сотрудников из третьей категории, чья зарплата меньше 40 000 рублей. В этом нам помогут упомянутые диады:

```
      (Categories=3) ∧ (Salaries < 40000)
0 0 1 0 0 0 1 1 0 0
```

Как вы могли заметить, операция XOR является той же самой операцией, что и проверка на неравенство.

Применение двоичных векторов

Двоичный вектор, который мы получили в предыдущем примере, выглядит не очень-то полезным. Тем не менее APL предоставляет удобные средства для работы с такими данными. Например, мы можем использовать такой вектор для того, чтобы подсчитать количество элементов, удовлетворяющих некоторому условию. Допустим, у нас есть следующая операция сравнения:

```
      Salaries < 25000
0 1 0 1 1 1 0 0 0 0
```

Мы сможем запросто посчитать количество людей, чья зарплата меньше двадцати пяти тысяч, с помощью следующего выражения:

```
      +/Salaries < 25000
4
```

Также мы можем использовать двоичные вектора в качестве битовых масок для извлечения данных из массивов:

```
      1 1 0 1 0 0 1/23 55 17 46 81 82 83
23 55 46 83
```

Это работает и для строк (которые, по сути, являются просто векторами символов):

```
      1 1 0 1 0 0 1 0 1 0 0 1 / 'Привет, мир!'
Прв,м!
```

В APL такая операция называется “сжатием”, или “репликацией”. Она полезна в том случае, когда вам необходимо извлечь некоторые данные, руководствуясь неким критерием. Выведем зарплаты всех сотрудников второй категории:

```
      (Categories=2)/Salaries
22400 20760 27350
```

Рассмотрим еще один пример с двоичными векторами. Предположим, что нам нужно определить индексы[8] тех элементов массива Val, значения которых больше 35. Введем следующие строки:

```
      Val
22 37 41 19 54 11 34
      Val>35
0 1 1 0 1 0 0
      ⍴Val
7
      ⍳⍴Val
1 2 3 4 5 6 7
```

Сравните два вектора:

```
      Val>35
0 1 1 0 1 0 0
      ⍳⍴Val
1 2 3 4 5 6 7
```

Можно увидеть, что с помощью первого вектора получится удалить те элементы массива индексов, которые не нужны нам, поскольку соответствующие им элементы Val не удовлетворяют нашему условию. Итоговое выражение будет выглядеть следующим образом:

```
      (val>35)/⍳⍴Mal
2 3 5
```

[8] Индексы массивов в APL начинаются с единицы.

Подобные конструкции можно использовать во многих случаях. Предположим, что нам надо найти индексы символов 'a' в строке:

```
      Phrase ← 'Предположим, что нам надо найти
индексы символов'
      (Phrase='а')/⍳⍴Phrase
19 23 28
```

Игра продолжается

Чуть выше мы нашли индексы всех символов 'a' в заданной строке. Предположим теперь, что мы хотим найти индексы всех гласных. Мы могли написать (Phrase='a'), поскольку 'a' — скаляр, однако выражение (Phrase='аеёиоуыэюя'), поскольку интерпретатор APL подумает, что мы захотим выполнить поэлементное сравнение. В качестве решения можно выполнить множество сравнений, однако такой подход будет совершенно неаккуратным и не подходит нам. Очевидно, что в данной ситуации необходимо ввести новую операцию, определяющую отношение принадлежности. Символом для этой операции служит привычный многим математикам “∈”.

Выражение `A ∈ B` возвращает в качестве результата двоичный вектор — маску, соответствующий тем элементам массива A, которые являются членами массива B, вне зависимости от формы A и B. К примеру:

```
      5 7 2 8 4 9 ∈ 3 4 5 6
1 0 0 0 1 0
```

или

```
      'возвращает' ∈ 'варю'
1 0 0 1 1 1 0 1 0 0
```

Наш пример, таким образом, можно переписать как:

```
      (Phrase ∈ 'аеёиоуыэюя')/⍳⍴Phrase
3 6 8 10 16 19 23 25 28 31 33 36 39 42 45 47
```

Таким же способом можно проверить и принадлежность элементов матрицы к некоему массиву[9]:

```
      Towns
москва
петербург
рязань
ярославль
иркутск
      Towns ∈ 'аеёиоуыэюя'
0 1 0 0 0 1 0 0 0
0 1 0 1 0 0 1 0 0
0 1 0 1 0 0 0 0 0
0 0 1 0 0 1 0 0 0
1 0 0 1 0 0 0 0 0
```

Результат операции “∈” имеет ту же форму, что и левый аргумент:

```
      'аеёиоуыэюя' ∈ Towns
1 1 0 1 1 1 0 0 0 1
```

Как можно видеть, ни один из городов в таблице не имеет букв “ё”, “ы”, “э”, “ю” в названии.

Еще об индексации

Выше мы рассмотрели способ нахождения индексов различных элементов массива в APL. Для той же цели в этом языке существует отдельная операция — бинарная (диадическая) форма операции ⍳.

```
Vec ← 15 40 63 18 27 40 33 29 40 88
⍝ вектор для поиска
Vec ⍳ 29 63 40 33 50
⍝ элементы, чей индекс нужно найти
8 3 2 7 11
```

[9] Первые буквы названий городов намеренно сделаны строчными, чтобы не увеличивать размер массива гласных.

Эта операция имеет две особенности:

• значение 40 встречается в исходном векторе трижды, однако в результирующий вектор включается только индекс его первого вхождения;

• значение 50 вообще не встречается в исходном векторе, поэтому его "индексом" является число на единицу большее, чем максимально возможный индекс.

Приведем реальный пример использования такой операции. Предположим, что фирма — производитель бытовой техники хочет предоставить покупателям скидки на некоторые позиции в своем каталоге. При этом размер скидки будет отличаться для каждой области, в которой есть магазины этой фирмы.

Регион	Скидка
17	9,00%
50	8,00%
59	6,00%
84	5,00%
89	4,00%
Остальные	2,00%

Нам необходимо выяснить, на какую скидку может рассчитывать покупатель, живущий в регионе D, к примеру, `D ←84`. Введем сначала две переменные:

```
      AREA  ← 17 50 59 84 89
      DISCT ←  9  8  6  5  4  2
```

Для начала проверим, входит ли D в список регионов со специальной скидкой:

```
      AREA⍳D
4
```

и убедимся, что это действительно так, D является четвертым элементом этого массива. Теперь высчитаем скидку для этого региона:

```
      DISCT[4]
5
```

То же самое мы получим, записав:

```
      DISCT[AREA⍳D]
5
```

Если покупатель живет в регионе с номером 75, 45 или 93, то результатом первой операции будет 6, а `DISCT[6]` вернет 2, как нам и хотелось бы.

Важная особенность всего вышесказанного состоит в том, что все операции являются векторными. Предположим, что мы хотим узнать скидку не для одного человека, а сразу для нескольких: в этом случае D будет вектором. Для этого нам не понадобится ничего менять в нашей программе:

```
      D←24 75 89 60 92 50 51 50 84 66 17 89
      DISCT[AREA⍳D]
2 2 4 2 2 8 2 8 5 2 9 4
```

Мы смогли получить этот результат, не написав ни единой подпрограммы и не использовав ни одной конструкции ветвления или организации цикла.

Системы отсчета

На самом деле мы только что перешли из одной системы отсчета в другую. Не пугайтесь, если эта формулировка кажется излишне сложной, она удобна для обобщенного разговора об операциях в APL. В данном случае мы просто отобразили вектор номеров регионов (исходный вектор) на вектор величин скидок (результирующий вектор).

Представим теперь, что исходным вектором у нас является латинский алфавит, состоящий из прописных и строчных букв, а также символа пробела. При этом результирующим вектором будет такой же алфавит, но включающий уже два одинаковых набора прописных букв и символ "*" для замены незнакомых литер:

```
      Almin
  a b c d e f g h i j k l m n o p q r s t u v w x y z
ABCDEFGHIJKLMNOPQRSTUVWXYZ
      Almaj
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
ABCDEFGHIJKLMNOPQRSTUVWXYZ*
      Fable ← 'Le petit Chaperon-Rouge a bouffé
le Loup'
```

Следующее выражение поменяет регистр строки Fable:

```
      Almaj[Almin ⍳ Fable]
LE PETIT CHAPERON*ROUGE A BOUFF* LE LOUP
```

При этом символ "é", не встречающийся в исходном алфавите, был заменен звездочкой.

Массивы и их структура

Ни один из традиционных языков программирования не позволяет нам выполнять осмысленные операции над массивами как над единым целым. Важной особенностью APL является возможность управления структурой, формой массивов. Немалое число программ может быть построено только при помощи операций, работающих с внутренней структурой.

Две первых операции, которые мы рассмотрим в данном разделе, — это Take (↑) и Drop (↓), позволяющие, соответственно, отобрать или отбросить часть данных из массива. В качестве примеров рассмотрим только работу с векторами, однако все то же самое применимо и к массивам более сложной формы.

Вспомним, что переменная Vec содержит значения 15 40 63 18 27 40 33 29 40 88.

```
      4 ↑ Vec      ⍝ отобрать 4 первых элемента
15 40 63 18
      5 ↓ Vec      ⍝ отбросить 5 первых элементов
40 33 29 40 88
```

Если левый аргумент этой операции — отрицательное число, отсчет элементов начинается с конца массива:

```
      ¯3 ↑ Vec ⍝ отобрать 3 последних элемента
29 40 88
      ¯7 ↓ Vec  ⍝ отбросить 7 последних элементов
15 40 63
```

Результат отбрасывания семи последних элементов равен результату выполнения операции 3↑Vec. Зачем же нам необходимы две разные функции?

Представим, что у нас есть бизнес, оборот которого рос в течение двенадцати лет. В переменной Tomr мы будем хранить значение оборота в миллионах рублей.

```
      Tomr
56 59 67 64 60 61 68 73 78 75 81 84
```

При этом мы хотим посчитать разницу оборота между каждым годом и каждым следующим годом. Посмотрим на вектор оборотов без первого года ("каждый следующий год") и без последнего ("каждый текущий год"):

```
      1↓Tomr
59 67 64 60 61 68 73 78 75 81 84
      ¯1↓Tomr
56 59 67 64 60 61 68 73 78 75 81
```

Все, что нам остается сделать, — это вычесть эти два списка друг из друга:

```
      (1↓Tomr)-(¯1↓Tomr)
3 8 ¯3 ¯4 1 7 5 5 ¯3 6 3
```

Если заменить вычитание делением и добавить очевидные операции расчета остатка и его масштабирования, мы получим значения темпов роста оборота:

```
      100 × ((1↓Tomr)÷(¯1↓Tomr)) - 1
  5.35  13.56  ¯4.48  ¯6.25  1.67  11.47  7.35  6.85
¯3.85 8 3.70
```

Таким образом, выбор конкретной функции зависит от семантики задачи и удобства.

Еще одним интересным семейством операций в APL являются операции отражений и поворотов. Имея начальную переменную Towns, определенную как:

```
      Towns
москва
петербург
рязань
ярославль
иркутск
```

и применив операцию отражения (“⌽”, “CircleStile”), мы получим:

```
      ⌽Towns
   авксом
грубретеп
   ьназяр
ьлвалсоря
  кстукри
```

Применяя операцию переворачивания (“⊖”, “CircleBar”), получим:

```
      ⊖Towns
иркутск
ярославль
рязань
петербург
москва
```

А применив операцию транспонирования (“⍉”, “CircleSlope”), получим:

```
      ⍉Towns
мпряи
оеярр
стзок
кеасу
врнлт
абьас
 у вк
 р л
 г ь
```

Диадические варианты этих операций также весьма интересны. Диада “⌽” вращает массив по последнему измерению на заданное число позиций:

```
      2⌽Towns
сква   мо
тербургпе
зань   ря
ославльяр
кутск  ир
```

Диада “⊖” вращает массив по первому измерению на заданное число позиций:

```
      2⊖Towns
рязань
ярославль
иркутск
москва
петербург
```

Наконец, диада “⍉” выполняет гораздо более сложный вариант операции транспонирования, согласно заданному вектору перестановок. Для простоты рассмотрим лишь случай получения главной диагонали двумерного массива (матрицы):

```
      1 1 ⍉ Towns
мезст
```

Примеров, когда диадический вариант операции в APL выполняет более сложную версию монадической операции, довольно много.

Возвращение в начальную школу

Вспомним первый-второй классы и таблицу умножения чисел от единицы до девяти:

×	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Такие таблицы обычно являются результатом сложной операции, носящей название “внешнего произведения”[10]. Суть этой операции состоит в том, чтобы попарно брать элементы двух векторов[11] и поочередно применять к ним некоторую функцию. Попробуем сгенерировать какую-нибудь таблицу умножения с помощью APL.

```
      5 4 10 3 ∘.× 8 5 15 9 11 40
40 25  75 45  55 200
32 20  60 36  44 160
80 50 150 90 110 400
24 15  45 27  33 120
```

Попробуем теперь заменить операцию умножения какой-нибудь другой функцией:

```
(⍳5)∘.=(⍳5)
1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1
```

или

```
      (⍳5)∘.<(⍳5)
0 1 1 1 1
0 0 1 1 1
0 0 0 1 1
0 0 0 0 1
0 0 0 0 0
```

или же

```
      (⍳5)∘.≥(⍳5)
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
```

Перейдем к небольшому практическому примеру. Предположим, что вектор Ages содержит информацию о возрасте респондентов некоторого опроса. Нам необходимо узнать, какое количество респондентов содержится в следующих возрастных категориях: 0 – 25 – 30 – 35 – 45 – 50 – 55 – 65 и старше. Тех людей, чей возраст находится на границе, будем относить к категории более молодых.

[10] Термин тензорной алгебры.

[11] В общем случае внешнее произведение применимо к любым двум массивам. Если число размерностей первого массива равно Q, а второго — R, то число размерностей результирующего массива будет равно Q+R.

```
      Ages ← 32 19 50 33 23 65 46 26 31 58 51 23
51 36 28 42
      Category ← 0 25 30 35 45 50 55 65
```

Подсчитаем внешнее произведение относительно операции “<”:

```
      Category ∘.< Ages
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1
1 0 1 1 0 1 1 0 1 1 1 0 1 1 0 1
0 0 1 0 0 1 1 0 0 1 1 0 1 1 0 1
0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0
0 0 0 0 0 1 0 0 0 1 1 0 1 0 0 0
0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Мы получили битовые маски респондентов, имеющих больший возраст, чем граница каждой категории в нашем векторе категорий. Просуммируем эти маски, чтобы получить количества:

```
      c ← +/Category∘.<Ages
      c
16 13 11 8 6 4 2 0
```

Результат нужно читать так: у нас есть 16 людей, которые “старше нуля лет”, 13 среди них старше 25 лет, а среди них 11 старше 30 лет, и т.д. Чтобы подсчитать количество респондентов в каждой категории, отбросим первое число в полученном результате, к концу получившегося вектора добавим 0 (с помощью операции конкатенации “,”) и получившийся результат вычтем из c:

```
      c-1↓c,0
3 2 3 2 2 2 0
```

Опять же, мы получили результат, не пользуясь циклами, подпрограммами, а наша программа будет работать для любого числа респондентов и категорий.

В APL есть также особая нотация для записи внутреннего произведения, но его подробное рассмотрение выходит за рамки нашей статьи.

Структурирование данных

Как вы уже могли заметить, в APL есть свой подход к разделению и структурированию данных. Представим, что у нас есть база данных из 10 000 записей, хранящаяся в файле. В базе есть как строковые, так и числовые поля, например, имя, фамилия, возраст, год рождения. В обычных языках эта база была бы представлена в виде 10 000 структур разнородных данных, в APL ее бы представили в виде четырех массивов из 10 000 однородных элементов. Как мы видели ранее, этот подход позволяет работать внутренней “магии” APL.

Да будет жизнь!

Теперь мы рассмотрим процесс создания одной из самых известных учебных программ на APL — программы, реализующей игру “Жизнь”. В этой игре есть квадратное (потенциально бесконечное) поле с клетками, у каждой из которых есть, соответственно, восемь соседей. Каждая клетка может быть либо “живой”, либо “мертвой”. На каждом шаге происходит как гибель клеток, так и рождение новых по следующим правилам:

- если у мертвой клетки ровно три живых соседа, то она становится живой;
- если у живой клетки два или три живых соседа, то она продолжает жить, в противном же случае она умирает.

Классическая функция, реализующая “Жизнь” на APL, выглядит так:

```
life←{⊃1 ⍵ ∨.∧ 3 4 = +/ +/ 1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂⍵}
```

Разберем этот пример подробно[12]. Предположим, что игровое поле хранится в переменной R и представляет собой матрицу, в которой живым клеткам соответствуют единицы, а мертвым — нули. Пусть начальное поле будет таким:

```
      R
0 0 0 0 0 0 0
0 0 0 1 1 0 0
0 0 1 1 0 0 0
0 0 0 1 0 0 0
0 0 0 0 0 0 0
```

Для начала нам надо посчитать живых соседей каждой клетки. Для этого мы сдвинем поле на единицу в каждую сторону:

```
      1 0 ¯1 ⌽¨ ⊂R
```

0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

Здесь мы столкнулись с несколькими новыми операциями. Монада “⊂” (“LeftShoe”) позволяет представить массив любой формы в виде единичного блока данных (в выводе интерпретатора такие блоки представлены в виде очерченного прямоугольника). Оператор “¨” применяет функцию, переданную ему к каждому элементу левого аргумента и правому аргументу: в данном случае мы производим по очереди три “поворота” и объединяем их результаты в единый массив. Однако для полноты картины нам не хватает еще шести матриц, в которых элементы были бы сдвинуты по вертикали на разную величину. Для их создания мы воспользуемся внешним произведением относительно функции “⊖”:

```
      1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂R
```

Просуммируем теперь полученный результат два раза подряд, чтобы вычислить число живых соседей у каждой клетки:

```
+/ +/ 1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂R
```

Теперь найдем такие клетки, у которых число соседей равно трем или четырем (для живых):

```
3 4 = +/ +/ 1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂R
```

Теперь добавим также условие того, что в последнем случае это должны быть соседи живой клетки.

Теперь объединим эти матрицы в одну, заменив конъюнкцию внутренним произведением из конъюнкции и дизъюнкции (мы как бы сложим эти матрицы после их вычисления), и вытащим итоговую матрицу из блочного представления с помощью операции “⊃”:

