№7–8 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 1september.ru
июль – август 2013
ИНФОРМАТИКА Подписка: «Роспечать» – 32291 (бумажная версия), 19179 (электронная); «Почта России» – 79066 (бумажная версия), 12684 (электронная)

НА ОБЛОЖКЕ

"Секретная" лаборатория Google X, которая не так давно "выстрелила" на рынок продуктом будущего – очками Google Glass, приступила к натурным испытаниям флотилии воздушных шаров для раздачи Интернета в труднодоступных регионах. Предполагается, что шары будут в течение длительного времени находиться на высоте около 20 км,самостоятельно корректировать собственное местоположение и обеспечивать скорость Интернета не хуже 3G (настоящего 3G, а не как это у нас часто бывает).

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

Исходные файлы и презентации к статьям номера

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский
(редактор вкладки "В мир информатики")
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 26 317 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 – М.Сартан
История – А.Савельев
Классное руководство и воспитание школьников – М.Битянова
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
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
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 11.06.2013,
фактически 11.06.2013
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

“Если друг оказался вдруг и не друг...”, или Если друг оказался вдруг... другом

Каждый день ко мне в Facebook стучатся несколько новых друзей. День на день не приходится, но в среднем — я сейчас специально подсчитал — около трех в день. В подавляющем большинстве случаев, заходя в хронику нового потенциального “незнакомого друга”, на первом экране видишь событие “присоединился к Facebook”

Не принимая в Facebook запрос на добавление в друзья, я не жму на кнопку “Отклонить”, а просто оставляю запрос без ответа. Во-первых, неловко, а во-вторых, незачем портить человеку фейсбучную “карму” — вроде как после какого-то количества отклоненных запросов Facebook налагает на пользователя некие санкции, своеобразный “визовый карантин”. В результате список не принятых в друзья у меня на порядок больше списка друзей.

Но я вот к чему это пишу. Моя личная стратегия, которая в сжатом виде выглядит так — “совсем незнакомых людей в друзья не добавлять, также не добавлять в друзья людей, с которыми не хотел бы общаться в жизни”, разумеется, не является более правильной, чем... чем любая другая. Вообще слово “правильная” здесь не подходит — у меня стратегия такая, у кого-то другая, у меня она — моя, у моих друзей — своя. Но, видимо, точно не здорово не иметь никакого подхода к организации своей активности в социальных сетях (понятно, что Facebook лишь частный случай). И лучше всего, хотя это сложно — личные стратегии приходят с личным опытом — начинать использовать свои правила организации активности с самого начала использования социальных сервисов. Мне, например, так сделать не удалось, но я довольно рано себя “поймал”, и объем работ по приведению в порядок жизни в социальных сетях был не очень велик. Годом позже, думаю, это было бы уже много труднее.

Кстати, что касается именно Facebook, в этой сети имеется достаточно технических средств для решения задач, связанных с “областями видимости” (термин наш, “программистский”, но оказывается очень к месту). Например, такая удобная штука, как “подписка”. Чтобы “на вас” можно было подписаться, соответствующую возможность вы сами должны включить у себя в настройках. После этого тот, кто понимает, что такое подписки ☺, сможет получать ваши публичные обновления, не добавляясь в друзья. Есть масса и других вещей, например, страницы. Кстати, довольно много компаний, не разобравшись, в чем дело, для публичного взаимодействия с клиентами заводили личный аккаунт вместо страницы, что запрещено (и совершенно справедливо) правилами Facebook. Польза от этого лишь в том, что такое поведение создало отдельный рынок услуг по миграции из аккаунта в страницу по достижению предельного количества в 5000 “друзей” ☺. Стоит не дорого. Можно, разумеется, все сделать и самому, описаний, что и как делать, достаточно.

Помимо подписок и страниц, есть группы, есть списки друзей (список в данном случае — термин) и другие инструменты. Как и чем пользоваться, каждый решает сам. Можно и на самотек все пустить и не тратить на это время. Но в будущем это может быть чревато проблемами. А может и не быть чревато — зависит от вашего собственного отношения к вопросу ☺.

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

XML: практикум

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

Введение

Одна из сложнейших задач, которую приходится решать специалистам в области информационных технологий, — это *интеграция*, то есть обеспечение совместной работы систем с различным аппаратным и программным обеспечением. Представьте себе, что вам нужно передать данные (например, из локальной базы данных) в другую фирму через Интернет. Вы понятия не имеете, какую технику и какое программное обеспечение использует эта фирма, однако по требованию заказчика данные должны быть:

- представлены в текстовой форме;
- переданы с сохранением структуры (например, иерархической);
- пригодными для автоматической обработки;
- понятными для человека при просмотре в текстовом редакторе.

В этой ситуации сразу возникает мысль о разработке “велосипеда” — своего текстового формата представления данных[1]. Однако при этом нужно сообщить вашему адресату, как работать с такими данными, то есть составить полное описание формата. Заказчику, в свою очередь, придется написать программное обеспечение для обработки ваших данных, что скорее всего не входит в его ближайшие планы. Поэтому такой вариант нельзя считать приемлемым.

Для решения поставленной задачи нужен *открытый* формат (описание которого общедоступно и не ограничено действием никаких лицензий), который является общепризнанным стандартом и широко поддерживается современными программными средствами. Такой формат должен содержать только “чистые данные”, без какого-либо указания на форму их представления на устройствах вывода. Сейчас в качестве универсального

[1] Двоичные данные, такие, как рисунки, звук, видео и т.п., тоже могут быть представлены с помощью набора символов ASCII, например, при кодировании в UUE (*http://ru.wikipedia.org/wiki/UUE*) или Base64 (*http://ru.wikipedia.org/wiki/Base64*), как это делают почтовые программы при пересылке файлов.

стандарта описания любых данных, которые можно представить в текстовом виде, служит XML (англ. *eXtensible Markup Language* — расширяемый язык разметки).

Язык XML, первая спецификация которого была принята к началу 1998 года, представляет собой подмножество обобщенного языка разметки SGML (англ. *Standard Generalized Markup Language*), который был разработан еще в 1986 году, но оказался слишком сложным для повсеместного практического использования. XML был построен на основе SGML с целью упрощения автоматической обработки документов.

Приведем несколько широко известных примеров применения XML в информационных технологиях:

• формат хранения документов *Open Document Format* [1], используемый в пакете *OpenOffice.org*;

• формат хранения документов *Office Open XML* [2], применяемый в *Microsoft Office* последних версий (файлы документов — это ZIP-архивы, содержащие набор XML-файлов);

• язык SVG (англ. *Scalable Vector Graphics* — масштабируемая векторная графика) для описания векторной графики, которую можно выводить на веб-страницах;

• язык описания математических выражений MathML (от англ. *Mathematical Markup Language* — язык математической разметки);

• семейство форматов RSS (англ. *Really Simple Syndication* — очень простой сбор сводной информации) для описания лент новостей сайтов и блогов в Интернете;

• формат хранения электронных книг `.fb2` (FictionBook);

• протокол SOAP (англ. *Simple Object Access Protocol* — простой протокол доступа к объектам) для обмена сообщениями между приложениями в сети, не зависящий от операционных систем, программного обеспечения и языков программирования;

• XML-файлы, хранящие настройки программ (конфигурационные файлы);

• язык XAML (англ. *eXtensible Application Markup Language* — расширяемый язык разметки приложений), который применяется для описания пользовательских интерфейсов при программировании в .NET во всех современных версиях *Microsoft Windows*, включая *Windows 8*.

Например, вот так выглядит начало электронной книги М.А. Булгакова “Мастер и Маргарита” в формате `.fb2`:

```
<?xml version="1.0" encoding="UTF-8"?>
<FictionBook xmlns="http://www.gribuser.ru/xml/fictionbook/2.0"
             xmlns:xlink="http://www.w3.org/1999/xlink">
<description>
  <title-info>
    <genre>literature_classics</genre>
    <author>
      <first-name>Михаил</first-name>
      <middle-name>Афанасьевич</middle-name>
      <last-name>Булгаков</last-name>
    </author>
    <book-title>Мастер и Маргарита</book-title>
    <date value="1940-01-01">1929-1940</date>
    <lang>ru</lang>
  </title-info>
</description>
...
```

Так хранится список команд контекстного меню *Undo* (англ. *отмена*) в редакторе GIMP:

```
<?xml version="1.0" encoding="UTF-8"?>
<ui>
  <popup action="undo-popup">
    <menuitem action="edit-undo" />
    <menuitem action="edit-redo" />
    <menuitem action="edit-undo-clear" />
  </popup>
</ui>
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <rect x="50" y="20" width="300" height="100"
        style="fill:rgb(0,0,255);stroke-width:2;stroke:rgb(255,0,0)" />
</svg>
```

По внешнему виду такие файлы напоминают веб-страницы на языке HTML — для разметки используются тэги, заключенные в угловые скобки; контейнеры (парные тэги) ограничены открывающим и закрывающим тэгами с одинаковыми названиями. Однако между HTML (который, кстати,

тоже основан на SGML) и XML есть существенные отличия.

Прежде всего у этих языков различные задачи, так что ни один из них не может вытеснить другой. XML был разработан специально для *описания данных*, тогда как значительная часть тэгов HTML управляет *отображением данных* на веб-странице. Поэтому во многом возможности XML и HTML дополняют друг друга.

С помощью XML-разметки легко описывать документы, имеющие иерархическую структуру (например, книги, разбитые на главы и разделы), а также небольшие базы данных, которые требуется отображать различными способами и организовывать поиск, сортировку, фильтрацию. В то же время для просмотра этих данных с помощью браузера нужно построить веб-страницу на языке HTML. Таким образом, мы разделяем содержание (данные) и оформление (форму их представления), используя для решения каждой задачи свой инструмент.

Тэги и структура HTML-документа заданы стандартами (например, в HTML нет элементов для описания музыкальных символов и математических формул), тогда как XML — это расширяемый язык, то есть пользователь может вводить свои тэги и задавать свою структуру документа[2].

В XML введены более строгие формальные требования к документу, которые позволяют упростить его автоматическую обработку:

- регистр букв в названиях тэгов имеет значение, например, **<author>** и **<Author>** — это разные тэги; чтобы не путаться, рекомендуется записывать все тэги строчными буквами;
- каждый элемент должен содержать открывающий и закрывающий тэги; пустые элементы (не содержащие в себе данных) можно закрывать символами “**/>**”, например, вместо **<foo> </foo>** можно писать **<foo и />**;
- все атрибуты тэгов (расположенные внутри открывающего тэга) обязательно заключать в кавычки;
- XML-документ должен содержать корневой элемент (контейнер), в который “вложены” все остальные элементы;
- нужно соблюдать вложенность тэгов: любой вложенный элемент обязан полностью содержаться во внешнем контейнере; например, недопустима запись, в которой области действия тэгов перекрываются:

```
<car><power>20</car></power>
```

Если дополнить HTML этими ограничениями, мы получим язык XHTML (англ. *eXtensible Hypertext Markup Language* — расширяемый язык разметки гипертекста), который описан в рекомендации консорциума W3C [3]. Популяризация XHTML как “правильного” наследника HTML в прошлом десятилетии оказалась не очень эффективной, поэтому сейчас развитие XHTML приостановлено, и новая версия HTML, названная HTML5 [4], не связана жесткими требованиями, введенными в XHTML.

Любой XML-документ начинается со строки объявления, в которой указывается используемая версия XML:

```
<?xml version="1.1"?>
```

В настоящее время последней является версия 1.1 [5]. Если в документе используются символы, не входящие в таблицу ASCII, например русские буквы, в объявлении указывают используемую кодировку (атрибут **encoding**):

```
<?xml version="1.1" encoding="UTF-8"?>
```

Основная часть XML-документа заключается в корневой элемент, который можно назвать любым именем, например,

```
<book>
...
</book>
```

В него должны быть вложены все остальные элементы. Каждый из элементов может иметь атрибуты и свои вложенные элементы. Таким образом обеспечивается описание иерархической структуры документа.

При использовании XML особое внимание уделяется правильности (*валидности*) документов. Для проверки XML-документов используют так называемые *схемы* — XML-документы, которые хранятся в виде файлов с расширением **.xsd** (англ. ***X**ML **S**hema **D**efinition* — определение схемы XML). В них определяется, какие тэги могут присутствовать в документе, в каком порядке и количестве, как они вкладываются друг в друга, какие атрибуты допустимы, какие элементы являются обязательными и т.п. Схема позволяет также проверить типы используемых данных. Подробно с правилами составления схем можно познакомиться на сайте *www.w3schools.com* [6].

Язык XML наиболее удобен для представления иерархических структур данных. В частности, XML-файлы могут использоваться для хранения небольших баз данных, подключаемых к веб-страницам. Для работы с такими базами не нужно использовать серверные языки программирования (например, PHP), потому что все операции, в том числе сортировка и фильтрация, выполняются в браузере на компьютере пользователя. Это позволяет разгрузить веб-сервер и дает возможность выполнять все действия в локальном режиме.