```
⊃1 R ∨.∧ 3 4 = +/ +/ 1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂R
```

Мы получили процедуру, реализующую алгоритм игры “Жизнь”. Оформим ее в виде однострочной функции (мы не рассматривали это раньше). Имя переменной R при этом надо заменить на “⍵”, представляющую правый аргумент функции. Само тело функции запишется в фигурных скобках:

```
life←{⊃1 ⍵ ∨.∧ 3 4 = +/ +/ 1 0 ¯1 ∘.⊖ 1 0 ¯1 ⌽¨ ⊂⍵}
```

На этом мы завершим рассмотрение APL, оставив за бортом ввод-вывод, загрузку библиотек и другие подсобные служебные функции.

[12] Это будет непросто, но вполне понятно.

J и K

Логичным развитием APL является язык J. Сам по себе он достаточно большой, и мы не будем здесь в него углубляться. Большая часть привычных операций из APL в нем присутствует с точно таким же или чуть измененным поведением, а сами операции записываются с помощью одиночных или двойных ASCII-символов. В J используется лингвистическая терминология для различения семантико-синтаксических конструкций. Каждый единичный кусочек кода на J может быть:

- "существительным", то есть литералом, данными, записанными в явном виде;
- "местоимением", то есть именем переменной, содержащей данные;
- "глаголом", то есть функцией, монадической либо же диадической;
- "отглагольным местоимением", то есть именем переменной, содержащей функцию;
- "наречием", то есть "оператором" в терминах APL, монадической функцией, аргументом которой является другая функция;
- "союзом", то есть диадической функцией, обоими аргументами которой являются другие функции;
- "герундием", то есть "отглагольным существительным", в J эта структура представлена двумя союзами и служит для организации ветвлений;
- "частицей", то есть вспомогательным оператором, таким, как начало комментария ("NB.");
- "пунктуацией", то есть символом, определяющим порядок выполнения кода; в качестве самого яркого примера можно привести круглые скобки.

Язык K привнес дополнительные средства из функционального программирования и программирования, ориентированного на данные, например, лямбда-выражения и словари. Рассмотрим, например, задачу Пола Грэма о создании аккумулятивного генератора. Задача состоит в том, что нужно написать функцию, принимающую аргумент `n` и возвращающую другую функцию, от аргумента `m` с результатом `n+m`. На K решение задачи можно записать так (в K присваивание обозначается простым двоеточием):

```
   accgen: {[n] {[m] n + m}}
{[n] {[m] n + m}}
   accgen[3]
{[m] n + m}
   accgen[3][4]
7
```

Выражение в фигурных скобках здесь означает анонимную функцию, в квадратных скобках записываются аргументы.

Взглянем теперь краем глаза на работу со словарями. Помните, что я говорил чуть ранее о каноническом представлении баз данных в APL? В K этот подход реализован на уровне языка. Создадим таблицу из двух столбцов пятнадцати строк, с названиями "a" и "b", заполнив эту таблицу случайными целыми числами от 0 до 10.

```
   t: .+(`a`b; 2 15 _draw 10)
.((`a
   4 2 5 3 8 9 6 4 7 1 0 6 5 9 2
   )
  (`b
   8 2 2 7 7 0 4 2 5 7 3 5 5 1 6
   ))
```

Функция `_draw` создает массив случайных чисел от 0 до значения своего правого аргумента (10) с формой, определенной левым аргументом (2 15). Обратная кавычка "`" служит для представления строкового литерала. Запись двух массивов (`a`b и сгенерированного массива случайных чисел) в скобках через точку с запятой создает двухэлементный массив, который затем поэлементно складывается, чтобы сформировать массив массивов. Функция "." в этом контексте служит для создания словаря из массива массивов. Можно отсортировать словарь по любому из ключей-столбцов, например, по `b`:

```
   <t`b
5 13 1 2 7 10 6 8 11 12 14 3 4 9 0
```

Приведенное выражение возвращает список индексов, отсортированных столбцов. Здесь выражение "`t`b`" возвращает столбец `b`, а функция `<` сортирует его и возвращает список индексов.

Другие векторные языки

Векторные языки достаточно многообразны, и описание их всех не поместится в рамках нашей статьи. Многие из них уже не поддерживаются или еще находятся в слишком экспериментальной фазе своего развития. Тем не менее, зная, как программировать на оригинальном APL, вы легко сможете перейти к разработке на любом из этих языков.

Будущее векторных языков

Дейкстра сравнивал APL с идеально ограненным бриллиантом: он красивый, манящий, затягивающий, сложный и простой одновременно, но если вы попытаетесь приделать к нему что-нибудь свое, то вы просто-напросто все сломаете. Отчасти в этом высказывании есть своя доля правды. APL — это такой язык, во внешнем виде которого все, и правда, кажется идеальным, таким, будто добавить уже ничего нельзя. Однако изменения могут быть сделаны во внутренностях интерпретаторов. В данное время ведутся обширные исследования по переносу APL на векторные и графические процессоры, на технологии CUDA и OpenCL. Также, поскольку квантовая механика имеет дело с большим количеством огромных матриц, APL может стать чуть ли не единственным языком современности, который почти без изменений подойдет к квантовым ЭВМ, если они будут когда-либо реализованы.

Литература и ссылки

1. *http: // www.dyalog.com / MasteringDyalogAPL / MasteringDyalogAPL.pdf* — официальное руководство по Dyalog APL, наиболее успешной реализации APL.
2. *http://tryapl.org/* — онлайн-интерпретатор APL от Dyalog.
3. *https://github.com/ngn/apl* — свободная реализация APL на Javascript.
4. *https://www.gnu.org/software/apl/* — GNU APL.
5. *http://www.jsoftware.com* — официальный сайт J.
6. *http://www.jsoftware.com/jwiki/Books* — книги о J.
7. *https://github.com/kevinlawler/kona* — Kona, свободная реализация K.

Создаем приложение для соцсети

1. Основные сведения

И.Б. Государев,
РГПУ им. А.И. Герцена, доцент кафедры информационных и коммуникационных технологий, кандидат педагогических наук, доцент кафедры инновационных образовательных технологий СПбАППО, Санкт-Петербург

Социальная сеть предоставляет пользователю множество разнообразных функций и возможностей. Владелец аккаунта может загружать фотографии и видеоролики, создавать сообщества, лайкать и комментировать чужие записи. Но бывают ситуации, когда хочется большего. Например, узнать, кто из друзей чаще всего заходил на твой профиль, или поиграть с друзьями в игру. Если какая-либо программа не может решить ту или иную задачу, для такой программы создают надстройку, плагин. Пользователь загружает ее и добавляет к программе. Например, значительной долей своей популярности браузер Mozilla Firefox обязан именно огромному количеству плагинов. Можно привести и более общий пример. Любая операционная система обеспечивает базовый набор услуг по управлению компьютером, но чтобы сделать что-то специфическое, нужно загрузить приложение (`application`). Сегодня пользователи таких мобильных операционных систем, как Android или iOS, загружают приложения из специальных хранилищ (Google Play, App Store). Написание приложений может быть развлечением начинающего программиста, а может стать источником серьезного дохода для профессионального разработчика. Для конкретного приложения Windows, MacOS, Linux, Android, iOS операционная система является платформой. Платформа предоставляет приложению готовые средства для решения типовых задач. Например, чтобы создать на экране окно, которое управляется мышью, не нужно писать собственную длинную и сложную программу. Достаточно вызвать функцию, которая уже определена в так называемом API — ***A**pplication **P**rogramming **I**nterface*. API представляет собой набор функций, констант, объектов или классов, которые можно использовать в своей программе. Мы уже обсуждали некоторые вопросы, связанные с API, в статье [1], посвященной трехмерному вебу, в соответствующем номере “Информатики”.

Социальная сеть — это тоже своего рода платформа. Она устроена весьма сложно. Часть ее работает в браузере — на стороне клиента, а часть — на стороне сервера.

Когда мы набираем в адресной строке браузера *vk.com*, браузер посылает запрос на сервер с IP-адресом 87.240.131.118, и работающая на сервере программа отвечает на запрос, посылая обратно команды на языках HTML, CSS и Javascript. Команды попадают в браузер, который выполняет их, результатом чего является появление на экране полей для ввода логина и пароля, некоторых изображений, гиперссылок и прочего.

Какой бы ни была социальная сеть (ВКонтакте, Facebook, Одноклассники и т.д.), ее создатели предусмотрели API для разработчиков приложений. Мы предпримем попытку познакомиться с API сети ВКонтакте и создать с его помощью небольшое приложение.

Но сначала — несколько необходимых технических пояснений.

Перейдите по адресу *http://dist-learn.spb.ru/html?5* и введите в верхнем поле редактора следующий код (нажав на кнопку **Play** и поставив галочку напротив обновления):

```
<!DOCTYPE html>
<html>
 <head>
  <title>Встраивание фрейма</title>
  <meta charset="utf-8">
 </head>
 <body>
    <h1>Встраиваемый документ</h1>
    <iframe src="iframe.html?ilia"
     width = "600" height ="150"></iframe>
 </body>
</html>
```

Вы видите, что внутри одной страницы находится другая. Тег `iframe` позволяет внедрять одну страницу в другую, и именно этот принцип внедрения используется в большинстве случаев, чтобы позволять пользователям играть или выполнять другие специальные действия, не покидая своего аккаунта. Если в верхней части окна поменять содержимое тега `h1`, то это значение будет меняться также в содержимом встроенного окна. Как это достигается? Если открыть код файла `iframe.html`, то мы увидим команды на языке Javascript:

```
var who = window.location.search.substr(1);
var head = parent.document.querySelector("h1").
innerHTML;
document.write("В окне " + who + " заголовок "
+ head);
```

Первая команда получает данные о том, как был выполнен вызов данной встроенной страницы (знак вопроса означает начало передачи данных методом `GET`). Вторая обращается к содержимому заголовка родительского окна. Третья выводит их в тело данной страницы.

Так выглядит, очень упрощенно, общий принцип обмена информацией между встраиваемым приложением и социальной сетью через API. Встраиваемое приложение обращается к родителю (parent) и получает из него необходимые для работы данные. Соответствующая функциональность ВКонтакте находится в файле *http://vk.com/js/api/xd_connection.js*, который необходимо подключить к своему встраиваемому приложению.

Вообще ВКонтакте поддерживает три типа приложений:

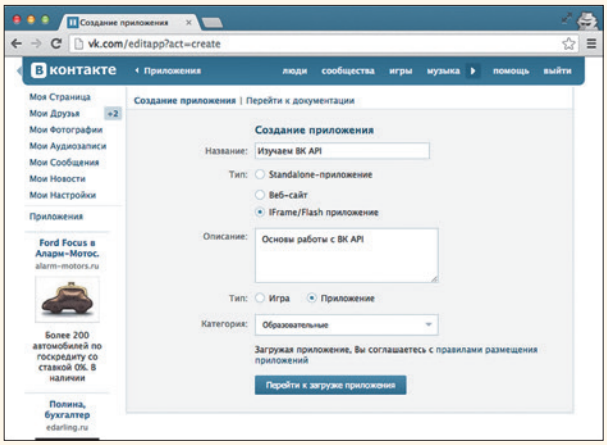
1. Файл *.swf* (флэш, флекс) или HTML-страница в IFrame;
2. Приложения, устанавливаемые на компьютер или мобильное устройство (например, клиент ВКонтакте на мобильном телефоне);
3. Приложения, встраиваемые в другие сайты для авторизации через ВКонтакте и использования данных аккаунта на другом сайте.

Создание приложений второго и третьего типов — весьма нетривиальная задача, требующая освоения таких технологий, как OAuth, а создание приложений на основе флэш требует, соответственно, специальных знаний в области ActionScript [3]. Поэтому мы рассмотрим создание HTML-страницы.

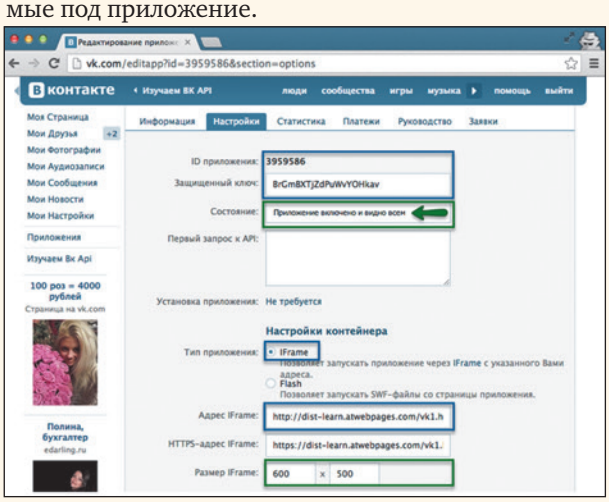
Приложение в виде HTML-страницы должно храниться на отдельном сайте, следовательно, нам понадобится веб-хостинг. Автор данной статьи обычно использует коммерческий хостинг, но для экспериментов с приложением, вероятно, можно воспользоваться одним из бесплатных хостингов. Примером такого хостинга является awardspace (*http://awardspace.info/free-web-hosting.html*).

Итак, создадим сайт на бесплатном хостинге. В нашем случае он будет иметь адрес *http://dist-learn.atwebpages.com/*.

На этом сайте мы создадим страницу, которая будет встраиваться внутрь ВКонтакте. Перейдя по ссылкам “Приложения” — “Мои приложения” — “Разработчикам” (или просто *http://vk.com/dev*), мы увидим кнопку “Создать приложение”, нажатие которой открывает первый этап разработки:



Приложение получает в системе идентификатор и защищенный ключ. Его необходимо “включить” и указать адрес местонахождения HTML-страницы. В нашем случае это *http://dist-learn.atwebpages.com/vk1.html*. Там же жестко задаются размеры, отводимые под приложение.



После этого приложение готово к использованию всяким, кому известен его адрес вида *http://vk.com/app3959586*.

Чтобы больше пользователей могло получить доступ к приложению, можно отправить запрос на его включение в каталог. Заявка рассматривается администрацией, и, в случае положительного решения, приложение можно будет найти в общем списке всех утвержденных приложений.

2. Специальные сведения

Итак, приложение в IFrame существует в виде веб-страницы, написанной на языке HTML, при этом к ней подключаются дополнительные ресурсы: каскадные стили, сценарии из сторонних библиотек, медиа. И, конечно, так как страница располагается на собственном сервере, она может обращаться к сценариям на нем (например, на языке PHP). Вполне может быть так, что приложение использует социальную сеть только как источник данных о пользователях, а все значимые действия осуществляются на своем сервере. Например, на сервере может находиться хранилище/магазин учебных фильмов или книг, пользователям предлагают те или иные продукты в результате анализа содержимого их постов на стене. Такие приложения представляют собой полноценные многоуровневые информационные системы, разработка которых связана с использованием широкого спектра технологий и языков. Мы ограничимся рассмотрением клиентских сценариев.

Итак, крайне популярным действием в соцсети является “лайканье”, и множество лайков под тем или иным постом является частой целью пользователя. Было бы интересно проследить у себя на стене статистику лайков — например, в наглядной форме представить, сколько лайков собрали последние *N* постов.

Любые действия, осуществляемые с данными пользователей ВК, реализуются через API (*http://vk.com/dev/main*). Соответствующий файл подключается в начале страницы:

```
<script      src="http://vk.com/js/api/xd_
connection.js?2" type="text/javascript">
</script>
```

Функции API вызываются в стандартной форме вида

```
VK.api("метод", {параметр:значение,…,},
       function(data) {
//обработка данных, пришедших в ответ
}
```

Возвращаемые данные имеют формат JSON (то есть объект, или ассоциативный массив):

```
{параметр1: значение1, параметр2:
 значение2, …}
```

Например, данные о конкретном посте на стене выглядят так:

```
{
      "id":3139,
      "from_id":513324,
      "to_id":513324,
      "date":1382902795,
      "post_type":"post",
      "text":"Пример сообщения на стене",
      "can_edit":1,
      "can_delete":1,
      "comments":{
            "count":5
            },
      "likes":{
             "count":22
             },
      "reposts":{
             "count":1
             }
}
```

Обращение к таким объектам в Javascript осуществляется через точку: *mypost.likes.count*.

Для получения информации о заданном количестве постов используется метод `wall.get`, ему передается количество постов, отступ номера от самого последнего (например, если постов всего 10 и отступ равен 4, то будут возвращены посты 6, 5, 4, 3, 2, 1).

Соответственно, обращение к методу может быть следующим:

```
VK.api("wall.get",
 {count:100, filter: "owner", offset:0},
 function(data) {
     var posts = data.response;
     var postsLen = posts.length – 1;
     var likes = 0;
     for (i = 1; i <= postsLen; i++) {
          likes += posts[i].likes.count;
    }
 });
```

где 100 — количество загружаемых постов, 0 — отступ.

Функция, которая указывается в вызове метода, — так называемая “функция обратного вызова” (`callback`). Вызов наступает тогда, когда API платформы обработает запрос и вернет нужные данные. Все необходимые действия по их обработке содержатся в теле этой безымянной функции.

В частности, в `data.response` помещается массив, в котором все элементы, кроме самого первого (нулевого, считая от нуля), представляют собой JSON-записи, описывающие посты. В результате работы приведенного выше цикла формируется более простой массив `likesarray`, который состоит из количеств лайков к каждому посту. А в переменную `likes` помещается сумма всех лайков.

Пример примитивного приложения, вычисляющего сумму лайков за 100 последних постов текущего пользователя, находится по адресу *http://vk.com/app3962349*.

```
<!DOCTYPE html>
<html>
 <head><script src="//vk.com/js/api/xd_connection.js?2"  type="text/javascript"></script>
  <title>Пример простого приложения</title>
  <meta charset="utf-8">
 </head>
 <body>
    <h1>Пример простого приложения</h1><h2></h2>
    <script type="text/javascript">
    VK.init(function() {
        VK.api("wall.get", {count:"100",filter:"owner", offset:"0"}, function(data) {
            var posts = data.response;
            var postsLen = posts.length - 1;
            var likes = 0;
            for (i=1; i<= postsLen; i++) {
                likes += posts[i].likes.count;
            }
            document.querySelector("h2").innerHTML="Лайков поставлено: " + likes;
        });
    });
  </script>
 </body>
</html>
```

Чтобы сделать отображение информации более наглядным, по массиву лайков можно построить диаграмму. Это можно реализовать, например, так: выбрать максимальное количество (*M*) и принять его за 100. Умножив все элементы массива на дробь (100/*M*), мы получим тогда после округления массив значений от 1 до 100. А затем достаточно в цикле создать для каждого элемента массива цветной прямоугольник соответствующей ширины (или высоты). Получившиеся столбики займут пропорциональное место относительно длины элемента-контейнера.

Отметим, что ВК определяет еще ряд методов, связанных не с получением социальной информации, а с установкой параметров самого приложения (*http://vk.com/dev/clientapi*), — они вызываются несколько иначе:

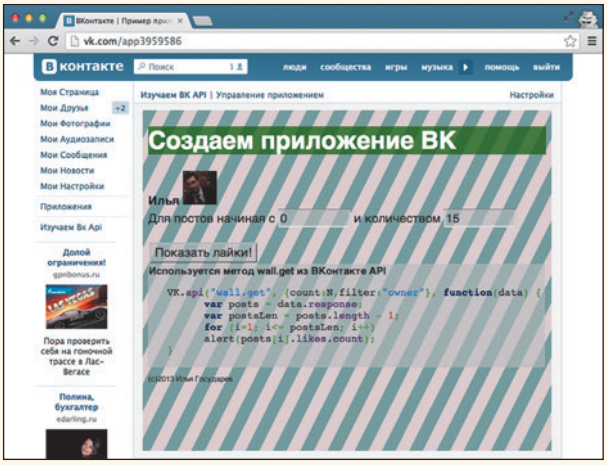
```
VK.callMethod("setTitle",
              "Пример приложения");
```

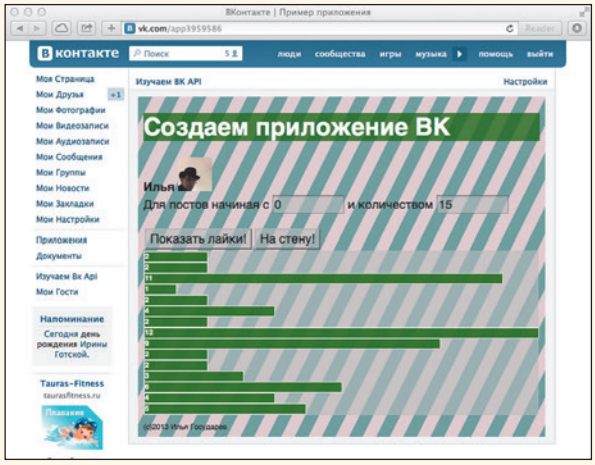
Очень часто приложения отображают какую-либо информацию на стене пользователя. Это может быть “текущий” пользователь (открывший приложение в своем аккаунте), пользователь, выбранный из списка друзей текущего пользователя, или просто произвольный пользователь. Разумеется, все зависит от настроек конкретной стены (разрешено ли постить на нее кому-либо, кроме владельца аккаунта).

Для отправки сообщения на стену используется метод `wall.post`, который принимает идентификатор пользователя и сообщение; если идентификатор не указан, то сообщение отправляется на стену текущего пользователя. Соответствующий код может выглядеть примерно так:

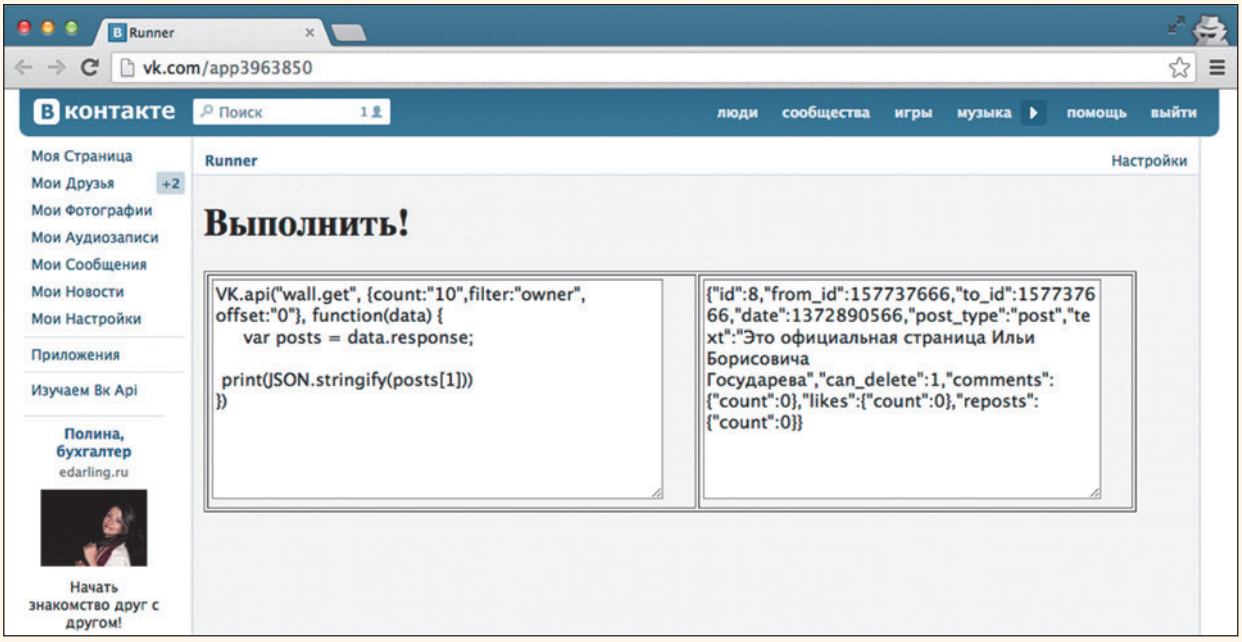
```
msg = "У меня есть " + likes + " лайков за " + np + " постов!";
          VK.api('wall.post',{message:msg},function(data) {
                if (data.response) {
                } else {  alert('Отменено');
                }
            });
```

Рассмотренные выше возможности реализованы в чуть более интересном приложении: *http://vk.com/app3959586* (см. рисунок слева).





После выбора количества постов и отступа от конца по нажатию кнопки “Показать лайки” строится диаграмма для количества постов < 20 (см. рисунок справа), и выводится суммарное количество лайков в против-



ном случае. Кнопка “На стену” позволяет добавить на свою стену вычисленное количество лайков.

Исходный код можно увидеть, щелкнув внутри фрейма правой кнопкой мыши и выбрав команду просмотра исходного кода фрейма.

В нем, например, используется получение имени и аватарки текущего пользователя методом `users.get`.

```
VK.loadParams(document.location.href);
//получить данные, переданные фрейму
var viewer_id = VK.params.viewer_id;
VK.api("users.get",
{"uids": viewer_id,"fields":"photo, first_
name"}, function(data) {
//имя data.response[0].first_name …
//аватарка
<img src="'+data.response[0].photo+'">';
});
```

Чтобы получить список друзей текущего пользователя, используется метод `friends.get`.

Если необходимо получить список друзей, включающий их имена и аватарки, то этого можно добиться с помощью аналогичного обращения:

```
VK.api("friends.get", {fields:"first_
name,photo"}, function(data) {
…
}
```

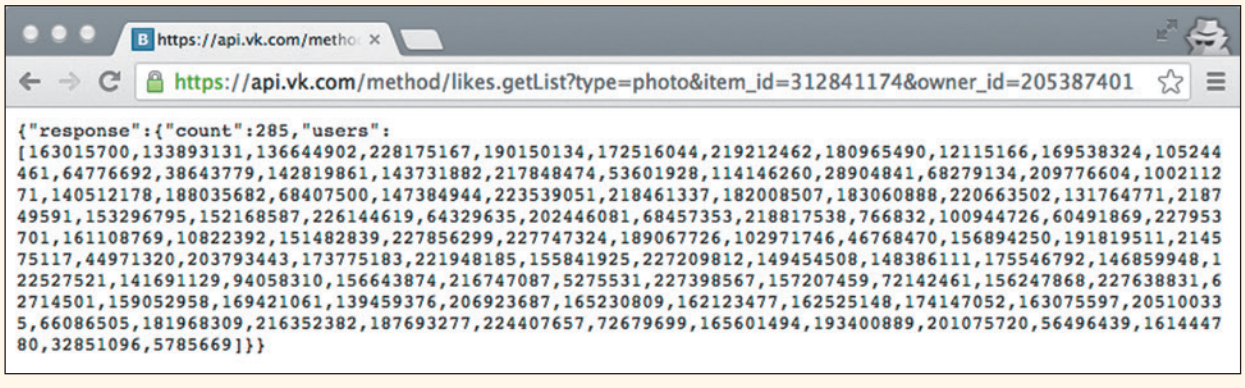
Внутри этой функции обратного вызова в `data.response` будет массив JSON-записей о друзьях (точно так же, как в случае с постами в примере выше). Далее, например, `data.response[3].photo` будет содержать ссылку на аватарку четвертого друга в полученном списке.

По адресу *http://vk.com/app3963850* автор разместил небольшое приложение, позволяющее запускать несложные API-запросы и просматривать результаты в том же окне (см. рисунок вверху страницы).

Воспользуйтесь этим приложением, чтобы выяснить, сколько пользователей лайкнули фотографию Тома Круза (его идентификатор пользователя равен 205 387 401, а идентификатор фотографии, например, 312 841 174), для этого воспользуйтесь методом `likes.getList`.

Некоторые обращения к API можно осуществлять непосредственно в адресной строке браузера (запрос GET). Например, чтобы решить поставленную выше задачу таким способом, достаточно написать в строке адреса *https://api.vk.com/method/likes.getList?type=photo&item_id=312841174&owner_id=205387401*.

Это будет работать, даже если вход в систему не осуществлен (см. рисунок внизу страницы).



Операции с данными, такими, как списки постов, друзей, фотографий и видео, производятся, таким образом, по стандартной схеме. Гораздо сложнее и интереснее дело обстоит, если нужно разместить медиаобъект из внешнего источника. Одна из самых распространенных задач — закачать фотографию с диска пользователя и разместить ее, к примеру, на стене. Эта задача решается в несколько этапов.

1. Получение адреса сервера, который осуществляет прием фотографии, с помощью метода `getWallUploadServer`; например, *cs614820.vk.com* или *cs410116.vk.com*.

Ответ приходит в виде

```
{"response":{"upload_url":"http://cs614820.
vk.com/upload.php?act=
do_add&mid=157737666&aid=-14&gid=0&hash=
7eed75373fbf6ce0d941698b62ee3066&rhash=
281ec72026acd8ad8f97eeae7cea500c&swfupload=
1&api=1&wallphoto=1",
"aid":-14,"mid":157737666}}
```

Чтобы этот метод заработал, необходимо отметить в настройках приложения, что требуются установка и доступ к фотографиям.

```
VK.api("photos.getWallUploadServer",  {},
function(data) {
// адрес содержится в
data.response.upload_url
})
```

2. Загрузка фотографии на этот сервер POST-запросом, в ответ на который приходят промежуточные данные `{server:614820, photo:…., hash:….}`, где `photo` представляет собой текстовое представление фотографии в формате `multipart/form-data`