Нужно отметить, что современные реляционные СУБД (*Microsoft SQL Server*, *Oracle*, *MySQL*) включают средства для работы с XML-документами и экспорта данных в формате XML. Это позволяет использовать как традиционную реляционную схему хранения данных, так и документоориентированные данные на основе XML в рамках одной базы данных. Например, при изменении реляционной базы данных можно обновлять XML-файл на веб-сервере, так что при просмотре связанной с ним веб-страницы будет всегда загружаться актуальная

[2] Поэтому часто говорят, что XML — это *метаязык*, то есть набор правил для составления языков разметки.

информация, причем это не потребует обращения к СУБД и существенного использования ресурсов сервера. Однако нужно учитывать, что этот подход неэффективен при работе с большими базами данных, в которых часто выполняются изменения и поиск по сложным запросам.

Поскольку в XML-файле хранятся только "чистые" данные, при выводе на экраны компьютеров пользователей они могут быть оформлены как угодно. Для представления данных используется язык расширяемых таблиц стилей XSL (англ. *eXtensible Stylesheet Language*). Правила преобразования данных, например, в другой XML-документ или HTML-страницу описываются в отдельном файле с расширением **.xsl** (который тоже представляет собой XML-документ).

В настоящей статье предлагается ознакомительный практикум по использованию XML-документов и таблиц стилей XSL в рамках изучения углубленного курса информатики в 11-м классе по учебнику [7]. Цель работы — познакомиться с языком XML и научиться выводить информацию из базы данных, хранящейся в формате XML, на веб-страницу. Предполагается, что учащиеся предварительно изучили основы языка HTML, умеют работать с каскадными таблицами стилей (CSS) и имеют представление о *JavaScript*.

Встречаем XML

1. Скопируйте на свой компьютер в отдельные папки файлы[3] **min.docx** (документ *Microsoft Office 2007*) и **min.odt** (документ *OpenOffice.org Writer*). Переименуйте файлы так, чтобы они имели расширение **.zip**, и распакуйте любым архиватором, который поддерживает этот метод сжатия.

Файлы с какими расширениями вы увидели после распаковки архивов? Просмотрите эти файлы в любом текстовом редакторе, изучите их структуру.

2. Скопируйте на свой компьютер файл **rect.svg** (векторный рисунок в формате SVG) и откройте его в браузере *Google Chrome, Mozilla Firefox* или *Opera*. Затем откройте этот файл в любом текстовом редакторе и изучите его структуру.

3. Скопируйте на свой компьютер файл **math.htm** (веб-страница с формулой в формате MathML) и откройте его в браузере *Mozilla Firefox* или *Opera*. Затем откройте этот файл в любом текстовом редакторе и изучите его структуру.

4. Найдите на компьютере файлы конфигурации с расширением **.xml**. В операционной системе *Microsoft Windows* используйте для поиска каталог *Program Files*, а в *Linux* — каталог **/usr**. Откройте один из найденных файлов в браузере или в любом текстовом редакторе и изучите его структуру.

[3] Все файлы, упоминаемые в статье, можно найти на диске в приложении к этому номеру.

База данных в формате XML

Работа может выполняться с помощью браузеров *Mozilla Firefox, Opera, Internet Explorer* или *Google Chrome*[4].

1. Скопируйте на свой компьютер файлы **europe.xml**, **europe2.xml** и **europe.xsl**. Первые два файла содержат базу данных по странам Европы (в разных форматах), а с помощью третьего мы будем задавать оформление данных на веб-странице.

2. Откройте файл **europe.xml** в браузере, используя команду меню *Файл — Открыть*, и изучите его структуру. Кроме того, файл **europe.xml** можно открыть в любом текстовом редакторе. В первой строке

```
<?xml version="1.0" encoding="UTF-8"?>
```

записаны используемая версия языка XML и кодировка (здесь — UTF-8). Далее расположены сведения о странах Европы в таком формате:

```
<СтраныЕвропы>
  <Страна>
    <Название>Греция</Название>
    <Столица>Афины</Столица>
    <Население>11</Население>
    <Площадь>132</Площадь>
  </Страна>
  ...
</СтраныЕвропы>
```

Корневой элемент в этом XML-файле называется *СтраныЕвропы*, в него вложено несколько элементов *Страна*, каждый из которых имеет вложенные элементы: *Название, Столица, Население* и *Площадь*.

Вывод данных на веб-страницу

1. В текстовом редакторе добавьте в файл вторую строку, которая подключает стилевой файл **europe.xsl**:

```
<?xml-stylesheet type='text/xsl'
                 href='europe.xsl'?>
```

Файлы с расширением XSL задают правила преобразования XML-документа в другой формат, например, в HTML. Браузер, открывая XML-файл с подключенной таблицей стилей, строит HTML-документ и выводит его на экран. Посмотрите, как выглядит теперь файл **europe.xml**, если открыть его в браузере.

2. Откройте файл **europe.xsl** в текстовом редакторе и изучите его. Найдите две строчки, которые выбирают информацию из базы данных:

```
<xsl:value-of select="Название"/>
<xsl:value-of select="Столица"/>
```

Здесь *Название* и *Столица* — это названия тэгов в XML-файле (полей базы данных). Обратите внимание, что это пустые элементы (без содержания), они заканчиваются знаками "**/>**", то есть сразу закрываются.

[4] Для того чтобы выполнить работу в *Google Chrome*, нужно запустить его с ключом **--allow-file-access-from-files**.

3. В XSL-файле можно использовать тэги языка HTML. Например, оформим данные в виде таблицы:

```
<table border="1">
<tr>
  <td><xsl:value-of select="Название"/></td>
  <td><xsl:value-of select="Столица"/></td>
</tr>
</table>
```

Самостоятельно добавьте в таблицу оставшиеся данные (население в миллионах человек и площадь страны в тысячах квадратных километров), а также заголовки столбцов (используйте тэг **TH**).

4. Если теперь открыть файл **europe.xml** в браузере *Mozilla Firefox*, то мы увидим данные только по первой стране, а в других браузерах — только заголовки столбцов таблицы[5].

Для того чтобы показать информацию по всем странам, используем цикл. Для этого в стилевой файл нужно добавить строчки, обозначающие начало и конец цикла:

```
<xsl:for-each select="//Страна">
...
</xsl:for-each>
```

В данном случае цикл выполняется по всем элементам *Страна*, то есть по всем странам. Двойной знак "**//**" (двойной слэш) показывает, что цикл применяется ко всем элементам с таким названием, независимо от уровня вложенности. Вместо этого можно было указать точный адрес элементов *Страна* так: "**СтраныЕвропы/Страна**" — это означает "все элементы *Страна* внутри элемента *СтраныЕвропы*".

Добавьте цикл в нужное место в XSL-файл и обновите файл в браузере. Вы должны увидеть в таблице информацию по всем странам.

Сортировка

1. Таблицу можно отсортировать по любому столбцу. Для этого в стилевой файл сразу после начала цикла добавим элемент

```
<xsl:sort order="ascending"
          select="Название"/>
```

Сортировка выполняется по полю *Название* в порядке возрастания (англ. *ascending*), для текстовых данных — в алфавитном порядке. Для сортировки по убыванию значение параметра **order** (англ. *порядок*) должно быть равно **descending** (англ. *убывающий*). Посмотрите, как теперь выглядит файл в браузере.

2. Измените порядок сортировки: сделайте сортировку по убыванию количества жителей страны. Подумайте, почему результат отличается от ожидаемого.

[5] Чтобы увидеть информацию по первой стране во всех браузерах, нужно перед названием полей поставить двойной слэш: например, написать "**//Название**" вместо "**Название**". Это означает "вывести значение элемента *Название* независимо от уровня вложенности".

3. Для того чтобы правильно выполнить сортировку по числовым данным, в команду сортировки нужно добавить параметр **data-type** (тип данных), который должен быть равен **number** (англ. *число*):

```
<xsl:sort ... data-type="number"/>
```

Внесите это изменение и проверьте правильность сортировки.

Выборка

1. С помощью стилевого файла можно задать условие отбора данных. Для этого применяется элемент **xsl:if**. Этот парный элемент (контейнер) нужно вставлять внутрь цикла после элемента сортировки:

```
<xsl:for-each ...>
<xsl:sort .../>
<xsl:if test="Население > 20">
...
</xsl:if>
</xsl:for-each>
```

Здесь отбор идет по значению поля *Население*. Выбираются только те страны, у которых население более 20 млн. человек. Проверьте работу этого элемента.

Можно также использовать другие операторы сравнения:

>= или **>=** больше или равно (англ. *greater than*);
< меньше (англ. *less than*);
<= меньше или равно.

Знаки "**>**" и "**<**" в коде веб-страниц заменяют, соответственно, на **>** и **<** для того чтобы браузер не путал их с угловыми скобками, ограничивающими тэги. Для знака "**<**" такая замена в стилевом XSL-файле обязательна.

2. Самостоятельно сделайте выборку по тем странам, площадь которых меньше чем 200 тысяч км^2, отсортируйте строки таблицы по возрастанию площади стран.

3. Вложенные элементы **xsl:if** дают эффект логической операции "И" (требуют выполнения двух условий одновременно). Добавьте в условие отбора еще одно ограничение: население больше чем 10 млн. человек. Проверьте результат в браузере.

Условное форматирование

1. С помощью элемента **xsl:if** можно сделать условное форматирование, то есть изменять свойства HTML-элементов. Например, чтобы изменить цвет для всех четных строк таблицы, после открывающего тэга **tr** внутри цикла нужно добавить строки:

```
<xsl:if test="position() mod 2 = 0">
  <xsl:attribute name="bgcolor">
   #E6E6FF</xsl:attribute>
</xsl:if>
```

Если позиция элемента, которая вычисляется функцией **position**, — четное число (остаток от деления на 2 равен 0), то для всей строки (предыдущего элемента **tr**) устанавливается свойство **bgcolor** (фоновый цвет), равный **#E6E6FF**.

Проверьте, как выглядит страница в браузере после этих изменений. Вы заметите, что строки раскрашены неправильно (фон не чередуется). Это произошло потому, что функция **position** использует *счетчик цикла*, который перебирает *все страны* в базе, хотя фактически выводятся данные только по тем странам, которые соответствуют двум вложенным условиям отбора.

2. Чтобы исправить ситуацию, сделаем так, чтобы цикл работал только для тех стран, которые нам нужны. Для этого уберем оба контейнера-условия **xsl:if** и изменим заголовок цикла, включив в него эти условия и связав их с помощью логической операции "**and**" (И):

```
<xsl:for-each
select="//Страна[Население > 10 and
                 Площадь &lt;200]">
```

Проверьте, что после этого и отбор данных, и раскраска строк работают правильно. В сложных условиях можно использовать также логическую операцию "**or**" (ИЛИ).

3. Измените условие так, чтобы отобрать страны, население которых меньше 20 млн. человек, а также все страны, площадь которых больше 100 тыс. км^2.

Подключение CSS-файлов

1. Создайте новый CSS-файл **europe.css** и подключите его к веб-странице. Для этого нужно ставить ссылку на него внутрь контейнера **head** в XSL-файле:

```
<link rel="stylesheet" href="europe.css"
type="text/css" />
```

Установите с помощью CSS серый цвет для ячеек таблицы, заданных тэгом **TH**. Проверьте результат в браузере.

2. Установите в CSS-файле для ячеек таблицы внутренние отступы (**padding**) 2 пикселя по вертикали и 5 пикселей по горизонтали.

3. Присвойте ячейкам, в которых выводится население и площадь стран, класс **number** и для него установите в CSS-файле выравнивание по правой границе. Окончательный результат должен выглядеть примерно так:

Страна	Столица	Население	Площадь
Нидерланды	Амстердам	15	42
Португалия	Лиссабон	11	92
Греция	Афины	11	132

Работа с атрибутами

1. Откройте файл **europe2.xml** в браузере, используя команду меню *Файл — Открыть*, и изучите его структуру. Как видите, эта база строится иначе — свойства стран заданы не как вложенные элементы, а как атрибуты пустого элемента *Страна*:

```
<Страна Название="Греция" Столица="Афины"
        Население="11" Площадь="132" />
```

Как ни странно, переход к такому формату базы данных потребует минимальных изменений в стилевом XSL-файле. Для обращения к атрибутам нужно перед названием атрибута просто поставить знак "@".

2. Скопируйте XSL-файл **europe.xsl**, назвав копию **europe2.xsl**. Перед всеми названиями атрибутов (которые раньше были названиями вложенных элементов) добавьте знак "@" (так вместо "**Название**" нужно написать "**@Название**").