```
{"server":410116,"photo":"[{\"photo\":\
"aca2736f01:x\",\"sizes\":[[\"s\",\
"410116852\",\"9426\",\"3GSpsXn-P6E\",
48,48],[\"m\",\"410116852\",\"9427\",  \"ME5NT
YSk0KM\",48,48],[\"x\",\"410116852\",\"9428\"
,\"RA2iEnMqDZk\",48,48],[\"o\",\"410116852\",
\"9429\",\"SVOeNhdIW4g\",48,48],[\"p\",\"410116
852\",\"942a\",\"iU2rlz_Eqyc\",48,48],[\"q\",\"
410116852\",\"942b\",\"Ciu0fJc-fMw\",48,48],[
\"r\",\"410116852\",\"942c\",\"g92ZwN9WH-Q\",
48,48]],\"kid\":\"8ee429baeb79236c574178b662
1b4c40\"}]","hash":"184fe6a805eaccb85864e0a8
ee791522"}
```

Поскольку приложение находится на нашем сервере, а отправка запроса производится на сервер ВК, то средствами Javascript запрос выполнить нельзя (запрет на кроссдоменные запросы), и для этого потребуется привлекать скрипт на серверном языке.

3. Передача серверу этих данных (`server`, `photo`, `hash`) с помощью метода `photos.saveWallPhoto` и получение идентификатора фотографии.

4. Размещение фотографии на стене методом `wall.post` с указанием этого идентификатора в качестве значения параметра `attachments`.

Рассмотрение серверного кода выходит за рамки данной статьи, но пример приложения, реализующего загрузку фотографий на стену в контакте, подробно рассмотрен в материале [2], где можно изучить исходный код на языке PHP.

Примером приложения, использующего все рассмотренные выше варианты взаимодействия с VK API наряду с многими другими, является каталог поздравительных открыток *http://vk.com/top_cards*, — это приложение позволяет подобрать открытку (или загрузить собственную) с возможным музыкальным приложением, отправить ее выбранным друзьям, оценить открытку, сохранить в свое собственное избранное.

Разработка такого многофункционального приложения на платформе соцсети — сложная многоуровневая задача, которая хорошо подходит на роль задания для выполнения курсовой или даже дипломной работы для студентов инженерных специальностей. Так, под руководством автора данной статьи студентом факультета информационных технологий РГПУ им. А.И. Герцена Денисом Халтуриным был выполнен проект “Видеохранилище”, доступный по адресу *http://vk.com/kinohranilishe*.

Соответственно, для самостоятельного выполнения в рамках данного материала можно предложить следующие несложные задания:

1. Получить список постов за конкретный указанный явно диапазон дат и вычислить количество лайков за него.

2. Определить частоту появления указанного слова в текстах постов.

3. Определить, сколько постов содержали вложения в виде фотографий.

4. Получить список всех друзей и отправить на стену выбранного друга какую-либо аудиозапись из числа аудиозаписей текущего пользователя.

5. Составить рейтинг своих фотографий из какого-то одного альбома, отсортировав их по убыванию лайков.

6. Получить список фотографий, не имеющих ни одного лайка, и переместить их в отдельный альбом “Плохие”.

7. Определить, с кем из друзей у вас больше всего общих друзей (*http://vk.com/dev/friends.getMutual*), и поделиться с этим другом самой популярной аудиозаписью на стене.

Литературные и интернет-источники

1. *Государев И.Б.* Трехмерный веб // Информатика, № 11, 2013.

2. Создание приложения для загрузки фотографий на стену [Электронный ресурс] *URL:http://habrahabr.ru/post/137334/* (дата обращения: 01.10.2013).

3. Принципы создания Flash-приложений (включая Flex, MXML, ActionScript, FlashVars) [Электронный ресурс] URL: *http://habrahabr.ru/post/71814/* (дата обращения: 01.10.2013).

“Азбука Роботландии”: работаем с буфером обмена, строим и читаем деревья

А.А. Дуванов,
г. Переславль-Залесский,
kurs@robotland.pereslavl.ru,

Н.Д. Шумилина,
г. Тверь,
nshumilina@yandex.ru

Вводное слово

Мы уже писали про “Азбуку” на страницах “Информатики” (см. ссылки в конце этой публикации). Причина этой заметки — желание поделиться новыми методическими идеями, которые родились в процессе доработки второй части курса “***Информация***”.

Напомним, что “**Азбука Роботландии**” — это **фундаментальный** курс азов информатики для младшей школы.

Что мы понимаем под фундаментальными азами? Прежде всего — ***способность мыслить алгоритмически***, ведь основа информатики — теория и практика составления алгоритмов для обработки информации. Алгоритмическое мышление (наряду с другими) непременно надо закладывать в младшем возрасте, оно полезно в любой деятельности, и особенно в той, которая не связана с компьютером. Ведь ***алгоритмическое мышление — это способность облечь абстрактную идею в последовательность конкретных шагов, необходимых для ее воплощения на практике***.

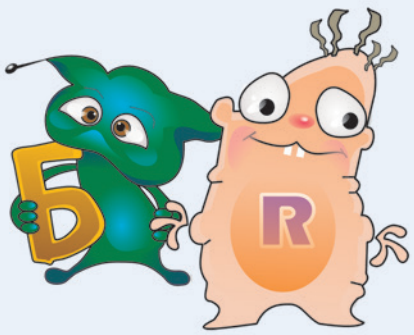
Курс охватывает четыре года:

- 1-й год (1-й класс). **Компьютер**
- 2-й год (2-й класс). **Информация**
- 3-й год (3-й класс). **Алгоритмы**
- 4-й год (4-й класс). **Интернет**

В настоящий момент готовы первые две части курса и идет работа над третьей.

Демо электронных учебников-лабораторий и методичек можно посмотреть на странице *http://robotlandia.ru/abc.htm*.

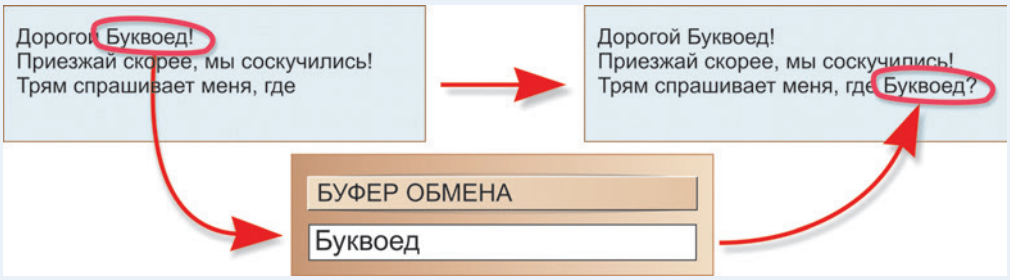
Курс можно заказать на странице *http://www.botik.ru/~robot/sale/index.htm*.



Этюд 1.
Буфер обмена: четыре алгоритма и одна схема

В этой заметке речь пойдет о ***буфере обмена*** операционной системы — удобном механизме копирования информационных элементов, как внутри одного приложения, так и между приложениями. Разговор о буфере возникает в теме “Многострочное редактирование”, основы которого дети изучают наряду с освоением информационных структур — списков, таблиц и иерархий.

Цитата из учебника: *“Буфер обмена — это специальная память для копирования информации. Если поместить в буфер обмена информацию, то потом ее можно будет многократно копировать”.*



Как создать копию?

Можно заново набрать копируемый текст (или заново нарисовать копируемую картинку).

Долго, неудобно и проблематично — можно в копии наделать ошибок.

Компьютер для того и создан, чтобы работу с информацией сделать удобной, быстрой и эффективной.

Буфер обмена — это специальная память, в которую можно помещать (***записывать***) копию объекта, не создавая его заново.

После записи информации в буфер ее можно многократно копировать (***читать*** из буфера), до тех пор, пока в буфер не будет записан новый объект.

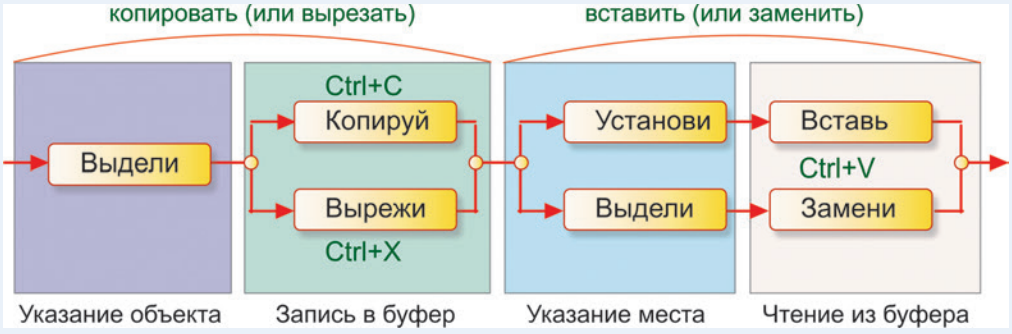
При записи информации в буфер оригинал в источнике может сохраняться (операция “***копирование в буфер***”) или удаляться (операция “***вырезание в буфер***”).

При чтении информации из буфера она может вставляться в приемник (операция “***вставка из буфера***”) или заменять в нем некоторый информационный объект (операция “***замена из буфера***”).

Таким образом, порядок работы пользователя с буфером обмена можно описать в виде вариативной цепочки действий:

копировать (или вырезать), вставить (или заменить)

Получаются четыре варианта, и они описываются схемой, которая приводится в разделе учебника “Четыре алгоритма и одна схема”. Рассмотрим детальнее каждую часть схемы.



Первое действие в этой схеме — ***выделение*** — предшествует записи в буфер, ибо необходимо указать, что записывать.

Хочешь записать в буфер информационный объект — сначала выдели его!

Запись в буфер выполняется двумя разными командами. Первая оставляет оригинал, вторая удаляет его.

Чтение из буфера выполняется одной командой, но может означать вставку или замену. Для замены фрагмента нужно предварительно его выделить. Если выделения нет, выполняется вставка на место курсора.

Уважаемые коллеги, вам ничего не напоминает приведенная выше схема? Ну, конечно, это типичный образец ***синтаксической диаграммы***! Один из способов формального описания грамматики языка программирования наряду с формулами Бэкуса — Наура.

Давайте посмотрим, как эта диаграмма появляется на страницах нашего учебника. Слово Кукараче!

Четыре алгоритма и одна схема

Кукарача, известный программист Роботландии, решил внести свою алгоритмическую лепту в рассказ РМ-1 (персонаж курса, исполнитель ***Р**едактор **М**ногострочный № **1***) о буфере обмена.

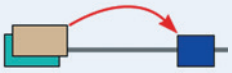
— Друзья! — сказал Кукарача. — Все возможные варианты работы с буфером обмена можно описать четырьмя алгоритмами.

Алгоритм 1. Выдели → копируй → установи → вставь

1. ***Выдели*** фрагмент
2. ***Копируй*** из текста в буфер (Ctrl + C)
3. ***Установи*** курсор на место вставки
4. ***Вставь*** из буфера (Ctrl + V)

Алгоритм 2. Выдели → копируй → выдели → замени

1. ***Выдели*** фрагмент
2. ***Копируй*** из текста в буфер (Ctrl + C)
3. ***Выдели*** другой фрагмент
4. ***Замени*** из буфера (Ctrl + V)



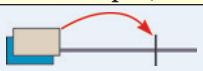
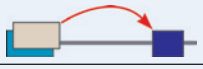
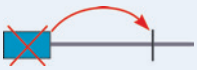
Алгоритм 3. Выдели → вырежи → установи → вставь

1. ***Выдели*** фрагмент
2. ***Вырежи*** из текста в буфер (Ctrl + X)
3. ***Установи*** курсор на место вставки
4. ***Вставь*** из буфера (Ctrl + V)

Алгоритм 4. Выдели → вырежи → выдели → замени

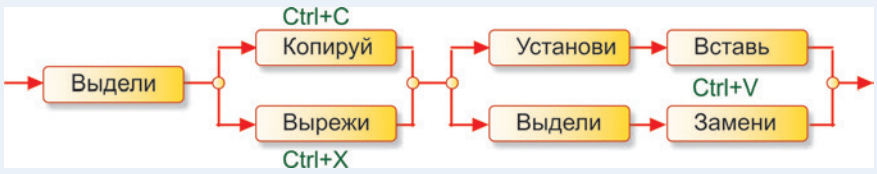
1. ***Выдели*** фрагмент
2. ***Вырежи*** из текста в буфер (Ctrl + X)
3. ***Выдели*** другой фрагмент
4. ***Замени*** из буфера (Ctrl + V)

Запишем кратко эти алгоритмы в таблицу:

№	Формула	Иллюстрация
1	Выдели → копируй → установи → вставь	
2	Выдели → копируй → выдели → замени	
3	Выдели → вырежи → установи → вставь	
4	Выдели → вырежи → выдели → замени	

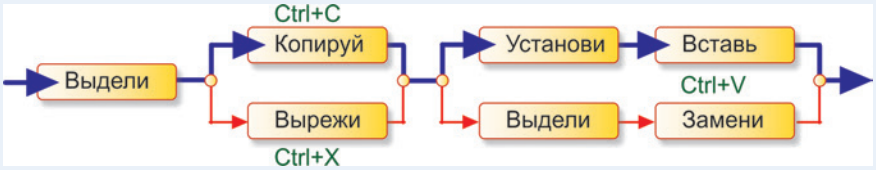
Эту таблицу можно использовать как справку — она показывает возможные варианты работы с буфером обмена.

Еще короче варианты работы с буфером обмена можно изобразить в виде такой схемы:



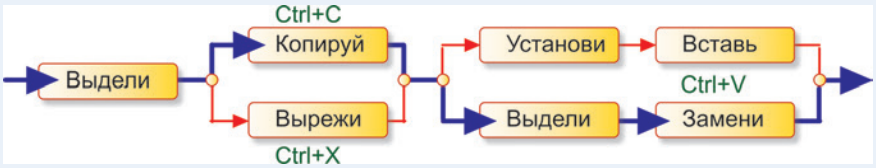
— А как пользоваться этой схемой?

— Очень просто! Нужно прокладывать на схеме пути по направлению стрелок. Каждый такой путь (а их всего 4) будет соответствовать одному из четырех алгоритмов работы с буфером. Вот сейчас я нарисую один из таких путей:



Этот путь соответствует алгоритму 1.

Вот еще один путь:



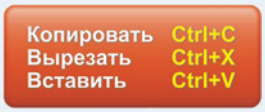
Какому алгоритму он соответствует?

— Этот путь соответствует алгоритму 2!

Работаем клавиатурой

Когда пользователь набирает текст, он, как правило, использует клавиатуру, значит, редактирующие действия, выполненные на клавишах, значительно повышают скорость работы за компьютером по сравнению с другими интерфейсными механизмами, связанными с мышью (контекстное меню, меню программы, панель инструментов).

Это замечание относится и к интерфейсу работы с буфером обмена. Клавиатурные аккорды полезно запомнить:



Можно дать детям мнемонические правила для запоминания букв аккордов: C — совпадает с первой буквой слова Copy (копирование), X — вторая буква слова eXtract (извлечение), кроме того, изображение X похоже на зачеркивание (вырезание) исходного текста. Происхождение обозначения V неизвестно, возможно, буква взята из-за близости клавиш X-C-V, но изображение V похоже на указание места вставки (уголок вниз). Кроме того, V совпадает с русским Вставить (хотя сомнительно, что авторы аккордов имели это в виду).

Свойства буфера обмена

Как уже отмечалось, буфер обмена предоставляется операционной системой, поэтому позволяет через него копировать и перемещать данные не только внутри одного приложения, но и между приложениями.

Буфер обмена — универсальная память. Она позволяет хранить информацию разного типа (текст, рисунки, видео, файлы, папки и пр.) и даже смесь разнотипной информации (например, текст с картинками).

Буфер обмена располагается в ОЗУ, поэтому его содержимое теряется при выключении компьютера.

Буфер обмена имеет только одну ячейку памяти. Новый фрагмент, сохраненный в буфере обмена, затирает старый.

Дополнительный буфер обмена

Пакеты программ и отдельные приложения могут иметь собственные буферы обмена в дополнение к общему буферу обмена операционной системы.

Так, пакет программ MS Office 2007 (и старше) имеет собственный буфер обмена, в котором не одна, а 24 ячейки памяти. С общим буфером обмена связана та ячейка, которая содержит последний сохраненный фрагмент.

С буфером обмена MS Office можно работать точно так же, как и с общим буфером операционной системы, но при желании внутри пакета можно использовать и дополнительную возможность: работать не с одним, а несколькими (до 24) сохраненными фрагментами информации.

Пример алгоритмического задания

Это задание дети выполняют не за компьютером, ученики собирают решение-алгоритм из заранее приготовленных бумажных карточек.

Задание. Задан текст (слайд на экране проектора в классе). Построить алгоритм для вставки слова “Буквоед” перед знаком вопроса в третьей строке и вместо слова “Кукарача”.

Для составления шагов алгоритма используем команды на карточках, располагая их в правильном порядке. Набор карточек (сообщается, что в наборе есть лишние карточки):

- Выделить Буквоед
- Выделить Кукарача
- Выделить ?
- Ctrl + C (две карточки)
- Ctrl + X (две карточки)
- Ctrl + V (две карточки)
- Установить курсор перед ?
- Установить курсор перед Кукарача

Практикум (п)

Дорогой Буквоед!
Приезжай скорее, мы соскучились!
Трям спрашивает меня, где ?
И Прям спрашивает, где Кукарача?

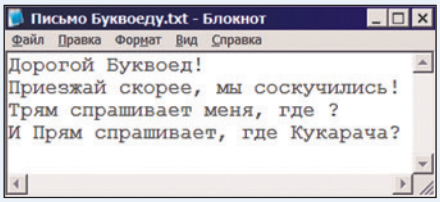
Решение

1. Выделить Буквоед
2. Ctrl + C
3. Установить курсор перед ?
4. Ctrl + V
5. Выделить Кукарача
6. Ctrl + V

Тестирование алгоритма

На компьютере учителя открываем в редакторе Блокнот текст Письмо Буквоеду.txt и выполняем построенный алгоритм следующим образом.

Шесть учеников вытягивают наугад карточки из стопки перемешанных карточек с шагами алгоритма (6 пунктов, 6 карточек) и выполняют на компьютере учителя свои действия.



Этюд 2.
Строим и читаем деревья

Тема “Иерархия” заставила нас задуматься: как учить детей строить и читать деревья? Были сформулированы два способа:

- ***“построение (чтение) по уровням”*** и
- ***“построение (чтение) по ветвям”***.

Но сначала небольшое…

Теоретическое введение

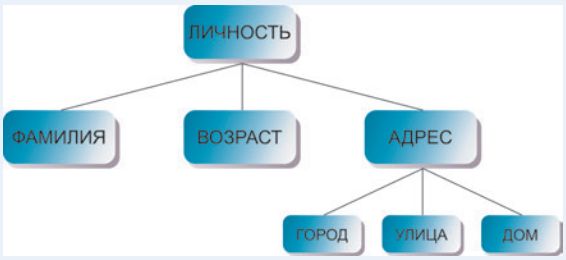
Определение дерева

Дерево (древовидная структура, иерархия) — это такая организация информационных элементов, которая отражает отношения иерархии (или ветвления), показывающие, как правило, порядок ***подчинения*** или ***вхождения*** элементов.

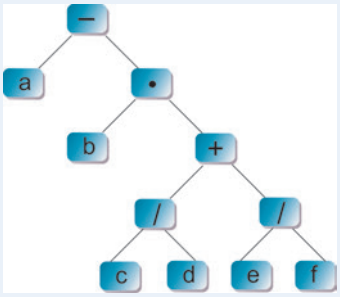
Дерево можно записать в виде именованного списка. Имя такого списка называется ***корнем***, а элементы — ***потомками***.

Каждый потомок, в свою очередь, может снова быть деревом. Если это не так, то он называется ***листом***.

Информационная структура “дерево” популярна в информатике, так как отражает естественную логику взаимосвязей объектов в окружающем мире. Например, для описания отдельных личностей в некоторых случаях вполне может подойти такая структура:



А вот пример записи выражения a – b • (c / d + e / f) в виде дерева:



Строгое определение дерева — рекурсивное. Вот как, например, определяется дерево в книге Д.Кнута “Искусство программирования для ЭВМ” (т. 1 “Основные алгоритмы”):

Дерево — это конечное множество T, состоящее из одного или более узлов, таких, что:

a) Имеется специально обозначенный узел, называемый корнем данного дерева.

b) Остальные узлы (исключая корень) содержатся в $m \geq 0$ попарно непересекающихся множествах $T_1, \ldots, T_m$, каждое из которых, в свою очередь, является деревом. Деревья $T_1, \ldots, T_m$ называются поддеревьями данного корня.

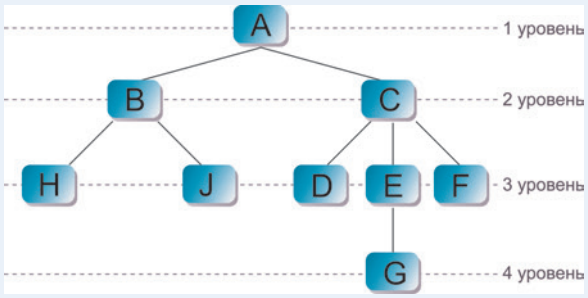
Это определение явно исключает ситуацию, при которой узел может иметь более одного родителя.

В самом деле, если бы узел имел двух родителей, множества узлов этих родителей имели бы в пересечении этот узел, что противоречит определению (“попарно непересекающиеся множества”).

Как изображают деревья

Схема

Графически дерево можно изобразить в виде схемы, состоящей из блоков (***вершин***, ***узлов***) и линий (***ветвей***, ***связей***):



Схему рисуют по уровням сверху вниз. На первом уровне располагается корень. На втором — прямые потомки корня. На третьем — прямые потомки вершин второго уровня и так далее.

Лесенка

Популярна запись дерева в виде ступенчатого списка (***лесенки***).

На первой ступеньке (первом уровне) записывают корень дерева.

На второй ступеньке (втором уровне) записывают прямые потомки корня.

На третьей ступеньке (третьем уровне) записывают прямые потомки вершин второго уровня.

И так далее.

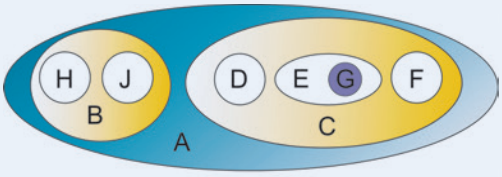
Список

Как уже отмечалось, дерево можно записать в виде именованного списка:

A(B(H,J), C(D,E(G),F))

Вложенные множества

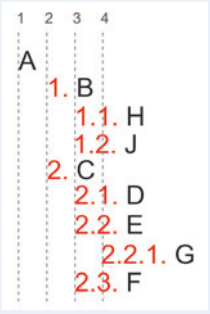
Дерево можно изобразить в виде системы вложенных множеств:



Заметим, что всякая пара множеств или не пересекается, или же одно из множеств пары содержит другое (пересечение потомков в дереве невозможно по определению).

Десятичная система Дьюи

Десятичная система Дьюи — это система классификации книг, разработанная в XIX веке американским библиотекарем Мелвилом Дьюи.



Принцип, положенный в основу системы Дьюи, хорошо подходит для отображения информационных деревьев. Такой метод часто называют ***десятичной системой обозначений Дьюи*** для деревьев.

Для наглядности дерево по системе Дьюи часто отображают как лесенку с уступами.

Доступ к вершинам дерева по системе Дьюи легко трансформируется в индексную запись:

$A_1 = B$
$A_2 = C$
$A_{1,1} = H$
$A_{1,2} = J$
$A_{2,2} = E$
$A_{2,2,1} = G$

И так далее.

Адресация информационных элементов

Информационный элемент

Информационный элемент — это выделенная по какому-либо признаку часть информации.

Информационные элементы упорядочивают в разные ***структуры***, подбирая подходящую по условию задачи, чтобы в дальнейшем облегчить поиск информации.

По общепринятому формализму, информационный элемент можно обозначать ***именем*** (буквой, фразой — не пустым набором символов).

Обозначим информационный элемент через X.

Мы можем рассматривать элемент **X** как единое целое, потому что X — ***информационный атом***, не допускающий деления на части, или потому что нам удобно оперировать с X как с единым целым.

Но если элементы (подэлементы) X упорядочены в структуру, возникает вопрос о доступе к элементам, составляющим X.

В списке адрес элемента — это его порядковый номер. В таблице — композиция из номера строки и номера столбца. В дереве — путь от корня до вершины, в которой находится элемент, или система индексации, в которой число индексов увеличивается от уровня к уровню.

Такого рода адресацию принято записывать следующим образом.

Список

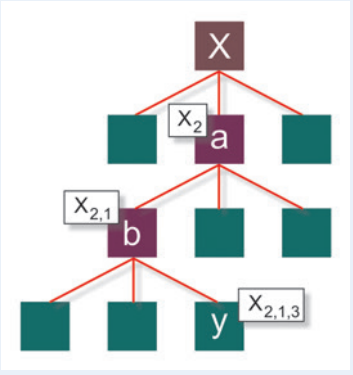
Если X — список, то его элементы обозначают формулой X_i, где i — порядковый номер элемента в списке (индекс).

Таблица

Если X — таблица, то ее элементы обозначаются как $X_{i,j}$, где i и j — соответственно номер строки и номер столбца (индексы).

Дерево

Пусть X — дерево. Корень обычно обозначают тем же именем, что и само дерево.



Путь от корня к вершине. Доступ к элементу, находящемуся в вершине y, указывают как путь от корня к вершине y, используя в качестве разделителя точку или слэш (прямой или обратный). Например:

- X.a.b.y
- X/a/b/y
- X\a\b\y

Индексная запись. Как и в случае со списками и таблицами, для адресации в иерархии можно использовать индексы. Только количество

индексов не является величиной постоянной, а зависит от уровня, на котором расположен информационный элемент.

Пусть y есть *3*-й потомок вершины b, которая есть *1*-й потомок вершины a, которая есть *2*-й потомок корня. Тогда доступ к y можно обозначить при помощи индексной записи следующим образом:

$$X_{2,1,3}$$

Как строить дерево?

Рассматриваем два способа, обозначая их как

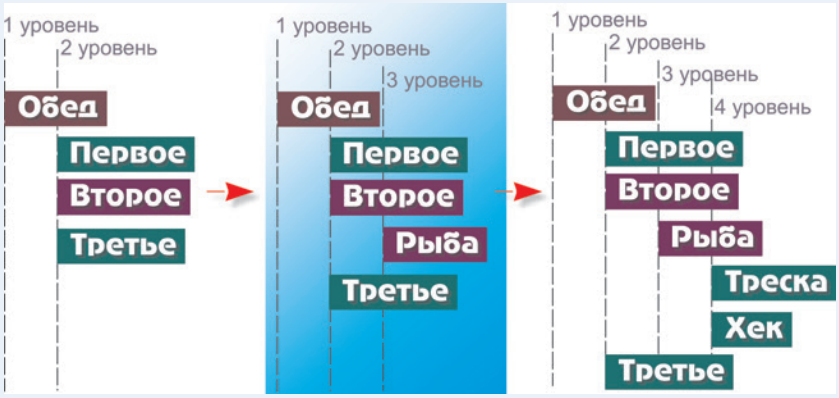
- ***построение по уровням***
- ***построение по ветвям***

Продемонстрируем эти способы на примере построения иерархии Обед:

Построение по уровням

Лесенка

Этот способ демонстрирует нижеприведенный рисунок:



Записываем корень на первом уровне, а на втором — его потомки.

Затем строим ***все*** вершины на третьем уровне, потом ***все*** вершины на четвертом уровне и так далее.

Заметим, что приходится раздвигать строки при переходе к построению следующего уровня. Это особенно неудобно, когда лесенка записывается вручную на бумаге.

Графическая схема

Порядок построения вершин "***по уровням***" показан на рисунке:

Построение по ветвям

Лесенка

В этом варианте строки не раздвигаются. Дерево записывают, продвигаясь по текущей ветви до листьев, выписывая всех потомков, а потом возвращаются и продолжают неоконченный уровень.

Это построение основано на рекурсивном определении дерева:

Дерево — это вершина (корень), с которой связан список непересекающихся деревьев (поддеревьев).

Из определения следует рекурсивный алгоритм построения дерева.

Построить дерево
  Записать корень
  Построить деревья, связанные с корнем

Алгоритм *Построить дерево* строит корень и вызывает сам себя для всех потомков корня.

Построим дерево Обед ***по ветвям***, исходя из приведенного выше рекурсивного алгоритма.

	Строим дерево Обед. Записываем корень на первом уровне и приступаем к построению поддеревьев. Первое поддерево содержит единственный корень-лист Первое. Строим второе поддерево. Записываем его корень Второе и приступаем к построению его поддеревьев.
	Строим первое поддерево Рыба. Записываем корень и приступаем к построению его поддеревьев.
	Первое поддерево (корень-лист) Треска. Второе поддерево (корень-лист) Хек. Дерево Рыба построено. Дерево Второе тоже построено, так как у этого корня нет больше поддеревьев. Продолжаем строить поддеревья корня Обед. Осталось построить только одно поддерево (корень-лист) Третье. Построение дерева Обед завершено.

Графическая схема

Порядок построения вершин “***по ветвям***” показан на рисунке:

Как читать дерево?

Следуя алгоритмам построения, будем читать дерево так, как строим, сверху вниз по уровням или ветвям.

Еще один важный вопрос: как различать при чтении вершину с “полным” набором потомков от вершины, у которой этот набор не полон?

О “полноте” построенного дерева

Посмотрите на эту схему:

Европа
  Северная Европа
  Южная Европа

Что отражено в этом дереве?

Вариант 1. Европу разделяют только на две части: северную и южную. Список потомков корня “полный”.

Вариант 2. Европу разделяют на четыре части, но восточная и западная части в схеме не отражены (например, сравнивается экономика стран, входящих только в Северную Европу и Южную Европу). Список потомков корня “не полный”.

Информация, организованная в виде дерева, представляет собой ***информационную модель*** иерархически организованного объекта.

С точки зрения формальной информатики не важно, является ли список потомков какой-либо вершины в этой модели полным или нет. Информатика работает с построенным деревом (хранит, передает, обрабатывает), и для нее не важна смысловая интерпретация, как для математики не важно, что суммируется, бараны или верблюды.

Другое дело, когда мы начинаем вникать в смысл построения (поднимаясь над формальным уровнем) и размышлять над тем, что у нас получается в итоге: стадо рогатых или караван горбатых?

На этом смысловом уровне хочется, прочитывая иерархию, находить такие выражения, по возможности лаконичные, которые различали бы два варианта интерпретации схем, подобных нашему географическому примеру (“полных” и “не полных” иерархий).

Приведенные ниже рекомендации не являются общепринятыми, но выглядят разумными с нашей точки зрения.

Полный вариант 1 предлагается читать так: Европа “***разделяется на***” Северную Европу и Южную Европу (только на эти две части, других частей нет).

Неполный вариант 2 предлагается читать так: “***в состав Европы входят***” Северная Европа и Южная (возможно, наряду с другими частями).

Таким образом, будем различать выражения “***в состав входят***” и “***разделяется на***”.

Будем говорить “***разделяется на***”, когда требуется подчеркнуть полноту построенных отношений, когда важно отметить, что, кроме обозначенных потомков, в состав вершины больше ничего не входит (“***состоит исключительно из***”).

Выражение “***в состав входят***” — будем считать общим. Оно применимо и к случаю “***разделяется на***”, и к варианту, когда, кроме обозначенных потомков, есть и другие, которые не показаны (“***в состав входят наряду с другими***”).

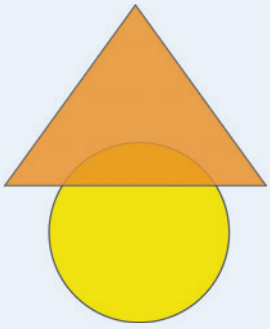
Разделяется на	**В состав входят**
	

Рисунок
  Треугольник
  Круг
Рисунок ***разделяется на*** треугольник и круг.
	Продукты
	  Мед
	  Молоко
	В состав продуктов ***входят*** мед и молоко.

Когда иерархия отражает отношения подчинения, будем говорить “***подчиняются***” в общем случае или “***подчиняются только***”, когда важно подчеркнуть полноту набора потомков данной вершины.

О чтении деревьев

Дерево строится сверху вниз. Таким же образом, сверху вниз, будем и читать его. При этом возможны два способа чтения, соответствующие двум способам построения, которые мы условно обозначили “*по уровням*” и “*по ветвям*”. Продемонстрируем эти способы на примере чтения дерева Обед.

Чтение по уровням

В состав обеда входят первые, вторые и третьи блюда.
Среди вторых блюд есть блюда из рыбы.
Среди рыбных блюд имеются блюда из трески и хека.

Чтение по ветвям

В состав обеда входят: первые блюда, вторые блюда (среди которых есть блюда из рыбы, среди которых есть блюда из трески и хека) и третьи блюда.

Второй способ сложнее, как по записи, так и по восприятию. Поэтому будем опираться в основном на чтение деревьев ***по уровням*** (хотя для некоторых иерархий предпочтительнее чтение ***по ветвям*** или комбинирование этих двух способов).

Правило чтения по уровням следует за правилом построения по уровням. Читаем корень на первом уровне. Затем читаем потомки корня на втором уровне. Читаем потомки детей корня на третьем уровне и так далее.

Правило чтения по ветвям следует за правилом построения по ветвям. Читаем корень на первом уровне. Затем читаем потомки корня слева направо как отдельные деревья.

При чтении (любым способом) используем обороты типа: “***в состав целого входят части***”, “***главному элементу подчиняются элементы***” или выражения типа “***целое состоит из частей***”, “***главному элементу подчиняются только элементы***”, если хотим подчеркнуть полноту построенного дерева.

Принимая на вооружение принцип чтения всей иерархии ***сверху вниз***, при разборе отношений “***вершина, ее прямые потомки***” будем допускать и вариант чтения ***снизу вверх***, произнося: “***вершины такие-то входят в состав вершины такой-то***”. Например, блюда из трески и хека входят в состав рыбных блюд; память и процессор входят в состав компьютера; Северная Европа и Южная Европа входят в состав Европы; Сотрудник 1, Сотрудник 2 и Сотрудник 3 подчиняются Начальнику цеха 2 и так далее.

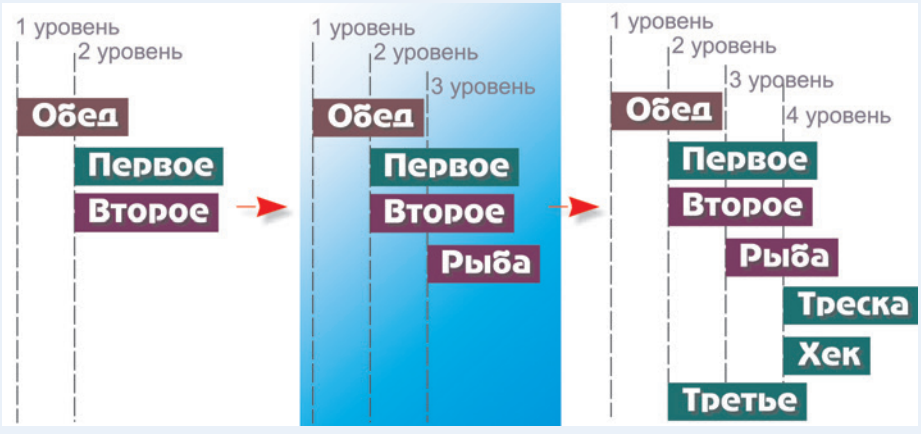
Как преподнести детям рекурсивный алгоритм?

Как инициировать рекурсивное построение и чтение иерархий?

Конечно, формальный разговор о рекурсии бесполезен для наших учеников. Но мы обнаружили (и это нас сильно порадовало), что если показать детям несколько примеров убедительных рекурсивных рассуждений, то рекурсия становится рабочим инструментом, прописываясь в детском сознании не на формальном, а на интуитивно-прикладном уровне.

Вот как в учебнике объясняется рекурсивный алгоритм “по ветвям”.

“Посмотрите на рисунок, который следует ниже. На первом уровне всегда только одна запись — корень (Обед). Двигаемся вниз по второму уровню. Вершина Первое — лист, продолжаем спуск на втором уровне. Вершина Второе — не лист, записываем ее потомки ниже на третьем уровне. Потомок один — Рыба, но это не лист, выписываем на четвертом уровне его потомки Треска и Хек. Возвращаемся на третий уровень, сестер у вершины Рыба нет, возвращаемся на второй уровень и записываем последнюю вершину Третье”.



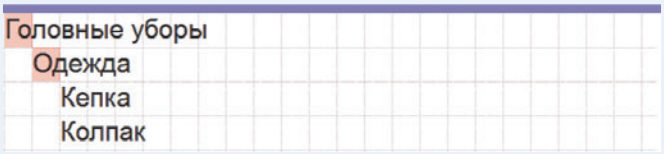
Иерархические построения дети осваивают на электронных страницах учебника-лаборатории в среде исполнителя “Строитель иерархий”. Ученики строят дерево в виде лесенки, перетаскивая вершины на свои места мышкой.

Часть заданий для “Строителя” мы специально сформулировали в манере “по ветвям”, чтобы подтолкнуть детей к такому же способу построения. К нашей радости, большинство детей на удивление легко выполняли рекурсивные действия, следуя приведенному в задании описанию.

Вот пример одного из таких упражнений.

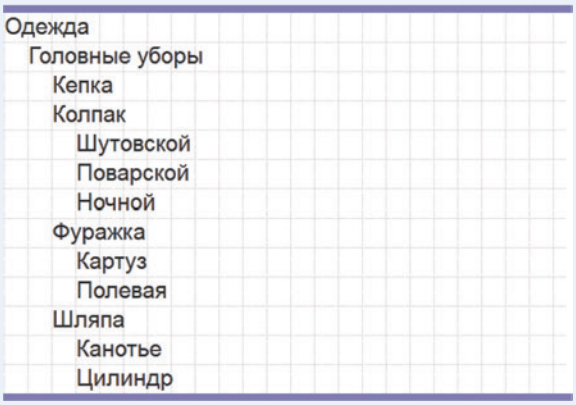
Задание. Головные уборы (урок 19, практикум 5)

Отметим небольшую провокацию в задании. Вершина Одежда описана так, что если не вникать в смысл, то можно подумать, что она является потомком вершины Головные уборы. И дети нередко попадают в эту ловушку:



Диагностические пометки исполнителя помогают быстро понять причину ошибки и внести нужную правку.

Решение показано на иллюстрации справа.



Дети, конечно, разные. И то, что одним дается легко, для других может стать проблемой, и эту проблему приходится преодолевать отдельными индивидуально подобранными усилиями.

Но есть дети, и это радует, которые сразу воспринимают рекурсивный способ построения дерева и берут его на вооружение. Для таких ребят работа со “Строителем иерархий” легка и радостна: они первыми выполняют базовые задания, они в творческом полете, они все поняли! По мере того, как другие дети постепенно вникают и схватывают суть, роботландские “профессора” с увлечением выполняют трудные дополнительные упражнения (которых в учебнике заготовлено изрядное количество по разнообразной тематике: конфеты, шоколад, напитки, пряности, самолеты, корабли, кошки и другие).

Список публикаций в “Информатике”

1. *Дуванов А.А., Шумилина Н.Д.* Азбука Роботландии. Информатика (Издательский дом “Первое сентября”) № 14, сентябрь, 2011.
2. *Дуванов А.А., Шумилина Н.Д.* Алгоритмы и программы “Азбуки Роботландии”. Информатика (Издательский дом “Первое сентября”) № 14, сентябрь, 2011.
3. *Дуванов А.А., Шумилина Н.Д.* Устройства ввода и вывода. Теория относительности. Информатика (Издательский дом “Первое сентября”) № 14, сентябрь, 2011.
4. *Дуванов А.А., Шумилина Н.Д.* Азбука Роботландии. Информация. Информатика (Издательский дом “Первое сентября”) № 7, август, 2012.
5. *Дуванов А.А., Шумилина Н.Д.* Алгоритмы редактирования текста. Информатика (Издательский дом “Первое сентября”) № 8, сентябрь, 2012.
6. *Дуванов А.А., Шумилина Н.Д.* Структуры данных. Информатика (Издательский дом “Первое сентября”) № 9, октябрь, 2012.
7. *Дуванов А.А., Шумилина Н.Д.* Азбука Роботландии: информатика для началки должна быть фундаментальной. Информатика (Издательский дом “Первое сентября”) № 4, апрель, 2013.