3. Подключите к файлу **europe2.xml** таблицу стилей **europe2.xsl** и проверьте, что получается тот же самый результат, что и раньше.

На этом выполнение основной части работы закончено.

Рисунки и ссылки[6]

В XML-файле можно хранить пути к файлам изображений и ссылки на ресурсы Интернета. Как известно, адрес изображения на веб-странице указывается как значение атрибута **src** тэга **<img>**, а адрес перехода по гиперссылке — как значение атрибута **href** тэга **<a>**. Проблема состоит в том, что значение, взятое из XML-файла, нужно "встроить" внутрь тэга веб-страницы, передать как значение атрибута. Для этого используют специальный элемент языка XSL, который называется **xsl:attribute** (мы уже встречались с ним в разделе *Условное форматирование*).

Пусть в XML-файле хранятся данные об автомашинах в виде записей следующего формата:

```
<Авто Название="Ford Fusion"
      Фото="images/fordfsn.jpg"
 Ссылка="http://www.ford.com/cars/fusion/"/>
```

Для того чтобы вставить рисунок, адрес которого записан в базе данных как атрибут **Фото**, в стилевом XSL-файле используем следующий код:

```
<img>
  <xsl:attribute name="src">
    <xsl:value-of select="@Фото"/>
  </xsl:attribute>
</img>
```

Элемент **xsl:attribute** изменяет значение атрибута **src** тэга **<img>**, внутри которого он находится. Новое значение атрибута будет равно значению атрибута **Фото** записи из базы данных.

Аналогично меняется атрибут **href** у гиперссылки:

[6] Материал для любознательных. Не входит в основную часть работы.

```
<a>
  <xsl:attribute name="href">
    <xsl:value-of select="@Ссылка"/>
  </xsl:attribute>
  <xsl:value-of select="@Название"/>
</a>
```

В данном примере из записи базы данных берем два атрибута: **Название** (оно становится текстом ссылки) и **Ссылка** (адрес перехода).

Задание для самостоятельной работы[7]. Постройте базу данных в формате XML, содержащую рисунки и адреса перехода по гиперссылкам, и напишите стилевой файл для вывода этих данных на веб-страницу.

Использование JavaScript

Покажем, как можно использовать язык программирования *JavaScript* для управления выводом данных из XML-файла на веб-страницу. Наша задача — сделать заголовки столбцов гиперссылками, щелчок по которым сортирует данные по выбранному столбцу.

Как вы знаете, сортировка данных задается в таблице стилей. Поэтому задача сводится к тому, чтобы после щелчка по гиперссылке динамически *заменить таблицу стилей* текущей веб-страницы. При этом нельзя, как мы делали раньше, жестко связывать XML-файл со стилевым файлом, поэтому ссылку на стилевой файл в начале XML-документа нужно убрать.

Основным файлом будет HTML-страница, которую мы назовем **europe.htm**. После загрузки этой страницы на нее будут динамически подгружаться данные из XML-файла и некоторая таблица стилей (начальный вариант, без сортировки). После щелчка по гиперссылке таблица стилей должна заменяться. Это означает, что придется создать несколько стилевых таблиц с расширением **.xsl**, которые будут отличаться только порядком сортировки данных.

1. Скопируйте файлы **europe.css**, **europe2.xml** и **europe2.xsl** в отдельный каталог (например, с именем **JS**).

2. Создайте веб-страницу **europe.htm** следующего содержания:

```
<html>
<head>
<link rel="stylesheet"
 href="europe.css" type="text/css" />
</head>
<body>
  <div id="data">
  </div>
</body>
</html>
```

На этой странице подключается стилевой файл **europe.css**, и в теле документа размещен блок с идентификатором **data**, в который мы будем загружать данные из XML-файла.

3. Скопируйте в каталог JS файл **loadxml.js**, содержащий функции для динамической загрузки XML-документа и стилевой XSL-таблицы (см. Приложение). Подключите этот файл к веб-странице с помощью тэга **script**:

```
<script type="text/javascript"
        src="loadxml.js"></script>
```

4. Удалите из файла **europe2.xml** строку, в которой подключается стилевой файл **europe2.xsl**.

5. Удалите из файла **europe2.xsl** строки, содержащие HTML-тэги (теперь они находятся в теле HTML-страницы), а также команды для сортировки и поиска.

6. Добавьте в тело веб-страницы скрипт с функцией **init**, которая загружает XML-документ **europe2.xml** и таблицу стилей **europe2.xsl** в блок с идентификатором **data**:

```
<script type="text/javascript">
function init() {
  source = loadXMLDoc("europe2.xml");
  loadStyle("europe2.xsl", "data");
  return true;
}
</script>
```

— и вызовите функцию **init** в обработчике события **onLoad** (он срабатывает сразу после загрузки страницы):

```
<body onLoad="init();">
```

Откройте веб-страницу **europe.htm** в браузере и убедитесь, что выводятся данные по всем странам.

7. Теперь остается организовать сортировку. Заголовки столбцов таблицы находятся в стилевом файле **europe2.xsl**, их нужно сделать гиперссылками, так чтобы при щелчке по ним вызывались функции, выполняющие сортировку (подгружающие другую таблицу стилей). Например, вместо строки

```
<th>Страна</th>
```

нужно записать

```
<th>
<a href="#" onClick=
   "loadStyle('europe2n.xsl','data');"
 >Страна</a>
</th>
```

Вызов функции **loadStyle** в обработчике события **onClick** (при щелчке мышью) загружает таблицу стилей **europe2n.xsl**, которая должна выполнить сортировку по названию страны.

Тем же способом превратите все заголовки столбцов таблицы в гиперссылки. Конечно, имена таблиц стилей должны быть разными. Например, последняя буква названия XSL-файла может обозначать порядок сортировки: **n** (от англ. *name*) — по названию страны; **c** (от англ. *capital*) — по названию столицы; **p** (от англ. *people*) — по числу жителей; **a** (от англ. *area*) — по площади.

8. Скопируйте основную таблицу стилей **europe2.xsl** четыре раза, получив файлы с теми именами, которые указаны в гиперссылках. В каждый из них добавьте нужный вариант сортировки.

9. Откройте веб-страницу **europe.htm** в браузере и убедитесь, что все варианты сортировки работают правильно. Особое внимание обратите на сортировку по числовым данным (количеству жителей и площади).

[7] Примеры выполнения этого задания можно найти на диске в приложении к этому номеру.

Выводы

XML является международным стандартом для передачи данных между приложениями в текстовой форме. Это очень удобно при использовании протокола HTTP, который оптимизирован для обмена текстовыми сообщениями и поддерживает их эффективное сжатие без потерь.

Хотя язык XML был разработан прежде всего для облегчения машинной обработки данных, документы в формате XML понятны для человека, их можно просматривать в любом текстовом редакторе. Основные достоинства XML проявляются в задачах хранения и передачи данных иерархической структуры.

В то же время, эта технология обладает рядом недостатков. Во-первых, использование XML приводит к увеличению объема файла в сравнении с двоичными форматами, особенно если содержание полей данных невелико, потому что при этом основную часть XML-документа составляет разметка (тэги), а не полезная информация. Во-вторых, существует неоднозначность моделирования: данные о каком-то объекте можно хранить как элементы, вложенные в другой элемент, а можно — как атрибуты этого элемента (вспомните два варианта базы данных по странам Европы, которые использовались в практикуме). Поэтому в последнее время приобретают популярность альтернативные форматы передачи данных, например, YAML и JSON. Несмотря на это, XML — это весьма полезный инструмент, который широко используется во многих задачах, связанных с компьютерными технологиями.

Литература

1. ГОСТ Р ИСО/МЭК 26300-2010. Госстандарт России (01.06.2011). Информационная технология. Формат Open Document для офисных приложений (OpenDocument) v 1.0: [Электронный ресурс]: *http://protect.gost.ru/document.aspx?control=7&baseC=6&page=3&month=5&year=2011&search=&id=177075.*
2. Office Open XML File Formats: [Электронный ресурс]: *http://www.ecma-international.org/publications/standards/Ecma-376.htm.*
3. XHTML 1.1 — Module-based XHTML: [Электронный ресурс]: *http://www.w3.org/TR/xhtml11/.*
4. HTML Working Group: [Электронный ресурс]: *http://www.w3.org/html/wg/.*
5. Extensible Markup Language (XML) 1.1: [Электронный ресурс]: *http://www.w3.org/TR/xml11/.*
6. XML Schema Tutorial: [Электронный ресурс]: *http://www.w3schools.com/schema/default.asp.*
7. *Поляков К.Ю., Еремин Е.А.* Информатика. Углубленный уровень: учебник для 11-го класса: в 2 ч. М.: Бином, 2013.

ПРИЛОЖЕНИЕ

loadxml.js:

```
// Функции для загрузки XML-документов и стилевых таблиц
var source; // XML-файл (база данных)
var style;  // XSL-файл (стилевая таблица)
//---------------------------------------
// loadXMLDoc – загрузить XML-документ
//---------------------------------------
function loadXMLDoc(dname) {
  if (window.ActiveXObject) { // для браузера IE
    xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
    xmlDoc.async=false;
    xmlDoc.load(dname);
    return xmlDoc;
    }
  // для браузеров, кроме IE
  xhttp = new XMLHttpRequest();
  xhttp.open("GET",dname,false);
  xhttp.send("");
  return xhttp.responseXML;
}
//---------------------------------------
// loadStyle – загрузить новую стилевую таблицу
//---------------------------------------
function loadStyle ( styleName, id ) {
  style  = loadXMLDoc( styleName );
  if (window.ActiveXObject) { // для браузера IE
    res = source.transformNode(style);
    document.getElementById(id).innerHTML = res;
    }
  else if (document.implementation && document.implementation.createDocument)
    { // для браузеров, кроме IE
    xsltProcessor=new XSLTProcessor();
    xsltProcessor.importStylesheet(style);
    resultDocument = xsltProcessor.transformToFragment(source,document);
    res = document.getElementById(id);
    while (res.hasChildNodes())
      res.removeChild(res.lastChild);
    res.appendChild(resultDocument);
    }
}
```

Теги и холст (svg vs canvas). Декларативная и динамическая веб-графика

И.Б. Государев,
РГПУ им. А.И. Герцена, доцент кафедры информационных и коммуникационных технологий, кандидат педагогических наук, учитель информатики ГОУ СОШ № 328 Невского р-на, Санкт-Петербург

Краткая предыстория

Использованный нами в заголовке термин “веб-графика”, вообще говоря, довольно широк. Это прежде всего файлы в графических форматах, таких, как JPEG, PNG, GIF, которые вставляются в тело веб-страницы с помощью тега `img`. Такие файлы можно редактировать только специализированными редакторами, то есть при необходимости изменить что-то в изображении нужно будет “перезаливать” обновленный файл обратно на хостинг.

Сегодня решить эту проблему очень легко. Размещаем файл в сервисе типа Flickr, помещаем в теге `img` абсолютную ссылку на него, редактируем с помощью облачного редактора [1] — все изменения будут сразу же видны на веб-странице.

Однако изображение, о котором идет речь, растровое. А это значит, что при масштабировании веб-страницы четкость его неизбежно потеряется. Кроме того, файл может не загрузиться по самым разным причинам.

Весьма удобно, если изображение создается внутри веб-страницы, редактируется вместе с остальным контентом, масштабируется без потери четкости и не поглощает лишнего трафика.

Создателями браузера Microsoft Internet Explorer была предпринята попытка достичь этого удобства в виде встроенного языка разметки VML. Если вы в редакторе Microsoft Word вставите в текст автофигуру (например, прямоугольник), а затем сохраните документ как веб-страницу и рассмотрите его исходный код, то среди огромного количества запутанных тегов вы увидите примерно следующее:

```
<!--[if gte vml 1]>
<v:rect id="_x041f"
o:spid="_x0000_s1026" style='position:absolute;margin-left:55.9pt;
```

```
 margin-top:16.5pt;width:64.8pt;height:64.8pt;z-index:251659264;
visibility:visible; mso-wrap-style:square;.....'
 o:gfxdata= "UEsDBBQAB.....VAUAAAAA" fillcolor="#f60"
strokecolor="#4579b8 [3044]"><w:wrap type="through"/>
</v:rect>
<![endif]-->
```

Этот сильно сокращенный блок отображает в окне MS IE прямоугольник (при сохранении, ради других браузеров, создается еще и растровое изображение в отдельной папке).

Наши дни

В современных версиях всех популярных браузеров внедрена поддержка языка разметки векторной графики SVG (***S**calable **V**ector **G**raphics*). Мы будем называть такую графику декларативной, так как изображения в ней задаются объявлениями вида `<Прямоуг ширина=100 высота=50></Прямоуг>`.

Такие изображения составляются из “примитивов” — эллипсов, многоугольников, элементов кривых. Для создания изображения можно воспользоваться любым доступным векторным редактором — например, Inkscape — для рисования изображения и его сохранения в формате SVG. Но, учитывая наш опыт общения с онлайновыми и облачными решениями, это логично сделать в каком-либо онлайновом приложении, например, *svg-edit.googlecode.com*.

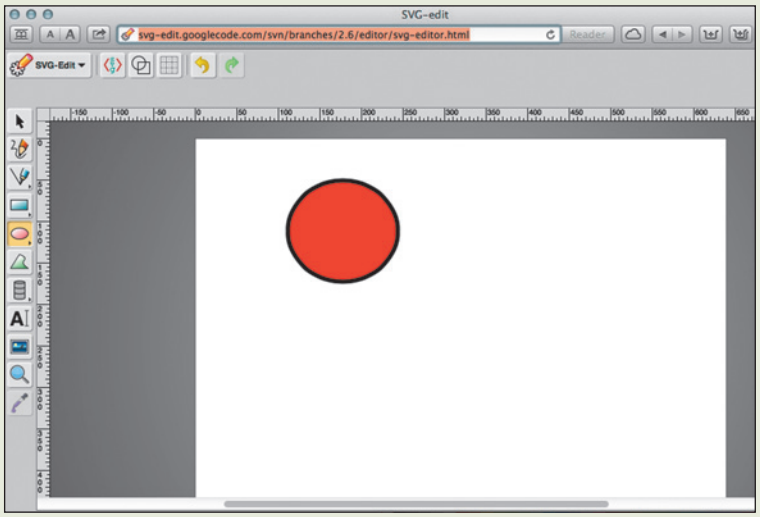


Рис. 1. Интерфейс онлайн-редактора векторных изображений

Для приведенного выше изображения редактор сгенерировал следующий код:

```
<svg width="640" height="480" xmlns="http://www.w3.org/2000/svg">
 <!-- Created with SVG-edit - http://svg-edit.googlecode.com/ -->
 <g>
  <title>Layer 1</title>
  <ellipse ry="61" rx="66" id="svg_1" cy="112" cx="178"
   stroke-width="5" stroke="#000000" fill="#FF0000"/>
 </g>
</svg>
```

Этот код можно непосредственно вставить в HTML-код веб-страницы. Разумеется, в рамках “облачной парадигмы” мы рекомендуем сделать это в онлайновом редакторе разметки. Их существует великое множество, и в числе самых интересных и многофункциональных можно порекомендовать *jsfiddle.net*. Однако автор данной статьи создал и собственный небольшой редактор для быстрого просмотра и редактирования HTML-кода онлайн:

dist-learn.spb.ru/html?5.

Протестируйте этот код в различных браузерах. При написании статьи использовались браузеры Safari 6, Chrome 27, Opera 12, Firefox 21 и Internet Explorer 10.

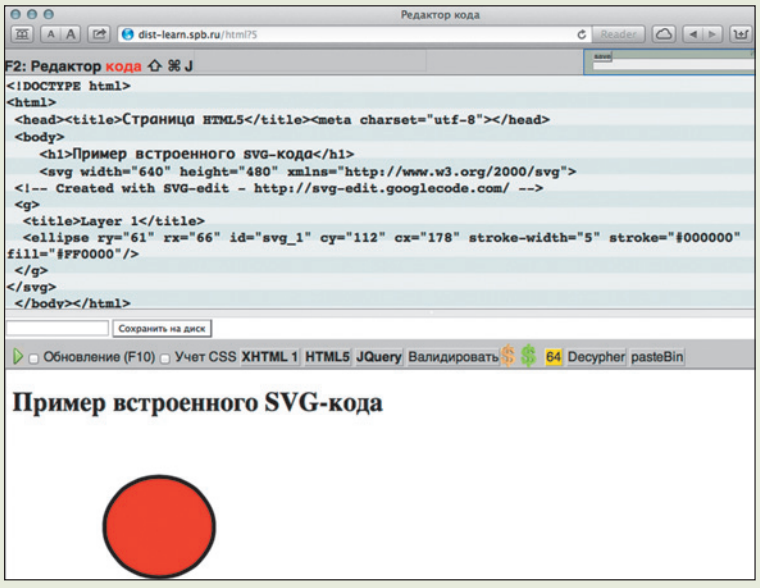


Рис. 2. Отображение встроенного SVG-кода в веб-странице

Открыв исходный код различных графических приложений (например, графики в электронных таблицах Google Drive), мы увидим сгенерированный код SVG.

Если рисовать в редакторе сложные изображения, то движок будет аппроксимировать контуры кусочками кривых, и код получится очень громоздкий и плохо поддающийся осмыслению. Это и неудивительно: векторные инструменты приспособлены в первую очередь для легкозадаваемых уравнениями фигур.

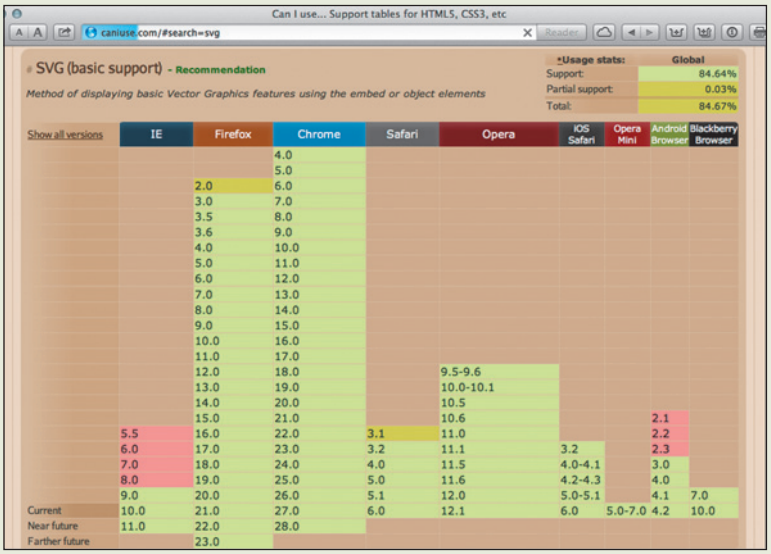


Рис. 3. Поддержка SVG в браузерах

Современные веб-приложения характеризуются динамичностью интерфейса. Содержимое веб-страницы — а значит, HTML-разметка и DOM — модифицируются “на лету”, в ответ на действия пользователя. Это может быть результатом работы Javascript-сценариев или серверной стороны, которая генерирует разметку и отсылает ее клиенту (браузеру). Вполне можно генерировать SVG-теги динамически, для этого существуют целые библиотеки. Одна из самых известных — Raphaël (*raphaeljs.com*):

```
var canvas = Raphael(10, 50, 300, 200); //создать прямоугольный холст 300x200
var circle = canvas.circle(50, 40, 10); //создать круг радиуса 10
circle.attr("fill", "#f00"); //залить созданный круг красным
circle.attr("stroke", "#ff0");//сделать ободок круга желтым
```

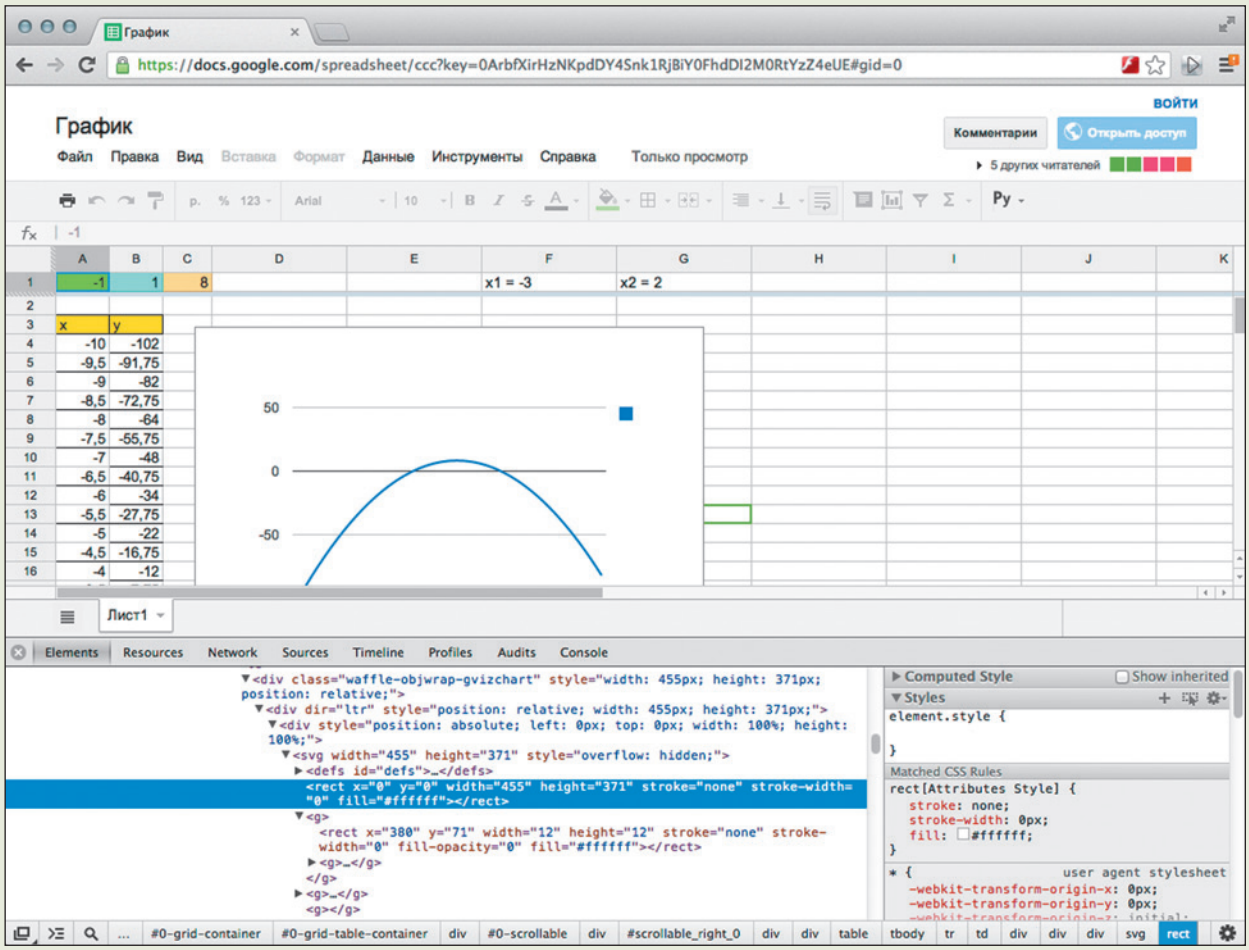


Рис. 4. Отображение графиков в Google Drive с помощью SVG

Эти библиотеки до недавнего времени были единственным способом обеспечить кроссбраузерность при выводе графики, но на сегодняшний день ситуация, очевидно, изменилась.

Приведенный выше фрагмент кода демонстрирует типичный подход в программировании с помощью Javascript-библиотек: создается объект, с которым далее производятся манипуляции и через который создаются прочие объекты.

Холст

Переменная *canvas* в вышеприведенном фрагменте хранит объект, создаваемый сложным Javascript-кодом. Но уже достаточно давно разработан и используется гораздо более простой и удобный способ вывода графики на веб-страницу — в специально отведенную для этого область с таким же названием: canvas. Подход к построению изображения напоминает работу с DirectX или OpenGL.

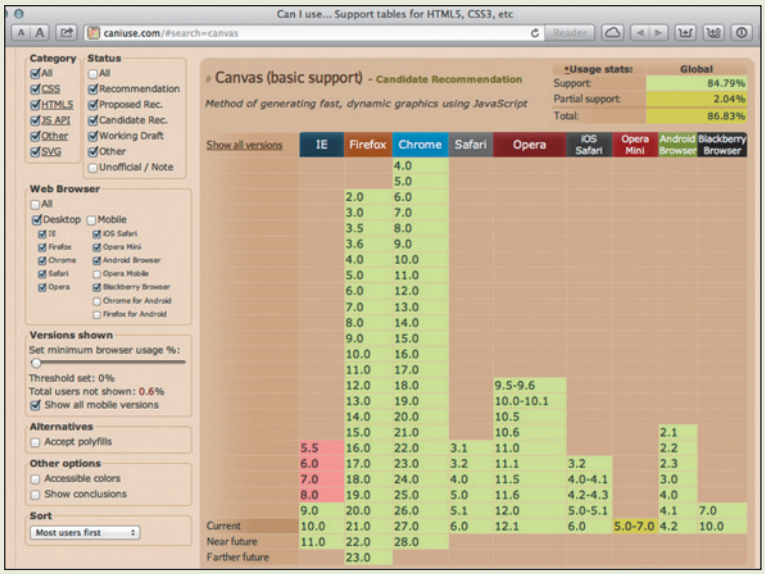


Рис. 5. Поддержка canvas в браузерах

Создание canvas и построение изображения происходит в два этапа:

- Размещение тега canvas в теле веб-страницы;
- Исполнение специальных команд, являющихся методами объекта canvas, в рамках Javascript-сценария.