ИСТОРИЯ ИНФОРМАТИКИ

Логарифмическая линейка — предшественник калькулятора

Д.М. Златопольский,
Москва

Уверен, что многие читатели видели так называемую "логарифмическую линейку" — счетный прибор, применявшийся до появления калькуляторов и персональных компьютеров. Это было достаточно универсальное устройство, на котором можно было умножать и делить, возводить в квадрат и куб, вычислять квадратные и кубические корни, синусы, тангенсы и другие значения. Причем делалось это с достаточно большой точностью — до 3–4 знаков после запятой.

В данной статье мы расскажем об основных принципах работы прибора.

Если взять две обычные линейки, то с их помощью можно производить действия сложения и вычитания. В этом легко убедиться, решив примеры: 2 + 4 = 6 или 8 – 3 = 5.

В первом случае, совместив шкалы линеек, как показано на *рис.* 2, найдем ответ: 6. Во втором, рассматривая положение шкал на *рис.* 3, читаем ответ: 5.

Однако использовать линейки для сложения и вычитания нецелесообразно. Эти действия было проще и удобнее производить на счетах. Но описанный прием оказывается очень полезным для умножения, деления, а также выполнения многих алгебраических и тригонометрических действий, если воспользоваться не обычными линейками, имеющими равномерные шкалы, а линейками с так называемыми "логарифмическими шкалами".

Чтобы понять, что это такое, проведем горизонтальную прямую, на которой будем отмечать вертикальные штрихи, рядом с которыми — ставить метки (числа).

Рис. 4

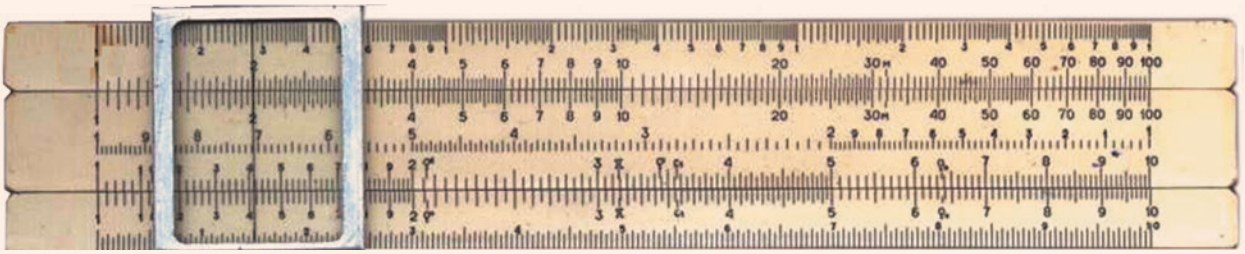


Рис. 1. Общий вид логарифмической линейки

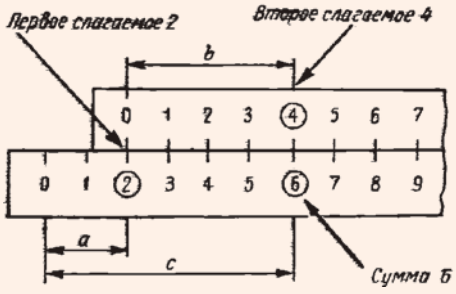


Рис. 2. Сложение при помощи двух обычных линеек

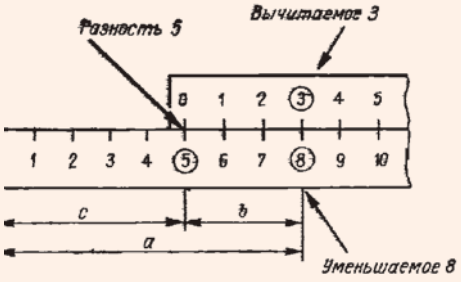


Рис. 3. Вычитание при помощи двух обычных линеек

Далее, для построения логарифмической шкалы воспользуемся следующим “уравнением логарифмической шкалы”:

$$\bar{a} = m \lg a,$$

где a — метка штриха, поставленного на расстоянии $\bar{a}$ см от начала шкалы;
$\lg a$ — десятичный логарифм числа a;
m — модуль шкалы, равный ее длине.

Стандартным считался размер шкалы счетной линейки длиной 25 см, поэтому дальнейшие расчеты проведем с этим значением.

Умножая модуль шкалы (25) на $\lg 1$, $\lg 2$, $\lg 3$, $\lg 4$ и т.д., получаем длины отрезков (в см), соответствующие штриху того или иного числа (см. табл. 1).

Таблица 1

Число a	**$\lg a$**	**Отрезок шкалы $\bar{a}$, соответствующий числу a**
1	0	0
2	0,3010	7,53
3	0,4771	11,93
4	0,6021	15,05
5	0,6990	17,47
6	0,7782	19,45
7	0,8451	21,13
8	0,9031	22,58
9	0,9542	23,86
10	1,0000	25,00

Нанеся штрихи согласно полученным длинам отрезков $\bar{a}$ на линейку и поставив рядом с ними соответствующие метки a, получим логарифмическую (неравномерную) шкалу (*рис.* 5).

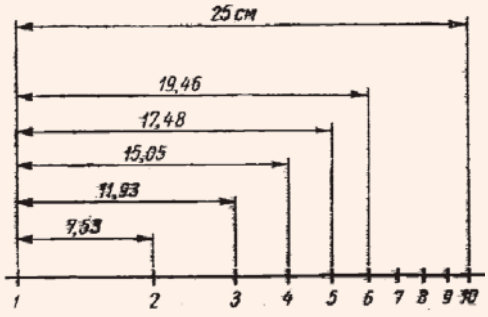


Рис. 5. Построение логарифмической шкалы длиной 25 см

Читатели, изучавшие логарифмы, знают, что

$$\lg (a \cdot b) = \lg a + \lg b,$$

$$\lg \left(\frac{a}{b}\right) = \lg a - \lg b,$$

$$\lg a^n = n \lg a,$$

$$\lg \sqrt[n]{a} = \lg a^{\frac{1}{n}} = \frac{\lg a}{n},$$

то есть логарифм произведения равен сумме логарифмов сомножителей, логарифм частного — разности логарифмов делимого и делителя, логарифм степени — логарифму основания, умноженному на показатель степени, и, наконец, логарифм корня — логарифму подкоренного числа, деленному на показатель корня.

Зная эти соотношения и воспользовавшись логарифмическими шкалами, можем свести действие умножения к сложению логарифмов, а действие деления — к вычитанию логарифма делителя из логарифма делимого и т.д.

Примеры

1) $a \cdot b = c$ при $a = 2$, $b = 3$.

Логарифмируя обе части равенства, имеем:

$\lg a + \lg b = \lg c$.

Взяв две линейки с логарифмическими шкалами и сделав установку, показанную на *рис.* 6, видим, что мы произвели сложение значений $\lg 2$ в $\lg 3$, получив в результате $\lg 6$, то есть произведение 2 на 3.

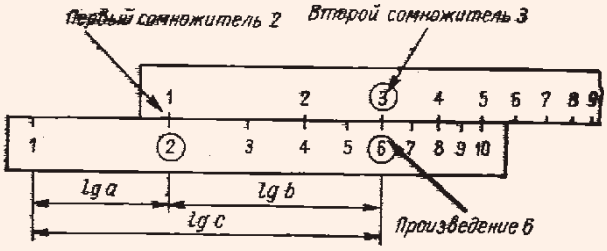


Рис. 6. Схема установки шкал при умножении

2) $\frac{a}{b} = c$ при $a = 8$, $b = 4$.

Логарифмируя обе части равенства, получим:

$$\lg a - \lg b = \lg c.$$

Соответствующая установка показана на *рис.* 7, из которого видно, что разность логарифмов делимого и делителя дает логарифм частного, в нашем случае — 2.

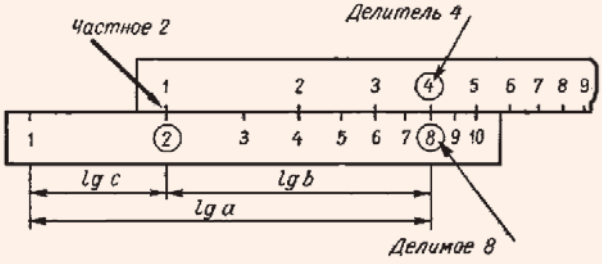


Рис. 7. Схема установки шкал при делении

Описанные особенности и лежали в основе работы логарифмической линейки. Она состояла из трех частей:

1) корпуса (*рис.* 8);

2) движка в средней части, свободно передвигающегося в пазах корпуса (*рис.* 9);

3) стеклянного бегунка, на котором нанесена линия, называемая “визирной линией”, или “волоском”, служащая для облегчения чтения цифр на шкалах (см. *рис.* 8).

Рис. 8. Корпус линейки

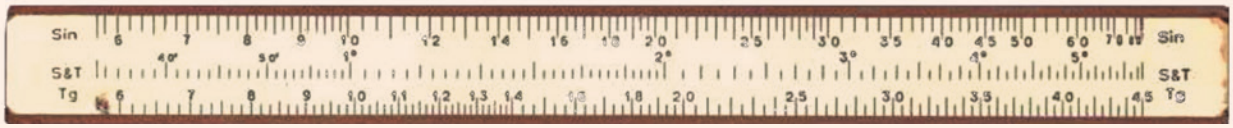


Рис. 9. Движок линейки (показан его оборот)

На корпусе и на движке нанесены различные логарифмические шкалы. Причем вертикальные штрихи на них имеются не только для целых значений меток, но и для десятых и других долей единицы.

Мы не будем здесь описывать все возможности линеек, а рассмотрим только примеры умножения и деления.

Умножение

Для того чтобы перемножить два числа, поступали следующим образом.

1. На основной шкале корпуса линейки (на *рис*. 8 она — вторая снизу) находился первый сомножитель, и на него устанавливали начало основной, нижней, шкалы движка (она на лицевой стороне последнего и точно такая, как основная шкала корпуса).

2. На основной шкале движка волосок бегунка устанавливался на втором сомножителе.

3. Ответ находился на основной шкале корпуса линейки под волоском.

Пример для умножения 2 на 3 будет аналогичным показанному на *рис*. 6 (второй множитель и произведение указывались волоском).

Если при этом волосок выходил за пределы шкалы, то на первом этапе на первый сомножитель устанавливали не начало, а конец движка (с числом 10). Пример для умножения 4,28 на 8 показан на *рис*. 10.

Деление

Для того чтобы разделить одно число на другое, делали так.

1. На основной шкале корпуса линейки находилось делимое, на которое устанавливался волосок бегунка.

2. Под волосок подводился делитель, найденный на основной шкале движка.

3. Результат определялся на основной шкале корпуса против начала или конца движка.

Пример для деления 8 на 4 иллюстрирует схема на *рис*. 7 (делимое и делитель указывались волоском).

Шкала квадратов чисел — вторая сверху (см. *рис.* 1), кубов — первая сверху. Волосок устанавливался на возводимом числе на основной шкале корпуса, и под волоском на соответствующей шкале считывался результат (при извлечении квадратного и кубического корней, наоборот, — результат находился на основной шкале).

В заключение — о “переносе” при расчетах запятой. Если, например, один из сомножителей равен 126, то на линейке использовалось значение 1,26, а найденное произведение увеличивалось в 100 раз. При возведении в куб числа 0,375 результат, найденный для числа 3,75, уменьшался в 1000 раз и т.п.

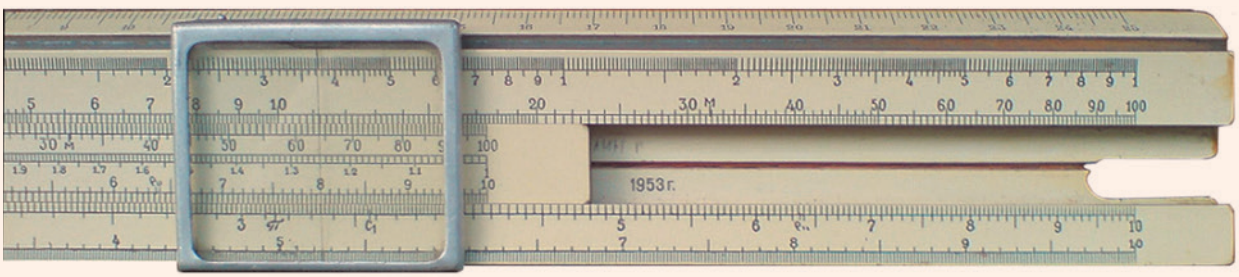


Рис. 10. Вариант умножения со смещением движка влево

Армянская алфавитная система счисления

Севак Карапетян,
ученик гимназии № 1530 г. Москвы

…лишь упорной работой легко достижима цель.
Анания Ширакаци

Будучи в гостях, я увидел часы, на которых вместо цифр использованы буквы армянского алфавита, и заинтересовался ими.

Оказывается, что в Древней Армении, как и в Греции, и на Руси, буквы алфавита служили также числовыми знаками [1–3]. Как известно, системы счисления, в которых для обозначения чисел использовались буквы и знаки некоторого алфавита, называются *алфавитными*.

В армянской алфавитной системе первые 9 букв служили обозначением единиц, следующие девять — десятков и так далее до тысяч. Все буквы и соответствующие числовые значения приведены в таблице:

	1	2	3	4	5	6	7	8	9
Единицы	Ա	Բ	Գ	Դ	Ե	Զ	Է	Ը	Թ
Десятки	Ժ	Ի	Լ	Խ	Ծ	Կ	Հ	Ձ	Ղ
Сотни	Ճ	Մ	Յ	Ն	Շ	Ո	Չ	Պ	Ջ
Тысячи	Ռ	Ս	Վ	Տ	Ր	Ց	Ո	Փ	Ք

Для того чтобы отличить буквы алфавита от числовых знаков, над последними проводили горизонтальную черточку или же в начале и конце числа проставляли точку или двоеточие. В многозначных числах сначала указывались бóльшие цифры-буквы. Например,

:Վ Խ Գ: = 3043
:Տ Շ Կ Է: = 4567

Десять тысяч армяне обозначали словом “бюр”, подобно русскому слову “тьма” [1–2] и греческому слову “мириада” [3]. Для обозначения этого числа применялся специальный знак бюра (^) рядом с количеством десятков тысяч, то есть 10 000 записывалось как Ա^. Для написания бóльших чисел пользовались теми же 36 буквами алфавита, приведенными в таблице. Например, число 100 000, или 10 бюров, выглядело так: Ժ^.

Еще примеры

:Ա^ՐՆՁԲ: = 15 482
:Ժ^ՅՁԼՋ: = 106 736

Также применялись простые дроби с единицей в числителе. Для обозначения таких дробей использовались буквы алфавита и дополнительный знак бюра, который ставился с левой стороны от буквы-цифры. Примеры:

^Դ = 1/4
^Ը = 1/8

А половина (½) имела специальный знак — ĉ.

Предлагаю читателям определить, какие “наши” числа записывались в виде:

1) :ՁԳ:
2) :ՅԿԲ:
3) :ՏՁԱ:
4) :^Է:
5) :Բ^ՐՁԸ:
6) :Հ^Է^ՈՃԽՋ:

Прим. ред. Ответы присылайте в редакцию.

Информация об армянской алфавитной системе счисления дошла до нас благодаря трудам великого армянского математика, географа, философа, историка и астронома VII века Анании Ширакаци (610-е годы — 685 г.), слова которого приведены в виде эпиграфа к статье.

Им был составлен армянский учебник арифметики. В этом учебнике, одном из самых древних из известных учебников, приведены таблицы сложения и вычитания, а также умножения.

Сложение и вычитание

Перед таблицами сложения стоит такое обращение, которое я несколько сократил: *“Цель моя, о, любители мудрости и желающие учиться у меня: представить творчество наших предков — искусство счисления, как живой голос доброго учителя. Учитесь на моих таблицах, хотя и изложил я их кратко, представив немного из многого”.*

Непосредственно за введением идут таблицы сложения. В виде примера приведу одну из них с их древними и современными обозначениями:

Ա Ա Բ	1 + 1 = 2
Ա Բ Գ	1 + 2 = 3
Ա Գ Դ	1 + 3 = 4
Ա Դ Ե	1 + 4 = 5
Ա Ե Զ	1 + 5 = 6
Ա Զ Է	1 + 6 = 7
Ա Է Ը	1 + 7 = 8
Ա Ը Թ	1 + 8 = 9
Ա Թ Ժ	1 + 9 = 10

Из приведенных таблиц видно, что в эпоху Ширакаци еще не существовало знаков сложения, вычитания (см. ниже) и равенства. Знаки “+” (плюс) и “–” (минус) вошли в употребление только в XV веке, а знак равенства (“=”) — и вовсе в XVI веке.

При сложении единственным правилом записи было — сумма должна была находиться в конце записи.

После таблиц сложения шли таблицы вычитания; на этот раз я покажу действия с десятками:

ժ ճ Ղ 100 – 10 = 90
ժ Ղ Ձ 90 – 10 = 80
ժ Ձ Հ 80 – 10 = 70
ժ Հ Կ 70 – 10 = 60
ժ Կ Ծ 60 – 10 = 50
ժ Ծ Խ 50 – 10 = 40
ժ Խ Լ 40 – 10 = 30
ժ Լ Ի 30 – 10 = 20
ժ Ի ժ 20 – 10 = 10

Пределы чисел, употреблявшихся в таблицах сложения и вычитания, доходят до 18 000.

Памятник Анании Ширакаци

Умножение

Таблицы умножения начинаются со следующего, частично сокращенного, предисловия: “*Странствующие в числах, продвигайтесь смело по пути, который я простираю перед вами подобно нежным и невысоким волнам, расступающимся перед быстроходным кораблем, уверенным в своей неуклонимости с правильного пути, ибо лишь упорной работой легко достижима цель*”.

Непосредственно за этим следуют таблицы умножения. Приведу некоторые из них:

Ա Ա Ա 1 × 1 = 1
Ա Բ Բ 1 × 2 = 2
Ա Գ Գ 1 × 3 = 3
Ա Դ Դ 1 × 4 = 4
Ա Ե Ե 1 × 5 = 5
Ա Զ Զ 1 × 6 = 6
Ա Է Է 1 × 7 = 7
Ա Ը Ը 1 × 8 = 8
Ա Թ Թ 1 × 9 = 9

ժ ժ ճ 10 × 10 = 100
ժ Ի Մ 10 × 20 = 200
ժ Լ Յ 10 × 30 = 300
ժ Խ Ն 10 × 40 = 400
ժ Ծ Շ 10 × 50 = 500
ժ Կ Ո 10 × 60 = 600
ժ Հ Չ 10 × 70 = 700
ժ Ձ Պ 10 × 80 = 800
ժ Ղ Ջ 10 × 90 = 900

Таблицы умножения состоят из 36 групп, по четыре таблицы в каждой. В трех таблицах заключены по 9 произведений, а в четвертой — 10. Таким образом, в каждой группе имеется 37 произведений, а всего — 1332.

Последним произведением в таблицах умножения Ширакаци является $9 \cdot 10^7$.

В заключение замечу, что хотя арабские цифры в Армении стали использовать еще в XVII веке, армянская алфавитная система счисления применялась вплоть до XX века, а в настоящее время ею пользуются подобно римским цифрам в русском языке, например: Գարեգին Բ. означает Гарегин II и Գ. գլուխ означает Глава III (в заголовке).

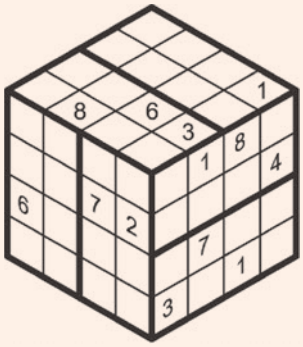
Литература

1. Буквенная цифирь. / Информатика, № 11/2009.
2. *Цайгер М.А.* Арифметика в Московском государстве XVI века. Беэр-Шева, 2010.
3. *Цайгер М.А.* Греческая алфавитная система счисления. / Информатика, № 6/2010.
4. *Абрамян А.Г., Петросян Г.Б.* Анания Ширакаци. Изд-во Ереванского университета. Ереван, 1970.

ЯПОНСКИЙ УГОЛОК

Судоку в формате 3D

Как показывает опыт, у читателей раздела “В мир информатики” большой популярностью пользуются японские головоломки “судоку”. Предлагаем решить, так сказать, объемный вариант этой головоломки:



Нужно расставить цифры от 1 до 8 так, чтобы в каждой строке, каждом столбце и каждой выделенной утолщенными линиями зоне цифры не повторялись. Обратите внимание, что строки и столбцы “перетекают” с одной грани куба на другую.

Ответы (можно не на все головоломки) присылайте в редакцию.

Судоку обычные

Решите также классические варианты судоку:

1)

8		2		7			9	
4				9				1
					3	5		
			8	4	7		2	
9	2						8	7
	8		9	2	6			
		1	7					
2				8				4
	5			6			9	3

2)

		5				9		8
				5			7	
	7	3	9		1			
3			7	4			2	1
		7	8				4	
		1				6		7
					9	5		
	4					7	8	
7		2						

ЗАДАЧНИК

Задача, которую вы решаете, может быть скромной, но если она бросает вызов вашей любознательности и заставляет вас быть изобретательным и если вы решите ее собственными силами, то вы сможете испытать ведущее к открытию напряжение ума и насладиться радостью победы.

Дьёрдь Пойа

Кто этот человек?

Н.Е. Кордина,
учитель информатики школы № 1,
г. Демидов, Смоленская обл.

Решив шесть приведенных ниже задач, по полученным результатам определите (используя Интернет или другие источники) фамилию и имя человека, о котором идет речь в заданиях. В ответе приведите также полученную при выполнении частных заданий информацию, год его рождения и ответьте на дополнительные вопросы:

1) кто из советских программистов повлиял на его судьбу;

2) какие книги написаны им и какая из них посвящена компьютеру, за которым герой работал в студенческие годы;

3) какие награды и премии получены этим человеком?

1. Дата рождения

Автомат получает на вход два трехзначных числа. По этим числам строится новое число по следующим правилам. Вычисляются три числа — сумма старших разрядов заданных чисел, сумма средних разрядов этих чисел и сумма младших разрядов. Полученные три числа записываются друг за другом в порядке убывания (без разделителей).

Пример

Исходные числа: 905 и 471; поразрядные суммы: 13, 7 и 6; результат 1376.

Определите, какое из следующих чисел не может быть результатом работы автомата:

1) 15121;
2) 12119;
3) 1001;
4) 1863.

Две первых цифры выбранного числа укажут день рождения искомого человека, две последние — месяц рождения.

2. Отец

Его отец преподавал в университете. Какой предмет он объяснял студентам, вы узнаете, выполнив следующее задание.

Система команд исполнителя **Робот**, “живущего” в прямоугольном лабиринте на клетчатой плоскости:

```
вверх
вниз
влево
вправо
```

При выполнении этих команд **Робот** перемещается на одну клетку, соответственно, вверх, вниз, влево и вправо.

Команда, записанная в теле цикла

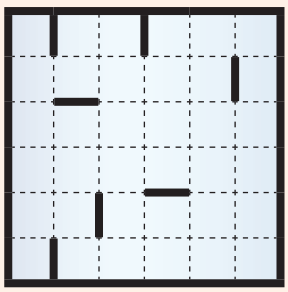
пока <условие> команда

выполняется, пока условие истинно.

В качестве условий используются четыре команды обратной связи, которые проверяют наличие или отсутствие стены с каждой стороны той клетки, где в данный момент находится **Робот**:

```
сверху свободно
снизу свободно
слева свободно
справа свободно
```

Сколько клеток лабиринта, изображенного на рисунке, соответствуют требованию, что, выполнив предложенную ниже программу, **Робот** окажется в той же клетке, с которой он начал движение?



Программа:

```
НАЧАЛО
пока <снизу свободно> вниз
пока <слева свободно> влево
пока <сверху свободно> вверх
пока <справа свободно> вправо
КОНЕЦ
```

Его отец преподавал (в зависимости от полученного результата):

— 2 — бухгалтерский учет;
— 3 — математику;
— 4 — программирование;
— 5 — геометрию.

3. Любимая игрушка героя в детстве

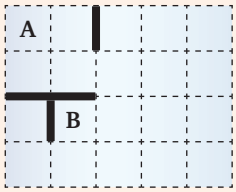
Система команд исполнителя **Робот**, “живущего” в прямоугольном лабиринте на клетчатой плоскости, такая же, как и в предыдущем задании. Командам присвоены условные номера:

1 — вверх
2 — вниз
3 — влево
4 — вправо

При выполнении каждой такой команды **Робот** перемещается в соседнюю клетку в указанном направлении. Если же в этом направлении между клетками имеется стена, то **Робот** разрушается.

Программы решения задач, которые исполняет **Робот**, записываются в виде последовательности номеров команд. Например, перемещение исполнителя по маршруту: вправо, вниз, влево, вверх — записывается как 3241.

Робот должен переместиться из клетки **A** в клетку **B**:



Какую последовательность из пяти команд должен выполнить **Робот** для этого, не разрушившись от встречи со стенами? Ответив на этот вопрос, вы узнаете любимую в детстве игрушку нашего героя.

Варианты (в зависимости от ответа):

1) 32323 — компьютер;
2) 23324 — калькулятор;
3) 31441 — кубик Рубика;
4) 22334 — танграм.

4. Музыкальный инструмент

Еще в школьные годы наш герой увлеченно занимался математикой, физикой и музыкой. Он также серьезно обучался игре на одном из музыкальных инструментов, сам написал несколько произведений и, вполне возможно, смог бы стать выдающимся музыкантом или композитором.

На каком инструменте он играл, вы узнаете, решив следующую задачу.

Система пронумерованных команд исполнителя **Вычислитель**:

1 — `допиши ноль справа`
2 — `вычти 2`

Первая команда дописывает к текущему числу ноль справа, а вторая вычитает из текущего числа 2.

Программы решения задач, которые исполняет **Вычислитель**, записываются в виде последовательности номеров команд. Например, программа для преобразования числа 5 в число 4980 имеет вид 1121, что соответствует следующим действиям:

1) `допиши ноль справа`;
2) `допиши ноль справа`;
3) `вычти 2`;
4) `допиши ноль справа`.

Какой должна быть программа для Вычислителя, преобразующая число 7 в число 6798? Команд должно быть не более пяти.

Варианты (в зависимости от ответа):

1) 21121 — гитара;
2) 12122 — скрипка;
3) 11221 — виолончель;
4) 12112 — пианино.

5. Первый компьютер

Во время каникул после первого курса Технологического института человек, о котором идет речь в задании, впервые занялся работой с компьютером. Некоторые из его преподавателей советовали ему держаться в стороне от компьютеров, утверждая, что это ни к чему хорошему не приведет. Но любопытство взяло верх, он проводил за компьютером ночи напролет. Позднее наш герой посвятит книгу по программированию именно этому компьютеру в память о многих приятных вечерах.

Чтобы выяснить, что это за компьютер, определите, сколько различных чисел может получить из числа 1 исполнитель **Вычислитель** с помощью программы, содержащей шесть команд.

Команды **Вычислителя**:

`Прибавь 4` — условный номер 1;
`Вычти 3` — условный номер 2.

Первая команда увеличивает текущее число на 4, вторая — уменьшает его на 3 (отрицательные числа допускаются).

Программы решения задач, которые исполняет **Вычислитель**, записываются в виде последовательности номеров команд.

Варианты (в зависимости от ответа):

1) 12 — “Стрела”;
2) 7 — IBM 650;
3) 8 — EDVAC;
4) иной ответ — ENIAC.

6. Разобраться до конца

Чтобы продолжить его высказывание “Лучший способ в чем-то разобраться до конца — это попробовать…”, выясните, какую фигуру нарисует исполнитель **Черепашка**.

Черепашка перемещается на экране компьютера, оставляя след в виде линии. Команды исполнителя:

— `Вперед n` (n — целое число) — **Черепашка** передвигается на n шагов в текущем направлении движения;

— `Направо m` ($0 \le m < 360$) — **Черепашка** поворачивается по часовой стрелке на m градусов.

Запись `Повтори k [команда1 команда2 команда3]` означает, что последовательность команд в квадратных скобках повторится k раз.

Черепашке для исполнения был дан алгоритм:

`Повтори 8 [Направо 35 Вперед 25 Направо 85]`

Варианты (в зависимости от ответа):

1) правильный восьмиугольник — “сделать это на компьютере”;
2) правильный треугольник — “научить этому компьютер”;
3) ромб — “создать программу для компьютера”;
4) незамкнутая ломаная линия — “решить задачу с помощью компьютера”.

От редакции. Ответы, пожалуйста, присылайте в редакцию.

Численность класса

В классе 1010 мальчиков, что составляет 101 000% от численности всего класса. Сколько в классе девочек и какова общая численность класса? Ответы получите в двоичной системе, в которой записаны все приведенные значения.

Три друга

Встретились три друга — Токарев, Слесарев, Плотников. Их отцы работают плотником, токарем и слесарем. “Интересно, что ни один из наших отцов не работает по той специальности, от которой произошла его фамилия”, — сказал мальчик, чей отец был слесарем. “А ведь ты прав”, — согласился Токарев. Кем работают отцы каждого из мальчиков?

Три задачи

Е.А. Мирончик,
учитель информатики МБНОУ “Лицей № 111”,
г. Новокузнецк Кемеровской обл.

1. Как, конечно, известно читателям, в операционной системе Windows *папкой* называют группу файлов, объединенных по какому-то признаку и снабженных именем. В любой папке могут также быть другие папки. До разработки системы Windows папки называли “каталогами”, вложенные в них другие каталоги — “подкаталогами”, а тот каталог, в который вложены подкаталоги, — “родительским по отношению к ним”. Каталог, у которого нет “родителя”, — это “корневой каталог”. Ясно, что один и тот же каталог может быть одновременно и родительским (по отношению к одним каталогам), и подкаталогом (другого).

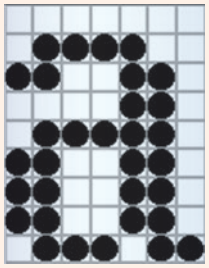
Можно ли построить дерево каталогов, соответствующее *всем* перечисленным ниже условиям?

1. В корневом каталоге имеется три каталога: A, B, C.
2. Каталог A является родительским по отношению к E.
3. Каталоги D, F являются подкаталогами каталога E.
4. В каталоге B есть подкаталоги F и M.
5. M является родительским каталогом каталога C.

2. Результат выражения $2^{354} + 2^{356}$ записали в системе счисления с основанием 8.

Сколько цифр в полученном числе и какая первая цифра?

3. На компьютерах первых поколений использовались так называемые “матричные” принтеры. В них изображение на бумаге создавалось с помощью печатающей головки с набором иголок через красящую ленту ударным способом. Каждый символ получался при использовании какого-то числа иголок (см. рисунок вверху страницы).



Представьте, что головка матричного принтера состоит из 24 иголок, расположенных в прямоугольнике 4 × 6. С ее помощью можно печатать различные символы — русские и латинские буквы, математические и технические знаки, ноты, знаки препинания, иероглифы, арабскую вязь и многие другие символы.

На листе бумаги формата A4 (210 × 297 мм) таким принтером можно напечатать 64 строки по 128 символов в строке.

Сколько потребуется листов бумаги для печати всех вариантов символов, которые можно напечатать с помощью принтера (каждый символ — в одном экземпляре)?

От редакции. Ответы присылайте в редакцию (можно решать не все задачи).

Дуэль мушкетеров[1]

— Ваше преосвященство, — докладывал Рошфор кардиналу Ришелье, — вчера вечером я оказался в Сен-Клу около харчевни “Голубой павлин”. Вскоре вслед за мной подъехала целая кавалькада мушкетеров со слугами. Я не знаю никого из них, но расслышал несколько произнесенных ими титулов и имен и разглядел некоторые гербы и эмблемы, украшающие шляпы, или одежду слуг, или лошадиную сбрую.

Посетителей в “Голубом павлине” было немного, и мушкетеры расселись по двое к отдельным столикам. Мушкетер с эмблемой медведя сидел с неким графом, некий д’Эстрэ — с мушкетером, к которому все обращались как к герцогу, а некий

[1] Эта задача была опубликована в далеком 2004 году (можно сказать — “до вашей эры” ☺). Почему редакция повторяет ее, мы объясним при публикации ответа к задаче.

Ла Кост — с мушкетером, герб которого, как видно, не зря содержал разъяренного вепря: вскоре вспыхнула ссора.

Перебранка закончилась дуэлью на весьма суровых условиях: каждому предстояло поочередно сразиться со всеми, кроме, разумеется, своего застольного друга. Здесь же, на улице, около "Голубого павлина" мушкетер с геральдическим львом скрестил шпагу с упомянутым графом; какой-то шевалье — с тем, кого он именовал Монтераном, а Бюзаньи — с мушкетером, имеющим в гербе оленя. Схватка закончилась незначительными царапинами. Тут появился гвардейский патруль и вся компания так проворно вскочила в седла, что я едва поспел за ними. Мы проскакали до монастыря Дешо, и там, на пустыре, возобновился бой, причем теперь каждый сменил своего визави: мушкетер с эмблемой волка дрался с неким Вьевилем; какой-то виконт — с Пьютанжем, а мушкетер с геральдическим вепрем — против маркиза.

— Погодите, Рошфор! — прервал его кардинал. — Ваши сведения никуда не годятся, такой доклад невозможно преподнести королю. Кто скрывается под гербами? Кто такие эти граф, виконт, герцог? Мне нужны только точные данные. Но рассказывайте дальше, чем закончилась схватка у монастыря.

— Тяжело были ранены двое, монсеньор, — ответил Рошфор. — Раненых унесли в монастырь, а сражение разгорелось с еще большим ожесточением, возможно, потому, что вновь произошла смена противников. Все закончилось тем, что мушкетер с эмблемой лисицы выбил шпагу у барона, а с гербом медведя загнал д'Эстрэ в угол, после чего произошло примирение.

— Рошфор! Стойте! Так у Вас же превосходные данные. Мне ясно, у кого какие эмблемы и титулы! Я немедленно иду к королю! — произнес кардинал. — На этот раз капитан мушкетеров де Тревиль не отвертится — ему не удастся.

А Рошфор, втайне обрадованный сменой настроения своего повелителя, не сразу смог уяснить, каким образом из отрывочных сведений можно было воссоздать что-то цельное. Он взглянул на листок, на котором кардинал начертал имя, титул и герб каждого из шести мушкетеров, и ему показалось, что, быть может, Его преосвященство и в самом деле обладал прозорливостью.

Может ли кардинал сделать точный доклад королю, в котором будут указаны имя, титул и герб каждого из шести мушкетеров? Ответы с обоснованием, пожалуйста, присылайте в редакцию.

Три пирата

У пиратов А, Б и В состоялся такой разговор:
А: "У Б — 2 глаза".
Б: "У В — 2 глаза".
В: "У А — 2 глаза".
А: "У нас 2 глаза на троих".
Б: "У нас 3 глаза на троих".
В: "У нас 4 глаза на троих".

Оказалось, что каждый соврал столько раз, сколько у него глаз. Сколько глаз у каждого из пиратов?

Ответы, решения, разъяснения

к заданиям, опубликованным в августовском выпуске "В мир информатики"

Задача "Бальзам"

Напомним, что требовалось разработать алгоритм решения задачи разделения 24 литров бальзама, имеющегося в 24-литровом сосуде, на три равные части, используя еще сосуды объемом 5, 11 и 13 л.

Решение представлено в виде таблицы:

№	Действие	*24 л*	*5 л*	*11 л*	*13 л*
Исх.		24	0	0	0
1	24 → 5	19	5	0	0
2	24 → 11	8	5	11	0
3	5 → 13	8	0	11	5
4	11 → 13	8	0	3	13
5	13 → 5	8	5	3	8
6	5 → 11	8	0	8	8

Правильные ответы прислали:

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

— Курапин Евгений, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Лакшина Ксения и Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Леонова Элина, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

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

— Труханов Глеб и Чухлов Вадим, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.** (Вадим и Глеб оформили решение в виде презентации Microsoft PowerPoint);

— Якушин Алексей, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**

Задача "Культурные развлечения"

Напомним условие: "Андрей, Борис, Евгений, Ольга, Рита, Полина, Дмитрий и Света — друзья. В одно из воскресений Андрей отправился на концерт, Борис провел вечер с Ольгой, Евгений так и не встретил Риту, Полина побывала в кино, Рита посмотрела спектакль в театре, какая-то пара посетила художественную выставку. Неизвестно, где были Света и Дима, но известно, что каждый юноша был в театре, на выставке, на концерте или в

кино с одной из девушек. Определите, кто с кем и где побывал в воскресенье”.

Ответ: Андрей и Света были на концерте, Борис и Ольга — на выставке, Евгений и Полина — в кино, Дмитрий и Рита — в театре.

Правильные ответы прислали:

— Васина Светлана и Хомутова Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Гильзер Ирина, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

— Коробов Сергей и Яснов Федор, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Еличева Анастасия и Труханова Яна, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.** (Анастасия и Яна оформили решение в виде презентации Microsoft PowerPoint);

— Курапин Евгений, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

— Леонова Элина, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Мухина Светлана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

— Филиппенко Михаил, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**

Викторина “Олимпийские игры”

Ответы

1. Согласно легендам, Олимпийские игры организовал Геракл (вариант ответа Б).

2. Олимпиадой звали мать Александра Македонского (вариант ответа Г).

3. В античности Олимпийские игры проходили, естественно, в Олимпии, на полуострове Пелопоннес (вариант ответа Б).

4. Участвовать в античных Олимпийских играх могли сначала только греки, а потом еще и римляне (вариант ответа Б).

Правильные ответы прислали:

— Аксененко Ирина и Чумаков Илья, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Герасимова Наталья и Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Гильзер Ирина, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

— Леонова Элина, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Марков Алексей, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Новикова Анна и Потапова Алевтина, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Филимонова Галина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Решение японских головоломок “судоку” представили:

— Бадикова Ольга, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

— Бородюк Анна и Василенко Татьяна, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

— Мухина Ирина, г. Фрязино Московской обл., школа № 4, учитель **Сенюта Е.И.**;

— Русскова Мария, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

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

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

— Фомина Маргарита, г. Саратов, школа № 52, учитель **Пуйшо Н.В.**

Задачи из августовского выпуска, на которые не пришли ответы ☹

Люди в комнатах

В n комнатах находятся $n + 1$ человек. На дверях первой написано: “Здесь находится один человек”, на дверях второй: “Здесь находятся два человека”, …, на дверях n-й: “Здесь находятся n человек”. Известно, что ровно одна из этих надписей неверна. Чему может равняться n и сколько человек находится при этом в каждой комнате?

Малыши изучают алфавит

Малыши Аня, Даня и Маня изучали алфавит. Оказалось, что из букв, которые знает Аня, Дане известны только “а” и “и”, а Мане — только “а” и “л”. При этом Аня с Даней, использовав все буквы, которые знают, сумели написать слово “АЛФАВИТ”, а Даня с Маней, опять-таки использовав все свои знания, написали слово “ПИЛОТКА”. Какие буквы изучил каждый ребенок?

Три задачи

Определите, если это возможно, какой карандаш самый короткий и какой — самый длинный из желтого, синего и красного карандашей, если известно, что:

а) синий карандаш короче желтого, а желтый короче красного,

б) желтый длиннее синего, а синий длиннее красного,

в) красный длиннее желтого, а желтый короче синего.

Задачи предназначены для учащихся начальной школы.

Новая задача о разрубании кубика

Имеется кубик размером 4 × 4 × 4. Какое минимальное число разрубаний потребуется, чтобы разрубить его на маленькие кубики размером 1 × 1 × 1 (при разрубании разрешены наложения кусков кубика друг на друга)?

На следующие задания из августовского выпуска ответы прислали:

— Андрющенко Александр, Остроухова Валерия, Пономаренко Анастасия и Уткина Ксения, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Калачев Андрей, Москва, школа № 872, учитель **Липина В.М.**;

— Леонова Элина, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Трептау Татьяна, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**

Спасибо!

В тире

Три студента — Андреев, Борисов и Володин — стреляли в тире по специальной мишени.

Каждый из них сделал по шесть выстрелов. Места попаданий в мишень отмечены на рисунке точками. Когда подсчитали результаты, оказалось, что каждый выбил по 71 очку. При этом из всех восемнадцати выстрелов только один дал попадание в центральный круг мишени (50 очков).

Кому из студентов — Андрееву, Борисову или Володину принадлежит этот удачный выстрел? Установить это можно по следующим данным: первые два выстрела дали Андрееву 22 очка; первый выстрел Володина дал ему только 3 очка.

Пять монет на столе

В ряд расположены пять монет:

Средняя монета лежит вверх орлом, остальные — решкой. За один ход разрешается перевернуть любые три рядом лежащие монеты.

1. Можно ли добиться, чтобы все монеты лежали орлом вверх?

2. А если вверх орлом первоначально лежала только первая монета?

3. А если только вторая монета?

Ответы, пожалуйста, оформите с использованием условных обозначений: *О* — орел, *Р* — решка.

Игрушки

У одного мальчика-инопланетянина было 12 игрушек. Когда ему подарили еще 5, то общее число игрушек стало равно 21. Как такое могло быть?

Пять вопросов

1. Кто, нападая на свою жертву, закрывает глаза, чтобы бьющаяся добыча их не поранила?

2. Что имели в виду братья Гонкуры, утверждая, что она при своем появлении никогда не бывает шедевром, а становится им потом?

3. Какой город в Смоленской области издревле славился своими пряниками и коврижками?

4. Кто из чемпионов мира по шахматам умел легко перемножать в уме трехзначные числа?

5. Какая популярная музыкальная группа приехала в Москву из Таганрога?

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

“РОССИЯ” — три числовых ребуса

Решите, пожалуйста, три числовых ребуса:

1. **РО^{С} = СИЯ**.
2. **РОС = СИ^{Я}**.
3. **РО^{С} = СИ^{Я}**.

В них, как принято в таких головоломках, одинаковыми буквами зашифрованы одинаковые цифры, разными буквами — разные цифры (в разных ребусах одной и той же букве могут соответствовать разные цифры).

Ответ получите не полным перебором возможных вариантов, а методом рассуждений, используя компьютер, например, программу Microsoft Excel или подобную для вспомогательных расчетов.

Жучок в пиджачке

Вот стихотворение, которое написал поэт Андрей Усачев:

Шел по улице жучок
в модном пиджачке.
На груди висел значок,
а на том значке
нарисован был жучок
тоже в пиджачке.
И на нем висел значок,
а на том значке
был еще один жучок…

Как в программировании называется прием, аналогичный описанному в стихотворении?

Орнаменты, полученные из повторяющихся геометрических фигур, — так называемые “гирихи”, о которых рассказывалось в статье “Постижение гириха” в августовском выпуске, подготовили:

— Ильин Виктор и Касимова Яна, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.** (Виктор представил также презентацию Microsoft PowerPoint, иллюстрирующую методику построения);

— Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**

Виктор, Дарья и Яна будут награждены дипломами. Поздравляем!

Ответы на задания, опубликованные в апрельском выпуске “В мир информатики”, представили также учащиеся школы № 9 г. Ноябрьска, Тюменская обл., Ямало-Ненецкий автономный округ (учитель **Коровин Д.В.**):

— ответы на пять вопросов, приведенные в рубрике “Поиск информации”, — Иванова Светлана, Комарова Валентина, Кузнецова Альбина, Миронов Игорь, Павлиогло Диана и Ступникова Лада;

— решения девяти ребусов, связанных с программой Microsoft PowerPoint, — Кузнецова Альбина и Ступникова Лада;

— ответы на кроссворд — Комарова Валентина, Кузнецова Альбина, Павлиогло Диана и Ступникова Лада;

— решение числового ребуса со словом-числом РЕПКА — Миронов Игорь (Игорь привел подробные комментарии к решению).

Решение задачи “Восемь семейных пар” (с подробным обоснованием) прислали также:

— Звездов Артем, г. Фрязино Московской обл., школа № 4, учитель **Сенюта Е.И.**;

— Максютина Эльвира, Удмуртская Республика, г. Можга, школа № 1, учитель **Колесникова С.В.**;

— Удалова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

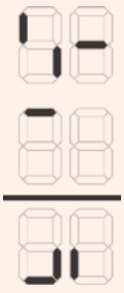
Как и другие читатели, не побоявшиеся взяться за решение этой непростой задачи (их фамилии опубликованы в предыдущем выпуске “В мир информатики”), Артем, Елизавета и Эльвира будут награждены дипломами. Молодцы!

“ЛОМАЕМ” ГОЛОВУ

Неисправный калькулятор

На калькуляторе цифры изображаются с помощью так называемых “7-сегментных” индикаторов:

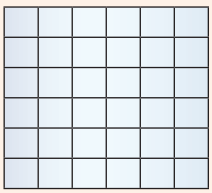
Представьте себе калькулятор, на котором отображаются оба числа, над которыми проводится действие, а также результат.



Но этот калькулятор сломался — у него горят не все сегменты. Сможете ли вы определить, какое действие и с какими числами выполнялось? Известно, что выполнялось либо сложение, либо вычитание. Не горят (а должны) 20 сегментов.

Расставить шашки

Расставьте шашки на клетчатой доске 6 × 6 так, чтобы на всех горизонталях стояло разное число шашек, а на всех вертикалях — одинаковое.



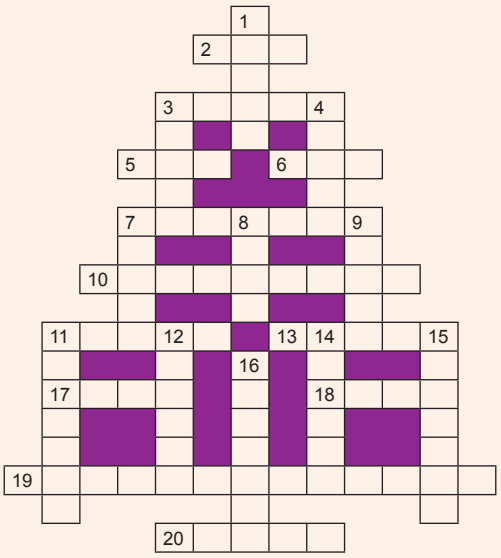
Ответ, пожалуйста, оформите в виде текстового документа.

Получить верное равенство

Перед вами равенство 7 + 4 – 4 = 0, которое не выполняется. Переложите всего одну спичку, чтобы оно стало верным.

Задание предназначено для учащихся 1–7-х классов.

Новогодний кроссворд-“елочка”



По горизонтали

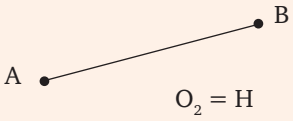
2. Число в системе условных обозначений символов.
3. Прямоугольник, ограничивающий меню или т.п.
5. … координат.

6. Цифра восьмеричной системы счисления.
7. Последовательная смена состояний, стадий развития.
10. Строение, внутреннее устройство.
11. Неисправность в работе компьютера.
13. Величина, с помощью которой осуществляется счет.
17.

18. Так называют любую запись на странице в Интернете (сообщение, вопрос, ответ, комментарий и т.п.).
19. Один из двух видов транслятора — системной программы, которая "переводит" программу с языка программирования высокого уровня в машинные команды.
20. Единица измерения высоты шрифта.

По вертикали

1. Один из объектов баз данных, а также внешнее очертание, наружный вид предмета.
3. Массив точек, из которых формируется изображение на экране.
4. Координаты ячейки в электронной таблице.
7. Непрерывная последовательность данных.
8. Алгоритмическая конструкция, обеспечивающая повторение одних и тех же операций.
9. Отмена всех установленных параметров компьютера.
11.



12. Устройство управления работой шины персонального компьютера, а также спортивный судья.
14. Ввоз товаров из-за рубежа, а также вставка в документ приложений Windows объектов из других приложений.
15. Один из результатов целочисленного деления.
16. Язык программирования.

Сколько стоят буквы

В одном магазине продавались английские буквы для адресов домов. Одинаковые буквы стоили одинаково, а разные имели различные цены.

Известно, что слово **ONE** стоит 6 долларов, слово **TWO** — 9 долларов, а слово **ELEVEN** — 16 долларов. Сколько стоит слово **TWELVE**?

Автор задачи — Григорий Гальперин

Цифры на кубике

На гранях игрального кубика написаны шесть различных цифр:

Сумма цифр на противоположных гранях одна и та же для каждой пары параллельных граней. Каковы остальные цифры?

Задание предназначено для учащихся начальной школы.

ДЛЯ ЭРУДИТОВ

Викторина "Три вопроса"

Выберите один из трех вариантов ответа, который, по вашему мнению, является правильным.

1. Англоязычное название Китая — China, вероятнее всего, произошло от императорской династии Цинь. При ее правлении Поднебесная превратилась в единую страну. "Объединителем" же суждено было стать императору Цинь Шихуанди. До какого года продлился нерушимый период упомянутой династии?

А. До 1912-го.
Б. До 1812-го.
В. До 1712-го.

2. Изначально на могиле Н.В. Гоголя на монастырском кладбище лежал камень, прозванный за глаза Голгофой из-за схожести с Иерусалимской горой. При перезахоронении на могиле установили бюст писателя. А то самое надгробие впоследствии переместилось на могилу другого отечественного классика, кто очень часто адресовал Николаю Васильевичу такие слова: "Учитель, укрой меня своей шинелью". Кто сейчас покоится под этим камнем?

А. Михаил Зощенко.
Б. Аркадий Аверченко.
В. Михаил Булгаков.

3. Известная картина Рафаэля "Сикстинская мадонна" в 1955 году была возвращена вместе с другими шедеврами в Дрезденскую галерею. Изобразительное мастерство художника обманывает наблюдателя: задний фон, издали кажущийся облаками, при тщательном рассмотрении оказывается головами ангелов. Какой головной убор размещен у ног святого Сикста?

А. Тиара.
Б. Терновый венец.
В. Диадема.

КРЕПКИЙ ОРЕШЕК

Задание “Сложение двух двоичных цифр”

Напомним, что было предложено на основе анализа таблицы сложения в двоичной системе счисления:

$$+\begin{array}{r}0\\0\\\hline 0\end{array}\qquad +\begin{array}{r}0\\1\\\hline 1\end{array}\qquad +\begin{array}{r}1\\0\\\hline 1\end{array}\qquad +\begin{array}{r}1\\1\\\hline 1\;0\end{array}$$

записать формулы, определяющие значения цифры результата r и значения переноса p в зависимости от значения слагаемых a и b, в которых операции сравнения (условия) не используются.

Изменим заданную в условии таблицу сложения, записав в нее нулевые значения переноса для трех первых случаев:

$$+\begin{array}{r}0\\0\\\hline 0\;0\end{array}\qquad +\begin{array}{r}0\\1\\\hline 0\;1\end{array}\qquad +\begin{array}{r}1\\0\\\hline 0\;1\end{array}\qquad +\begin{array}{r}1\\1\\\hline 1\;0\end{array}$$

Ее анализ показывает, что значения r и p могут быть найдены по формулам:

$$r = a \textbf{ XOR } b;$$
$$p = a \textbf{ AND } b,$$

где **XOR** и **AND** — логические операции, которые применяются к числам. Эти операции выполняются в процессоре компьютера (поэтому их называют также логическими командами) над числами, представленными в двоичном виде. В отличие от арифметических операций над двумя операндами

логические команды являются *поразрядными*. Например, при сложении двух двоичных цифр возможен перенос в старший разряд, а при логических операциях все разряды рассматриваются изолированно друг от друга. Таблицы результатов применения этих операций следующие:

AND	0	0	1	1
	0	1	0	1
Результат	0	0	0	1

OR	0	0	1	1
	0	1	0	1
Результат	0	1	1	1

XOR	0	0	1	1
	0	1	0	1
Результат	0	1	1	0

Предлагаем читателям решить еще одну задачу, связанную с логическими командами.

Найти количество единиц

Дано двухразрядное двоичное число (может быть с начальным нулем или состоящее из двух нулей). Определим количество единиц в нем. Это количество можно записать в виде r_1r_2 (00, 01 или 10). Найдите зависимости значений r_1 и r_2 от цифр заданного числа. Например, если заданное число равно 10 (одна единица), то $r_1 = 0$, $r_2 = 1$, если 11 (две единицы) — то $r_1 = 1$, $r_2 = 0$.

Операции сравнения (условия) не использовать.

ЭТО ПОЛЕЗНО ЗНАТЬ

Швейцарский банк… цифровой информации

Известно, что Швейцария давно славится своими надежными банками — хранилищами денег. Но в настоящее время к ней, как “хранительнице тайн”, добавилась известность как страны, в которой создано самое надежное в Европе, а может быть и в мире, хранилище цифровой информации.

Его предыстория такова. В начале 90-х годов прошлого века отставному военному Кристофу Ошвальду пришла в голову идея: ставшие после окончания “холодной войны” ненужными подземные сооружения швейцарской армии можно использовать для надежного хранения компьютерной информации, где она будет защищена от всевозможных угроз. Пользуясь своими связями в генеральном штабе, Ошвальд получил в аренду систему бункеров, вырубленную в 70-х годах глубоко в альпийских скалах и рассчитанную на защиту от атомного, химического и бактериологического оружия. Вход в подземелье находится в одной из альпийских долин Бернского нагорья и замаскирован. Затратив почти 50 миллионов швейцарских франков, убежище оборудовали сетью оптоволоконных кабелей, пятикратно дублированной системой энергоснабжения и многоступенчатым контролем на входе. “Банк” начал работать в 1996 г.

Это защитное сооружение настолько фундаментально, что страховые компании оценивают степень риска хранящихся в нем материалов как нулевую.

Утрата данных считается сегодня одной из самых крупных катастроф, какие могут случиться с предприятием. Потеря сведений о клиентах, паролей доступа к счетам, секретов технологии и менеджмента часто может означать полное разорение фирмы. К уничтожению информации могут привести “естественные” случайности — удар молнии в кабель, пожар, наводнение, но значительно чаще причиной бедствия становится ошибка персонала или проникновение хакеров, шпионов и других кибертеррористов. Так что швейцарский банк цифровой информации, расположившийся в системе подземных бункеров, оказался удивительно своевременным.

В многочисленных разветвлениях этого подземелья достаточно места для размещения громадного количества носителей информации. В его темных помещениях при температуре 19°С и относительной влажности 45% хранится цифровой архив швейцарского правительства (возможно, и других стран), сотен местных фирм и десятков иностранных компаний.

Охраняют центр хранения информации элитные части швейцарской армии — те самые, что охраняют государственный банк Швейцарии. “Фирменный продукт нашей страны, — говорит Кристоф Ошвальд, — безопасность и серьезный подход к делу”. Если по каким-то причинам у клиента погибнет база данных, то сберегаемая здесь информация в надежно закодированном виде по самому широкому каналу связи Швейцарии будет выслана хозяину компьютера, потерпевшего катастрофу. Если это по каким-то причинам невозможно или нежелательно, с замаскированной взлетно-посадочной полосы взлетит вертолет или реактивный самолет, который доставит клиенту информацию на носителях.

Надо заметить, что хранилища цифровой информации в настоящее время существуют не только в Швейцарии (например, почти все фирмы, размещавшиеся в двух небоскребах Всемирного торгового центра в Нью-Йорке, разрушенных 11 сентября 2001 года в результате нападения террористов “Аль-Каиды”, держали свои данные в одном из таких банков). Но они, конечно, не имеют столь мощного укрытия и тщательной законспирированности, как альпийское.

Литература

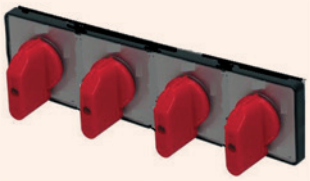
1. “Наука и жизнь”, 2002, № 1.

2. *Чеботарева А.В.* Примета XXI века. / “Потенциал”, 2013, № 4.

ВНИМАНИЕ! КОНКУРС

Конкурс № 106

Представьте, что вам нужно открыть сейф, в котором имеются четыре переключателя, каждый из которых может находиться в одном из двух положений — вертикальном или горизонтальном:



Сейф откроется при каком-то одном сочетании положений переключателей. В исходном состоянии все переключатели вертикальны. Предложите последовательность перебора *всех*[2] возможных вариантов состояния, при котором общее число переключений из одного положения переключателей в другое будет минимальным. Например, если сначала повернуть в горизонтальное положение 1-й и 4-й переключатели (1-е состояние), а затем вернуть 1-й переключатель в вертикальное положение, а горизонтально расположить 3-й переключатель (2-е состояние), то общее число переключений в этот момент станет равно четырем (2 + 2).

Ответ оформите в виде таблицы:

№ п.п.	Состояние переключателей	Число переключений
Исх.		
1		2
2		2
3		1
…	…	…
	Всего	26

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

[2] Ясно, что сейф может быть открыт и при некотором не последнем варианте.

Педагогический университет
«Первое сентября»

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

ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

С 1 ноября производится прием заявок на второй поток 2013/14 учебного года

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **108** УЧЕБНЫХ ЧАСОВ

  Стоимость – 2990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **72** УЧЕБНЫХ ЧАСА

  Стоимость – 2390 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

Перечень курсов и подробности – на сайте edu.1september.ru

Пожалуйста, обратите внимание:

заявки на обучение подаются только из Личного кабинета, который можно открыть на любом сайте портала www.1september.ru

Общероссийский проект

Школа цифрового века

Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

2013/14 учебный год

До завершения регистрации
на участие в проекте этого года
остался один месяц

Успейте подать заявку
до 31 декабря!

Педагогические работники образовательного учреждения получат доступ ко всем журналам и методическим брошюрам Издательского дома «Первое сентября». А также смогут повысить свою квалификацию на модульных курсах «Навыки профессиональной и личной эффективности» и «Инклюзивный подход в образовании».

Подробности и прием заявок на сайте

digital.1september.ru

Прием заявок на 2014/15 учебный год открывается 1 января 2014 года