Рассмотрим простой пример. В нем и следующих примерах мы будем исходить из базовой правильной HTML5-разметки вида:

```
<!DOCTYPE html>
<html>
 <head>
  <title>HTML5+Canvas</title>
  <meta charset="utf-8">
  <script>
     var canvas = document.querySelector('#myCanvas');
     var ctx = canvas.getContext('2d');
        //далее инструкции для управления холстом
  </script>
 </head>
 <body>
    <h1>HTML5+Canvas</h1>
    <canvas id='myCanvas' width='400' height='200'></canvas>
 </body>
</html>
```

Чтобы каждый раз не воспроизводить одинаковые части разметки, мы условимся, что секция сценария будет находиться перед закрытием секции head (перед тегом `</head>`), а холст — после главного заголовка веб-страницы (холстов, конечно, может быть более одного). Соответственно, в примерах будет указываться только содержимое секции сценария; кроме того, можно менять параметры ширины и высоты холста. Первые две команды в секции сценария — это создание двумерного графического контекста. Контекст рисования — привычное понятие для программируемой графики, он включает параметры, влияющие на рисуемые объекты. Мы задаем значение параметра (например, толщину пера), и он будет действовать на все рисуемое далее в коде, пока не будет задано новое значение.

В холсте действует прямоугольная система координат с началом координат в левом верхнем углу холста; ось абсцисс направлена вправо, а ось ординат — вниз.

Итак, в качестве первого упражнения мы обведем холст рамкой. Для этого нужно программно получать его ширину и высоту.

Простейший путь построения рамки — задать свойства контекста и написать команду рисования прямоугольника.

```
ctx.lineWidth = 2;
ctx.strokeStyle = 'blue';
ctx.strokeRect(1, 1, canvas.width-2, canvas.height-2);
```

Здесь действует линейный пошаговый подход: команды исполняются по мере того, как интерпретатор встречает их в тексте сценария. Рамка рисуется линией толщиной в два пикселя.

Отметим, что прямоугольник можно сначала создать в контексте, а затем "прорисовать" с помощью `stroke()` или `fill()`:

```
context.rect(1, 1, canvas.width-2, canvas.height-2);
context.stroke();
```

Если необходимо нарисовать круг или дугу, то используется именно эта методика: сначала создаем, затем прорисовываем или же заполняем заливкой. Методы `stroke()` и `fill()` рисуют так называемые "пути" (`path`), и дуга относится именно к этому классу объектов.

Для рисования дуг окружностей используется метод `arc(x,y,R, start, end, CCW)`.

Первые два числа в нем — координаты центра, затем начальный и конечный углы, и булево значение, указывающее, рисовать ли против часовой стрелки.

Рассмотрите следующий код:

```
ctx.fillStyle = "red";
ctx.arc(50, 50, 10, 0, 2 * Math.PI, false);
ctx.fill();
ctx.strokeStyle = 'blue';
ctx.rect(1, 1, canvas.width-2, canvas.height-2);
ctx.stroke();
```

Здесь сначала рисуется круг (начальный угол — 0 градусов, конечный — 360), а затем прямоугольник. Обратите внимание на то, что у круга будет синяя обводка, хотя на первый взгляд она для него не задавалась. Дело в том, что метод `stroke()` действует на то, что находится в контексте, на пути, а в нашей текущей ситуации в контексте находятся два пути: круг и прямоугольник. Или можно сказать, что они образуют один прерывный путь.

Попробуем теперь поменять эти два блока местами. Последним будет работать метод `fill()`, который зальет красным фоном прямоугольник, а затем круг, но так как прямоугольник больше, то круга в итоге мы не увидим. Как же решить эту проблему?

Для отделения одних объектов ("путей") от других используется метод `beginPath()` — если поместить его между этими блоками, то их можно переставлять местами как угодно, результат будет один и тот же: красный круг без обводки и незалитый синий прямоугольник.

Отметим, что в некоторых случаях можно сначала создать несколько путей, а потом одной командой `stroke()` прорисовать их все. Например, следующие команды рисуют контур холста, треугольник и окружность:

```
ctx.arc(60, 50, 10, 0, 2 * Math.PI, false);
ctx.rect(1, 1, canvas.width-2, canvas.height-2);
ctx.moveTo(20,20);
ctx.lineTo(20,100);
ctx.lineTo(70,100);
ctx.lineTo(20,20);
ctx.stroke();
```

Однако если первые две команды поменять местами, то возникнет лишняя линия, артефакт, из-за того, как рисуется окружность. Корректнее всего было бы каждую такую фигуру начинать командой `beginPath()` и заканчивать командой `stroke()`.

Треугольник в примере рисуется так: "точка начала пути" помещается в (20,20), затем рисуется левый катет, затем нижний, затем гипотенуза. Последнюю команду — `lineTo (20,20)` — можно было бы заменить на команду `closePath()`, которая соединяет последнюю точку с первой.

Из других методов работы с путями отметим `bezierCurveTo()` и `quadraticCurveTo()`, которые рисуют, соответственно, кубическую и квадратичную кривую Безье; `isPointInPath()`, которая позволяет определить, находится ли заданная точка внутри текущего пути, а также команды для аффинных преобразований, включая частные случаи масштабирования `scale()`, поворота `rotate()`, переноса `translate()` и общее преобразование по матрице `transform(a,b,c,d,e,f)`.

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & c & e \\ b & d & f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Рис. 6. Матрица аффинного преобразования

Например, с помощью матрицы можно из прямоугольника путем скоса (skew) получить параллелограмм:

```
ctx.fillStyle="yellow";
ctx.fillRect(0,0,250,100)
ctx.setTransform(1,0,1,1,40,50);
ctx.fillStyle="red";
ctx.fillRect(0,0,250,100);
```

Для работы с текстом используются методы `fillText()` и `strokeText()`; можно создавать тени и градиенты, регулировать прозрачность, импортировать изображения. Более полный список методов и свойств холста и контекста можно найти в [3] и [2].

Очень удобным средством обучения работе с canvas являются интерактивные демонстрационные примеры; например, по адресу *http://www.html5canvastutorials.com/advanced/html5-canvas-custom-transform/* можно найти интерактивный пример матричных преобразований, а по адресу *http://www.html5canvastutorials.com/labs/html5-canvas-modify-curves-with-anchor-points-using-kineticjs/* располагается интерактивный пример применения кривых Безье. Интерактивность никак не заложена в canvas — она реализуется обычными для Javascript способами, через обработчики событий.

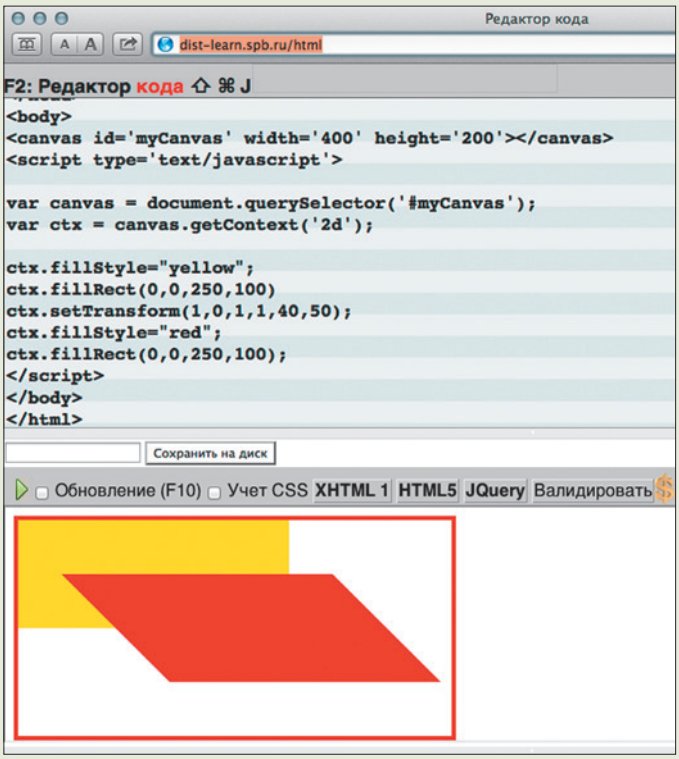


Рис. 7. Результат применения аффинного преобразования “скос”

В качестве самостоятельных заданий можно предложить для выполнения следующие:

- Напишите команды, рисующие круг в центре холста (независимо от конкретных его размеров);
- Напишите сценарий, который создает круг радиуса 1 с центром в точке, где имел место щелчок левой кнопкой мыши;
- Создайте на основе canvas приложение, рисующее диаграмму (график) по нескольким числам, введенным в поля input.

Технология, о которой шла речь в данной статье, продолжает развиваться. Трудно предсказать, как именно это будет происходить, но очевидно одно: с получением поддержки во всех лидирующих браузерах веб-графика на основе canvas выходит на качественно новый уровень существования.

Литературные и интернет-источники

1. *Государев И.Б.* Информатика в облаках // Информатика, № 6, 2013.
2. Шпаргалка по холсту [Электронный ресурс] URL: *http://www.nihilogic.dk/labs/canvas_sheet/HTML5_Canvas_Cheat_Sheet.pdf* (дата обращения: 01.05.2013).
3. W3schools.com HTML Canvas Reference [Электронный ресурс] URL: *http://www.w3schools.com/tags/ref_canvas.asp* (дата обращения: 01.05.2013).

В параллельных вселенных

И.А. Сукин,
г. Переславль-Залесский

Введение

Со времени появления первых вычислительных устройств и алгоритмизации как научных, так и повседневных задач прошло не слишком много лет, но уже появились серьезные концептуальные проблемы. Одна из таких проблем или даже, скорее, неудобств, если говорить кратко, состоит в том, что в классической модели вычислений мы рассматриваем алгоритм как некую последовательность действий, следующих друг за другом. От такого предположения отталкивалась и вычислительная техника на заре своего развития, что повлекло за собой создание компьютеров, полностью воссоздающих подобную модель. Тем не менее, если взглянуть на немалое подмножество задач, можно увидеть, что, по сути, нет нужды в том, чтобы одни части алгоритма ожидали того момента, когда завершатся другие. Иначе говоря, некоторые ветви алгоритма могут выполняться *параллельно* друг другу. Осознание этого факта послужило толчком к созданию устройств, которые соответствуют этой новой парадигме. Самыми известными “простому человеку” являются разрекламированные многоядерные процессоры. Существует также немалое число других подобных машин.

Более того, нужно сказать, что некий вид *параллелизма* появился задолго до того, как возникла реальная возможность реализации этой идеи, что называется, в железе. Операционные системы уже давно поддерживали псевдомногозадачность, позволяя нескольким алгоритмам, потокам кода, выполняться на одном физическом устройстве как бы одновременно. Ниже мы рассмотрим большую часть вариантов и приемов параллелизма, обеспеченного как реальными устройствами, так и виртуальными механизмами ОС.

Зачем нужен параллелизм?

Зачем нужен параллелизм простому человеку, пользователю компьютера? Сейчас такие вещи уже воспринимаются как сами собой разумеющиеся,

но на принципах параллельности основаны все современные многопользовательские и многозадачные операционные системы. Одновременная (или кажущаяся одновременной) работа офисных текстовых редакторов, браузеров, клиентов для быстрого обмена сообщениями и других прикладных программ обеспечивается встроенными в ОС механизмами поддержки параллелизма.

Однако даже если принять точку зрения Вирта, высказанную им в [2], о том, что в реальной пользовательской (домашней) операционной системе многозадачность не очень-то и нужна, потому что сложно представить человека, которому действительно нужно работать с несколькими задачами одно и то же время (фоновые задачи, которые в данный момент неактивны, можно, например, сохранять на диск в виде слепков-дампов памяти), параллелизм был бы полезен для системных нужд:

- Обработчики аппаратных прерываний желательно выполнять в одно и то же время с пользовательским кодом, поскольку часто нет никакого смысла простаивать и ждать ответа от устройства;
- Системные службы тоже удобно было бы запускать параллельно с прикладными задачами.

Кроме всего вышесказанного, большинство сколь-нибудь важных инженерных и научных задач очень удобно рассматривать с точки зрения распараллеливания. Отсюда можно сделать вывод, что параллелизм в целом полезен и нужен.

Виды параллелизма

Существует два основных вида параллельности: параллелизм кода и параллелизм данных. В первом случае на нескольких независимых вычислителях над одними и теми же (необязательно) данными выполняются разные действия, алгоритмы, программы. Во втором варианте на разных вычислителях выполняется одна и та же программа, но над разными данными, чаще всего — над частями одного большого набора данных.

В дополнение к этим двум основным видам можно выделить еще два дополнительных, менее абстрактных: параллелизм на уровне битов и параллелизм на уровне инструкций. К первому типу относятся VLIW (***V**ery **L**ong **I**nstruction **W**ord*) машины, в которых машинное слово имеет большой размер и вмещает в себя одновременно несколько команд, которые процессор может выполнять параллельно. Ко второму случаю относятся, как правило, суперскалярные архитектуры с несколькими конвейерами выборки машинных команд (а сейчас это большинство процессоров), в которых несколько последовательно идущих в потоке кода команд могут быть распределены процессором на разные конвейеры внутри себя и выполнены, соответственно, одновременно. Подвидом такого параллелизма, называемым еще параллелизмом уровня памяти, является архитектура, позволяющая в одно и то же время обрабатывать несколько операций с памятью, требующих ожидания, таких, как промах кэша или ассоциативного буфера. Здесь мы не будем вдаваться в такие низкоуровневые варианты параллелизма и ограничимся только вышеприведенным обзором.

Симметричность

Необходимо также отметить разделение вычислительных архитектур на симметричные и асимметричные. В симметричных (SMP) системах каждый вычислитель имеет одинаковый приоритет и может обрабатывать любой поток кода. В асимметричных архитектурах отдельные вычислители приспособлены для работы с определенными заданными потоками кода и данных. К примеру, если у нас есть два процессора, в асимметричной модели мы можем выполнять весь системный код на одном из них, а весь пользовательский — на другом.

Конкуренция и параллелизм

Можно очень легко запутаться, смешивая такие термины, как “конкуренция” и “параллелизм”. Под “конкуренцией” обычно подразумевается методика разработки и выполнения программ, при использовании которой две и больше программ, вычислителей, обрабатывают одни и те же ресурсы, причем этот процесс имеет место не обязательно на физически параллельной системе. В данной статье я практически не буду употреблять термин “конкуренция”, а под “параллелизмом” буду иметь в виду как реальный, так и мнимый параллелизм.

Классификация Флинна

Одним из общепринятых подразделений видов вычислений является классификация Флинна. В ней выделяются четыре типа вычислительных парадигм: SISD, SIMD, MISD, MIMD.

SISD — ***S**ingle **I**nstruction **S**ingle **D**ata* — соответствует обычному последовательному вычислению. Процессор в каждый момент времени работает с одним потоком кода и одним потоком данных, при этом вычисления полностью детерминированы. С некоторой оговоркой с этим подходом было спроектировано подавляющее большинство вычислительной техники до недавних времен.

SIMD — ***S**ingle **I**nstruction **M**ultiple **D**ata*. В рамках этой парадигмы вычислители работают в каждый момент времени над одной и той же командой, но разными данными. Таким образом, этот подход представляет параллелизм данных. Важнейшей областью применения SIMD-устройств является обработка больших объемов однородных данных, таких, как аудиозаписи, изображения, видео и другая мультимедийная информация. Реальные вычислители, реализующие данную модель, обычно бывают двух видов:

• Массивы процессоров;

• Векторные процессоры, или векторные арифметико-логические устройства, такие, например, как блок MMX/SSE в x86, AltiVec в PowerPC, и другие.

MISD — ***M**ultiple **I**nstruction **S**ingle **D**ata* — представляет собой нечто вроде конвейера, на котором разные вычислители выполняют разные операции над одними и теми же данными. В качестве примера такого вычислителя можно привести шейдерный конвейер в современных графических процессорах.

MIMD — ***M**ultiple **I**nstruction **M**ultiple **D**ata* — наиболее сложный, гибкий и мощный вариант реализации параллельной вычислительной системы, в которой в каждый момент времени каждый вычислитель может выполнять разные операции над разными данными. При этом вычисления могут быть как синхронными, так и асинхронными, как детерминированными, так и недетерминированными. Подавляющее большинство современных суперкомпьютеров, вычислительных кластеров, grid-систем используют подход MIMD.

Виды памяти

В этом разделе я опишу различные варианты организации работы с памятью в параллельных вычислениях.

Разделяемая память

Наиболее прямолинейным вариантом использования памяти несколькими одновременно работающими вычислителями является ее разделение между собой: все вычислители работают с одной и той же областью памяти, в одном и том же адресном пространстве. Существуют два типа организации работы с такой памятью:

• UMA — ***U**niform **M**emory **A**ccess*, когда вся память доступна каждому процессору “одинаково”, в том смысле, что доступ к ней осуществляется за время, равное для всех вычислителей и не зависящее от физического расположения памяти относительно процессора;

• NUMA — ***N**on-**U**niform **M**emory **A**ccess*, когда время доступа к ячейке памяти для каждого процессора определяется физическим расположением этой ячейки относительно вычислителя, существуют “близкие” и “далекие” области памяти. Простейшим, несколько утрированным, примером может служить кэш, или даже регистры, доступ к которым осуществляется за время несравнимо меньшее, чем то, которое требуется на доступ к обычной ячейке ОЗУ. Существуют, однако, и более сложные варианты реализации NUMA-систем.

Распределенная память

В моделях с распределенной памятью каждый вычислитель имеет свой “кусочек” памяти и не может обращаться к аналогичным “кусочкам” других процессоров. Это позволяет избавиться от значительного числа проблем, о которых я скажу чуть ниже, но предъявляет дополнительные требования к проектированию программ, которые при этом должны передавать друг другу данные с помощью механизма *пересылки сообщений*. Классические императивные процедурные языки программирования весьма плохо подходят для реализации этой техники, но объектная ориентированность и так называемая “акторная модель” дают возможность выразить *пересылку сообщений* естественным для соответствующих языков путем. Обо всем этом я буду более подробно говорить ниже.

Гибридные варианты

Большинство реально существующих в настоящее время систем параллельных вычислений используют как разделяемую, так и распределенную память. Представим следующий компьютер:

• Каждый узел является симметричным вычислителем с разделяемой памятью;

• Разные узлы соединены друг с другом по сети и не имеют прямого доступа к памяти других узлов.

По такой схеме устроены типичные суперкомпьютеры, и скорее всего подобная архитектура будет преобладать и в дальнейшем.

О проблемах

Хотелось бы уже в самом начале разговора отметить, что параллельные вычисления совсем не являются того или иного рода “серебряной пулей” разработки ПО и имеют достаточно серьезные ограничения и чуть менее серьезные технические проблемы.

Важнейшим фундаментальным ограничением параллельной обработки в реальном мире является тот факт, что количество вычислителей всегда ограничено и зачастую фиксировано в каких-то рамках, при том, что большинство распараллеливаемых алгоритмов, вообще говоря, в идеале требуют использования такого числа процессоров, какое только может потребоваться в динамике, лучше всего — бесконечного. В качестве примера здесь возьмем параллельную рекурсию при вычислении заданного числа Фибоначчи по определению. Как известно, N-е число Фибоначчи является суммой $(N-1)$-го и $(N-2)$-го. Здесь в голову сразу приходит[1] мысль вычислять $(N-1)$-е и $(N-2)$-е число на разных процессорах, поскольку эти вычисления не зависят от результатов друг друга. Вычисление $(N-1)$-го числа потребует вычисления $(N-2)$-го и $(N-3)$-го и так далее (здесь не будем обращать внимание на то, что $(N-2)$-е число уже где-то вычислено, и рассмотрим эти вы-

[1] Здесь нужно отметить, что этот вариант вычисления чисел Фибоначчи является крайне неэффективным и приведен лишь в качестве иллюстрации.

числения как независимые). Очевидно, что для заданного *N* можно точно сказать, сколько потребуется вычислительных узлов, но если у нас имеется параллельная подпрограмма вычисления чисел Фибоначчи, мы не сможем предсказать, какое *N* будет введено пользователем, что, следовательно, потребует потенциально бесконечного числа процессоров. Решение этой проблемы обычно лежит в перестройке алгоритма к более эффективному виду, но в некоторых случаях приводит к задаче правильного распараллеливания исходной постановки, что является не самым простым делом.

Когерентность

Одной из важнейших проблем параллельных вычислений является обеспечение когерентности памяти и кэша. Несколько вычислителей, работающих над одной и той же ячейкой памяти, могут изменять ее произвольно по своему желанию. Если один вычислитель изменит ячейку, а второй после этого прочитает ее в рамках двух несвязанных друг с другом алгоритмов, то результат, очевидно, будет неправильным. Для решения проблемы применяются такие средства, как атомарные операции, блокировки, семафоры и другие подобные методы.

Более серьезной проблемой является проблема когерентности кэша. Допустим, что изменение ячейки одним вычислителем и ее последующее чтение другим является как раз тем, чего мы хотим достичь. В системах, не имеющих кэширования (а таковых сейчас почти нет), здесь не возникает никакой проблемы. Однако при использовании кэша, если вычислители уже давно работают с ячейкой, она скорее всего хранится в их кэшах и все операции изменяют кэшированное значение, а не реальное. То есть второй вычислитель может простонапросто не знать о том, что кто-то другой сделал какие-то изменения! Эта проблема крайне серьезна, поскольку отключение кэширования может вызвать значительный регресс производительности. Для решения проблемы в данный момент применяются аппаратные решения, уведомляющие каждый вычислительный узел об изменении кэшированного значения.

В заключение этого раздела нужно отметить, что, несмотря на существующие проблемы, параллельные вычисления хоть и являются далеко не идеальными, заслуживают самого пристального рассмотрения и позволяют добиваться удивительных результатов в повышении производительности вычислений.

Блокировки, гонки, критические секции

При вычислениях с разделяемой памятью, как я уже говорил выше, всегда существует возможность того, что несколько вычислителей будут одновременно обращаться к одной и той же ячейке / области памяти, что может привести к непредсказуемым результатам для работающих алгоритмов. Такая проблема называется проблемой “гонки за ресурсами”, а часть программы, в которой может присутствовать потенциальная гонка, называется “критической секцией”.

Для решения проблемы гонок применяются средства синхронизации: семафоры, блокировки, мьютексы, фьютексы. Понятие семафора было введено Эдсгером Дейкстрой и представляет собой объект, ячейку памяти, хранящую целое число, позволяющий только заданному числу вычислителей войти в критическую секцию. Семафор инициализируется этим числом (обозначим его *N*) и привязан к определенной критической секции. Каждый вычислитель, который хочет войти в эту секцию, сравнивает значение семафора с нулем и, в случае утвердительного результата, входит в состояние ожидания до тех пор, пока значение не станет больше нуля. После этого (или сразу же, если ожидание не требовалось) вычислитель входит в критическую секцию и уменьшает значение семафора на единицу. При выходе из критической секции вычислитель должен увеличить значение семафора. Разумеется, ответственность за правильную и “честную” реализацию этих операций лежит на программисте. Кроме того, было бы нелишним отметить, что все изменения семафора должны проходить *атомарно*, то есть таким образом, чтобы во время их самих не возникла ситуация гонки. В самом идеальном случае должна существовать специальная инструкция процессора, позволяющая произвести такое изменение. Семафор также должен поддерживать очередь ожидающих его вычислителей.

Мьютекс является семафором, позволяющим войти в критическую секцию ровно одному вычислителю, и применяется в практических программах наиболее часто, поскольку прост в реализации, понимании и управлении. Фьютекс отличается от мьютекса тем, что последний обычно реализован в ядре ОС, а первый — в виде пользовательского объекта.

Еще одним простым и низкоуровневым средством синхронизации является спинлок, или циклическая блокировка. Она, как и мьютекс, представляет собой переменную, которая может принимать значения, трактуемые исключительно с точки зрения двоичной логики “свободно” / “занято”. Вычислитель либо меняет состояние спинлока с “свободно” на “занято” и входит в критическую секцию, либо в цикле ожидает момента, когда спинлок станет “свободным”. Как видно, в данном варианте блокировки отсутствует очередь ожидающих вычислителей.

Взаимоблокировка

Несмотря на то что средства, описанные выше, позволяют избавиться от проблемы гонок, они сами привносят некоторые сложности. Представим ситуацию, когда вычислитель (A) хочет получить доступ к ресурсу (mB), который в данный момент

занят другим вычислителем (B). Вычислитель же B, в свою очередь, хочет получить доступ к ресурсу mA, который занят вычислителем A. Такая ситуация, когда оба вычислителя не могут двинуться с места, называется взаимоблокировкой и является одной из самых серьезных проблем синхронизации. Для ее решения существуют специальные техники, например, алгоритм банкира, разработанный Дейкстрой.

Методики распараллеливания задач

Чаще всего проблема распараллеливания существующей задачи ложится на плечи программиста. Как к ней подойти? Вот несколько небольших рецептов, которые могут упростить жизнь, с технической стороны:

- Использовать парадигмы программирования, способствующие автоматическому распараллеливанию таких операций, как циклы (например, функциональное программирование);
- Использовать компиляторы, умеющие автоматическое распараллеливание;
- Изучить соответствующие директивы, которые могут помочь компилятору;

С точки зрения дизайна программы:

- Найти места, в которых происходит больше всего вычислений, и понять, зависимы эти вычисления друг от друга или нет;
- Обнаружить узкие места в программе, такие, как ввод/вывод, и выделить их в отдельные блоки, поскольку они могут помешать распараллеливанию;
- Разбить задачу на независимые подзадачи либо находя однородности в данных (и применить параллелизм на данных), либо находя однородности в действиях при разнородных данных (и применить параллелизм кода);
- Определить, насколько сильно ваши подзадачи нуждаются во взаимной коммуникации;
- Выбрать наиболее подходящий тип коммуникации — разделяемую память или же обмен сообщениями, часто в зависимости от объемов данных, которые должны быть общими;
- Выбрать правильную тактику синхронизации;
- Обеспечить такое разбиение на подзадачи, чтобы нагрузка на каждый вычислитель была примерно одинаковой.

В целом вышеприведенные советы являются очень общими и в каждом конкретном случае возникают свои, порой непредвиденные, сложности.

Параллельные вычисления с разделяемой памятью

В этом разделе я рассмотрю самый популярный и широко распространенный вид параллелизма — параллелизм с разделяемыми ресурсами на более практическом уровне. Для начала введем все необходимые термины и классификации.

fork

Программа, которая хочет использовать параллелизм и выполнить несколько операций одновременно (как правило, асинхронно), должна создать некие объекты, поддерживающие такой тип вычислений. В самом общем случае такими объектами являются самые обычные процессы, представляющие другие программы или копии текущей. Самым простым способом создания такой копии (в операционных системах семейства UNIX) является системный вызов fork, имеющий следующий прототип в языке C (описан в файле <unistd.h>):

```
pid_t fork(void);
```

Программа, вызвавшая эту функцию, создает свою полную копию, включая адресное пространство, данные об открытых файлах, текущий рабочий каталог и многие другие сведения. Выполнение как родительского процесса, так и свежесозданного продолжается с точки, расположенной сразу за вызовом fork. Рассмотрим следующую программу на C:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main()
{
  fork();
  printf("Hi there! I'm %i\n", getpid());
  exit(0);
}
```

Первым делом в теле этой программы вызывается fork, что приводит к созданию копии. Затем, как в исходной, так и в новой программе, выполняется строчка:

```
printf("Hi there! I'm %i\n", getpid());
```

— которая выводит на экран фразу “Hi there! I’m ...” с цифровым идентификатором текущего процесса. После этого программа завершается, при этом завершаются и все порожденные ей процессы. В том случае, если порожденный процесс завершится, а родительский нет, родитель должен вызвать функцию wait или waitpid (определены в <sys/wait.h>), в противном случае “ребенок” повиснет в списке процессов в качестве так называемого “процесса-зомби”, а его PID (идентификатор процесса) не будет освобожден. Как правило, вызов fork применяется для запуска какой-либо другой программы из текущей, а не для обеспечения реальной параллельности, однако все адреса в памяти (если не рассматривать сложные манипуляции с виртуальной памятью) и файлы являются общими. В современных системах рекомендуется использовать специальные средства межпроцессного взаимодействия: библиотеки разделяемой памяти, сокеты, неименованные каналы, файлы.

Для запуска другого процесса (из файла) применяется семейство функций exec.

Нити

Пожалуй, самым важным вариантом так называемой *многозадачности* является *нитевая многозадачность*, в которой вводится понятие *нити* (thread[2]). Нить является отдельным объектом исполнения в рамках процесса. Все нити внутри процесса имеют одно и то же адресное пространство памяти и одни и те же разделенные ресурсы, однако есть и локальные данные каждой нити. Как правило, считается, что создание нити требует гораздо меньше времени, чем создание полноценного процесса. Нити могут находиться под управлением как ядра ОС, так и пользовательской программы. В последнем случае, особенно если разговор ведется о каком-нибудь языке программирования с небольшой виртуальной машиной (как Python, допустим), употребляют термин "зеленые нити" (green threads). Соответственно, полной поддержкой многопроцессорности может обладать только первый вариант реализации нитей, управляемый ядром, поскольку обычно из пользовательского кода невозможно напрямую получить доступ к таким низкоуровневым средствам. Для конечного программиста механизм поддержки нитей может выглядеть либо как библиотека с необходимыми функциями, либо как набор директив компилятора. Рассмотрим оба варианта.

Нити обычно используют так называемую *вытесняющую многозадачность*. В этом случае каждому вычислителю, каждой нити предоставляется определенный квант времени (возможно, в зависимости от некоего *приоритета*), в течение которого она находится в рабочем состоянии, после чего планировщик переводит ее в состояние ожидания и отдает физическое устройство другой нити. Проектирование планировщиков, которые обеспечивают вытесняющую многозадачность на системах с большим количеством физических процессоров, — не самая простая задача.

Ярчайшим примером реализации полноценных нитей на уровне ядра ОС является библиотека pthread — POSIX threads, реализованная во многих вариантах UNIX, в том числе GNU/Linux. Рассмотрим следующий код, написанный с ее использованием (для успешной компиляции gcc нужно передать ключ -pthread):

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 10
void *thread_proc(void *threadid)
{
  long tid;
  tid = (long) threadid;
  printf("Hi there! I'm %li\n", tid);
  pthread_exit(NULL);
}
int main()
{
  pthread_t threads[NUM_THREADS];
  long t;
  for (t = 0; t < NUM_THREADS; t++)
  {
    pthread_create(&threads[t],
    NULL, thread_proc, (void *) t);
  }
  pthread_exit(NULL);
}
```

В данной программе создается 10 (NUM_THREADS) нитей, код каждой из которых представляет собой функцию thread_proc. Каждая нить печатает на экран свой идентификатор. Функция создания нити — pthread_create принимает четыре аргумента: указатель на значение типа pthread_t, атрибуты нити (здесь мы не будем их рассматривать), указатель на функцию, которая представляет поток кода нити и аргумент, который будет передан вышеуказанной функции (как правило, этим аргументом будет номер нити или что-то подобное). Каждая нить, в том числе и главная, представленная функцией main, перед завершением должна вызвать функцию pthread_exit.

Допустим, что главной нити необходимо дождаться результата работы какой-то из дочерних. Что делать? Для этого в pthread предусмотрены соответствующий механизм синхронизации, называемой "присоединением к нити", и функция pthread_join. Приведем такой пример:

```
#include <pthread.h>
#include <stdio.h>
void *long_long_loop(void *arg)
{
  long i;
  long a = 1;
  for (i = 0; i < 10000000; i++)
  {
    a++;
  }
  printf("Summing finished!\n");
  pthread_exit(NULL);
}
int main()
{
  pthread_t thread;
  pthread_create(&thread, NULL,
                 long_long_loop, NULL);
  printf("Summing started!\n");
  pthread_join(thread, NULL);
  printf("Really finished!\n");
  pthread_exit(NULL);
}
```

Если бы функция pthread_join не была вызвана, сообщение "Really finished!" было бы выведено на экран раньше, чем нить long_long_loop завершилась бы в действительности. Здесь также необхо-

[2] Перевод этого слова как "поток" может вызвать конфликт терминологии с "потоком" — "stream".

димо отметить, что для присоединения к нити она должна быть создана с атрибутом PTHREAD_CREATE_JOINABLE, что обычно (но не во всех реализациях) происходит по умолчанию. Для явного присваивания нити этого атрибута потребуется код вроде этого:

```
pthread_attr_t attr;
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr,
PTHREAD_CREATE_JOINABLE);
pthread_create(&thread, NULL,
               long_long_loop, NULL);
pthread_attr_destroy(&attr);
```

В pthread также поддерживаются механизмы синхронизации, основанные на мьютексах. Рассмотрим такую выдуманную программу, в которой две нити обращаются к одной и той же глобальной переменной:

```
#include <pthread.h>
#include <stdio.h>
long a = 0;
pthread_mutex_t a_mutex;
void *thread_proc(void *threadid)
{
  long tid;
  long i;
  tid = (long) threadid;
  for (i = 0; i < 10000000; i++)
  {
    printf("I'm %li. Let me read it...\n",
           tid);
    pthread_mutex_lock(&a_mutex);
    if (tid == 1) a++; else a--;
    pthread_mutex_unlock(&a_mutex);
    printf("I'm %li. Unlocked! a=%li\n",
           tid, a);
  }
  pthread_exit(NULL);
}
int main()
{
  pthread_t t1, t2;
  pthread_mutex_init(&a_mutex, NULL);
  pthread_create(&t1, NULL, thread_proc,
                 (void *) 1);
  pthread_create(&t2, NULL, thread_proc,
                (void *) 2);
  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  pthread_mutex_destroy(&a_mutex);
  pthread_exit(NULL);
}
```

В этой программе две нити поочередно работают над глобальной переменной a: первая нить постоянно увеличивает ее значение на единицу, а вторая — уменьшает, при этом для блокировки доступа к a используется мьютекс a_mutex. Мьютекс имеет тип pthread_mutex_t и инициализируется функцией pthread_mutex_init. Попытка захвата мьютекса осуществляется с помощью функции pthread_mutex_lock, которая не возвращает управление до тех пор, пока мьютекс не будет захвачен. Для некоторых случаев, в которых желательно ручное управление синхронизацией, используется функция pthread_mutex_trylock, которая немедленно возвращает ошибку в случае, если мьютекс нельзя захватить. Освобождается мьютекс функцией pthread_mutex_unlock.

Еще одним важным механизмом синхронизации являются *условия*, в pthread представленные в виде сигналов. Допустим, что одной нити нужно дождаться, пока некая переменная примет нужное значение. Нить в таком случае может войти в бесконечный цикл чтения-проверки условия, что, однако, отнимет слишком большое количество процессорного времени и создаст чрезмерную нагрузку на систему. Рассмотрим следующую программу:

```
#include <pthread.h>
#include <stdio.h>
#define NUM_THREADS 10
#define A_LIMIT 31337
long a = 0;
int  c = 0;
pthread_mutex_t c_mutex;
pthread_mutex_t a_mutex;
pthread_cond_t  a_cond;
void *thread_proc(void *threadid)
{
  long tid;
  long i;
  tid = (long) threadid;
  for (i = 0; i < A_LIMIT; i++)
  {
    pthread_mutex_lock(&a_mutex);
    a++;
    pthread_mutex_lock(&c_mutex);
    c = tid;
    pthread_mutex_unlock(&c_mutex);
    if (a == A_LIMIT)
        pthread_cond_signal(&a_cond);
    pthread_mutex_unlock(&a_mutex);
  }
  pthread_exit(NULL);
}
void *watch_a(void *threadid)
{
  pthread_mutex_lock(&a_mutex);
  pthread_cond_wait(&a_cond, &a_mutex);
  pthread_mutex_lock(&c_mutex);
  printf("The winner is %i!\n", c);
  pthread_mutex_unlock(&c_mutex);
  pthread_mutex_unlock(&a_mutex);
  pthread_exit(NULL);
}
int main()
{
  pthread_t threads[NUM_THREADS],
  watcher;
```

```
  long i;
  pthread_mutex_init(&c_mutex, NULL);
  pthread_mutex_init(&a_mutex, NULL);
  pthread_cond_init(&a_cond, NULL);
  for (i = 1; i <= 10; i++)
  {
    pthread_create(&threads[i], NULL,
              thread_proc, (void *) i);
  }
 pthread_create(&watcher, NULL, watch_a,
                (void *) 0);
  for (i = 1; i <= 10; i++)
  {
    pthread_join(threads[i], NULL);
  }
  pthread_join(watcher, NULL);
  pthread_cond_destroy(&a_cond);
  pthread_mutex_destroy(&a_mutex);
  pthread_mutex_destroy(&c_mutex);
  pthread_exit(NULL);
}
```

Здесь десять нитей поочередно увеличивают значение переменной a и проверяют, достигло ли ее значение величины A_LIMIT. В случае, если это произошло, цикл завершается и посылается сигнал с помощью переменной a_cond типа pthread_cond_t. Одна нить является наблюдателем и ждет, с помощью функции pthread_cond_wait, пока не придет сигнал о завершении, после чего "объявляет победителя". Подобное поведение бывает достаточно удобным во многих случаях. На этом рассмотрение библиотеки POSIX Threads можно завершить.

В качестве аналога приведем также программу, работающую с нитями в системе Windows:

```
#include <windows.h>
#include <tchar.h>
#include <strsafe.h>
#define NUM_THREADS 10
#define BUF_SIZE 256
DWORD WINAPI ThreadFunc(LPVOID lpParam);
int b = 0;
HANDLE hBMutex;
int _tmain()
{
  int a = 3;
  int i;
  HANDLE hThreads[NUM_THREADS];
  DWORD  dwThreadId;
  hBMutex = CreateMutex(NULL, FALSE,
                        NULL);
  for (i = 0; i < NUM_THREADS; i++)
  {
     hThreads[i] = CreateThread(NULL, 0,
        ThreadFunc, &a, 0, &dwThreadId);
  }
  WaitForMultipleObjects(NUM_THREADS,
            hThreads, TRUE, INFINITE);
  for (i = 0; i < NUM_THREADS; i++)
  {
    CloseHandle(hThreads[i]);
  }
  CloseHandle(hBMutex);
  return 0;
}
DWORD WINAPI ThreadFunc(LPVOID lpParam)
{
  int *my_a;
  HANDLE hStdout;
  char msgBuf[BUF_SIZE];
  DWORD dwChars;
  DWORD dwWaitResult;
  hStdout =
      GetStdHandle(STD_OUTPUT_HANDLE);
  my_a = (int *) lpParam;
  dwWaitResult =
 WaitForSingleObject(hBMutex, INFINITE);
  if (dwWaitResult == WAIT_OBJECT_0)
  {
    b++;
    ReleaseMutex(hBMutex);
  }
  sprintf(msgBuf, "I'm %i, a = %i,
          b = %i\n",
          GetCurrentThreadId(),
          *my_a, b);
  WriteConsole(hStdout,
  msgBuf, strlen(msgBuf),
  &dwChars, NULL);
  return 0;
}
```

Как можно заметить, основные концепции здесь точно такие же. Нить имеет тип HANDLE и создается вызовом функции CreateThread. Аналогом "присоединения" или ожидания является функция WaitForSingleObject/WaitForMultipleObjects. Мьютексы тоже имеют тип HANDLE, создаются функцией CreateMutex. Ожидание и попытка захвата мьютекса производятся все той же функцией WaitForSingleObject, а освобождение — функцией ReleaseMutex. Однако более предпочтительным средством синхронизации нитей в Windows является использование блокировок критических секций, например, с их использованием вышеприведенная программа перепишется так:

```
#include <windows.h>
#include <tchar.h>
#include <strsafe.h>
#define NUM_THREADS 10
#define BUF_SIZE 256
DWORD WINAPI ThreadFunc(LPVOID lpParam);
int b = 0;
CRITICAL_SECTION bSection;
int _tmain()
{
  int a = 3;
  int i;
```

```
  HANDLE hThreads[NUM_THREADS];
  DWORD  dwThreadId;
  InitializeCriticalSection(&bSection);
  for (i = 0; i < NUM_THREADS; i++)
  {
     hThreads[i] = CreateThread(NULL, 0,
     ThreadFunc, &a, 0, &dwThreadId);
  }
  WaitForMultipleObjects(NUM_THREADS,
  hThreads, TRUE, INFINITE);
  for (i = 0; i < NUM_THREADS; i++)
  {
    CloseHandle(hThreads[i]);
  }
  DeleteCriticalSection(&bSection);
  return 0;
}
DWORD WINAPI ThreadFunc(LPVOID lpParam)
{
  int *my_a;
  HANDLE hStdout;
  char msgBuf[BUF_SIZE];
  DWORD dwChars;
  hStdout = GetStdHandle(STD_OUTPUT_HANDLE);
  my_a = (int *) lpParam;
  EnterCriticalSection(&bSection);
  b++;
  LeaveCriticalSection(&bSection);
  sprintf(msgBuf, "I'm %i, a = %i,
  b = %i\n", GetCurrentThreadId(),
  *my_a, b);
  WriteConsole(hStdout, msgBuf,
  strlen(msgBuf), &dwChars, NULL);
  return 0;
}
```

Как в Windows, так и в UNIX есть поддержка полноценных семафоров, однако здесь я не буду на них останавливаться. Интерфейсы библиотек, предоставляющих "зеленые нити", практически идентичны приведенным выше, а различие существует только во внутренней реализации.

Распараллеливание с помощью директив компилятора

Важным средством полуавтоматического распараллеливания кода является использование директив компилятора. Компилятор gcc поддерживает один из таких стандартов — OpenMP. Сразу рассмотрим небольшой пример (gcc нужно будет передать ключ -fopenmp):

```
include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
  #pragma omp parallel
  printf("Hello!\n");
  return 0;
}
```

Эта программа выведет строку "Hello!" столько раз, сколько в данной системе существует физических вычислителей, например, четыре раза, если у вас четырехядерный процессор. Распараллелить можно, например, цикл:

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define LEN 1000
int main()
{
  int i;
  #pragma omp parallel for
  for (i = 0; i < LEN; i++)
  {
    printf("Hello from %i\n",
           omp_get_thread_num());
  }
  return 0;
}
```

При этом компилятор автоматически создаст нужные процессы или нити и распределит вычисления между ними. Более сложные примеры использования OpenMP приведены в литературе.

Параллельные вычисления с распределенной памятью

Как правило, в реальных системах существуют вычислители, которые обладают некой приватной областью памяти, которая физически недоступна для других. Такие системы требуют специальных средств обмена сообщениями.

MPI

Индустриальным стандартом в пересылке сообщений является механизм MPI (его основные реализации: MPICH, MPICH2, OpenMPI). Рассмотрим такую программу:

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
#define BUF_SIZE 128
int main(int argc, char **argv)
{
  char idstr[32];
  char buf[BUF_SIZE];
  int num_procs;
  int my_id;
  int i;
  MPI_Status stat;
  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD, &num_procs);
  MPI_Comm_rank(MPI_COMM_WORLD, &my_id);
  if (my_id == 0)
  {
    for(i = 1; i < num_procs; i++)
    {
      sprintf(buf, "Hello %d! ", i);
```

```
      MPI_Send(buf, BUF_SIZE, MPI_CHAR,
               i, 0, MPI_COMM_WORLD);
    }
    for(i = 1; i < num_procs; i++)
    {
      MPI_Recv(buf, BUF_SIZE, MPI_CHAR,
               i, 0, MPI_COMM_WORLD,
               &stat);
      printf("%d: %s\n", my_id, buf);
    }
  }
  else
  {
    MPI_Recv(buf, BUF_SIZE, MPI_CHAR, 0,
          0, MPI_COMM_WORLD, &stat);
    sprintf(idstr, "Processor %d ", myid);
    strncat(buf, idstr, BUFSIZE-1);
    strncat(buf, "reporting for duty\n",
          BUFSIZE-1);
    MPI_Send(buf, BUF_SIZE, MPI_CHAR, 0,
           0, MPI_COMM_WORLD);
  }
  MPI_Finalize();
  return 0;
}
```

Эта программа инициализирует MPI, определяет количество доступных вычислительных узлов и от главного узла посылает сообщение всем остальным, принимая также ответы от них. Подробно рассматривать MPI я не буду, поскольку это довольно обширная и сложная тема для обзора. Отмечу также, что чаще всего используются гибридные техники вроде MPI + OpenMP.

Erlang и акторная модель

Еще одним подходом к реализации механизма пересылки сообщений является акторная модель, получившая наибольшее развитие в языке программирования Erlang. Акторы — это легковесные процессы с распределенной памятью, все общение между которыми происходит за счет пересылки сообщений. Не углубляясь в синтаксис и основы языка, взглянем на программу[3]:

```
-module(test).
-export([start/0, thread_func/0]).
start() ->
  OurProcess = spawn(test, thread_func, []),
  io:format("~p:: Hello ~p,
           I want to say I love you!~n",
           [self(), OurProcess]),
  OurProcess ! {hi_there, self(),
                "I love you"},
  receive
    good_answer ->
      io:format("~p:: I'm so glad!~n",
                [self()]),
      init:stop()
  end.
thread_func() ->
  receive
   {hi_there, P, Message} ->
     io:format("~p:: And hello to you,
                ~p! ~s too~n", [self(),
                 P, Message]),
     P ! good_answer,
     thread_func();
   stop -> ok;
   _ ->
     io:format("I don't understand
                you!~n"),
     thread_func()
  end.
```

В этой программе создается и запускается процесс, которому соответствует функция thread_func, после чего ей посылается сообщение с аргументами hi_there (это атом или символ, в терминологии Лиспа) и строкой “I love you”. Тело этой функции состоит из одного предложения receive, которое получает сообщения и сопоставляет их с образцами. При удачном сопоставлении сообщения {hi_there, P, “I love you”} переменная Message принимает значение “I love you”, а переменная P — значение объекта процесса, пославшего сообщение, и выполняется код, соответствующий нужной ветке, в частности, посылается ответ good_answer. В главном процессе тоже есть предложение receive, которое ждет, пока придет сообщение good_answer, на что выводится сообщение в консоль и происходит завершение программы. Вывод этой программы будеть выглядеть примерно так:

```
<0.2.0>:: Hello <0.29.0>,
          I want to say I love you!
<0.29.0>:: And hello to you, <0.2.0>!
           I love you too
<0.2.0>:: I'm so glad!
```

Акторы могут выполняться на разных процессорах, разных машинах, в разных узлах сети — допустима почти любая степень гибкости. Такая модель показала себя весьма успешной в распараллеливании задач телекоммуникаций.

Другие техники параллельного программирования

В этом разделе я кратко опишу менее популярные (но тоже достаточно весомые) приемы обеспечения параллелизма. Одним из таких способов, позволяющим максимально упростить автоматическое распараллеливание, является применение векторных языков программирования, таких, как APL, J, K и Nial. Наибольшую теоретическую ценность в этом списке представляет Nial, реализующий положения теории массивов. Простейшей автоматически распараллеливаемой конструкцией

[3] Для запуска программы необходимо ввести последовательность команд: erlc test.erl; erl -noshell -s test start.

будет мэппинг — применение функции к каждому элементу списка:

```
EACH (5+) [1, 2, 3, 4]
> [6, 7, 8, 9]
```

Более интересным будет использование так называемых “атласов” функций:

```
EACH [(5+), (6*)] [1, 2, 3]
> + ---+----+-----+
> |6 6 | 7 12 | 8 18 |
> +---+----+-----+
```

Здесь каждая функция была применена к каждому элементу списка в отдельности. Более сложные структуры подразумевают применение списков (“галактик”) трансформеров (вроде EACH) к атласам функций и последующее применение результата к другим спискам или массивам более сложной формы. Эта идиоматика хорошо подходит для автоматического распараллеливания математических и математико-физических задач.

Также в настоящее время набирают популярность такие технологии, как вычисление на GPU (графических процессорах), где особое преимущество имеют задачи компьютерной линейной алгебры, облачные вычисления, представляющие собой сильно распаралленные MIMD-системы с высоким уровнем дублирования и резервного копирования данных, генерация “на лету” параллельных процессоров из FPGA-микросхем, распределенные вычисления по модели @Home, MapReduce и т.д. Все эти технологии, вне сомнения, представляют большой интерес, но в основном базируются на принципах, описанных выше, а в технические детали я углубляться не буду.

Еще о приложениях

Кроме уже описанных приложений параллельных и псевдопараллельных вычислений в сфере вычислительной математики, сложной механики, робототехники, биологии — анализа биологических последовательностей и синтеза третичных структур белков, можно отметить еще несколько более “близких к народу” вариантов. Самым весомым примером тут являются получившие большое распространение в последние годы криптовалюты вроде BitCoin, LiteCoin, позволяющие пользователю как бы “сгенерировать” некоторое количество электронных денег и использовать их в транзакциях. Процесс генерации заключается в переборе хэшей и поиске среди них подходящих и очень хорошо распараллеливается. Изначально генерация была делом только центрального процессора, после ее сумели перенести на графические вычислители, а теперь и на специализированные высокопараллельные устройства.

Перспективы

Разумеется, параллельные вычисления представляют собой очень перспективную область научных исследований и технической практики. Сейчас наблюдается большой всплеск интереса к ним и создание все более и более производительных и продуктивных параллельных систем. Резюмируя, можно сказать, что в ближайшем будущем большая часть рынка, в том числе рынка трудоустройства для программистов, будет требовать знаний о параллельных технологиях.

Ссылки

1. Parallel Computing Tutorial, *https://computing.llnl.gov/tutorials/parallel_comp*.

2. *Цилюрик О., Горошко Е.* QNX/UNIX. Анатомия параллелизма. СПб. – М.: Символ, 2006.

3. T.J. Fountain Parallel Computing. Principles and Practice, Cambridge University Press, 1994.

4. *Боресков А.В., Харламов А.А.* Основы работы с технологией CUDA, ДМК Пресс, 2010.

5. Спецификация OpenMP, *http://openmp.org*.

6. Сайт OpenMPI, *http://www.open-mpi.org/*.

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)

Ведется прием заявок на 2013/14 учебный год

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ **108** УЧЕБНЫХ ЧАСОВ

  Стоимость – 2990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ **72** УЧЕБНЫХ ЧАСА

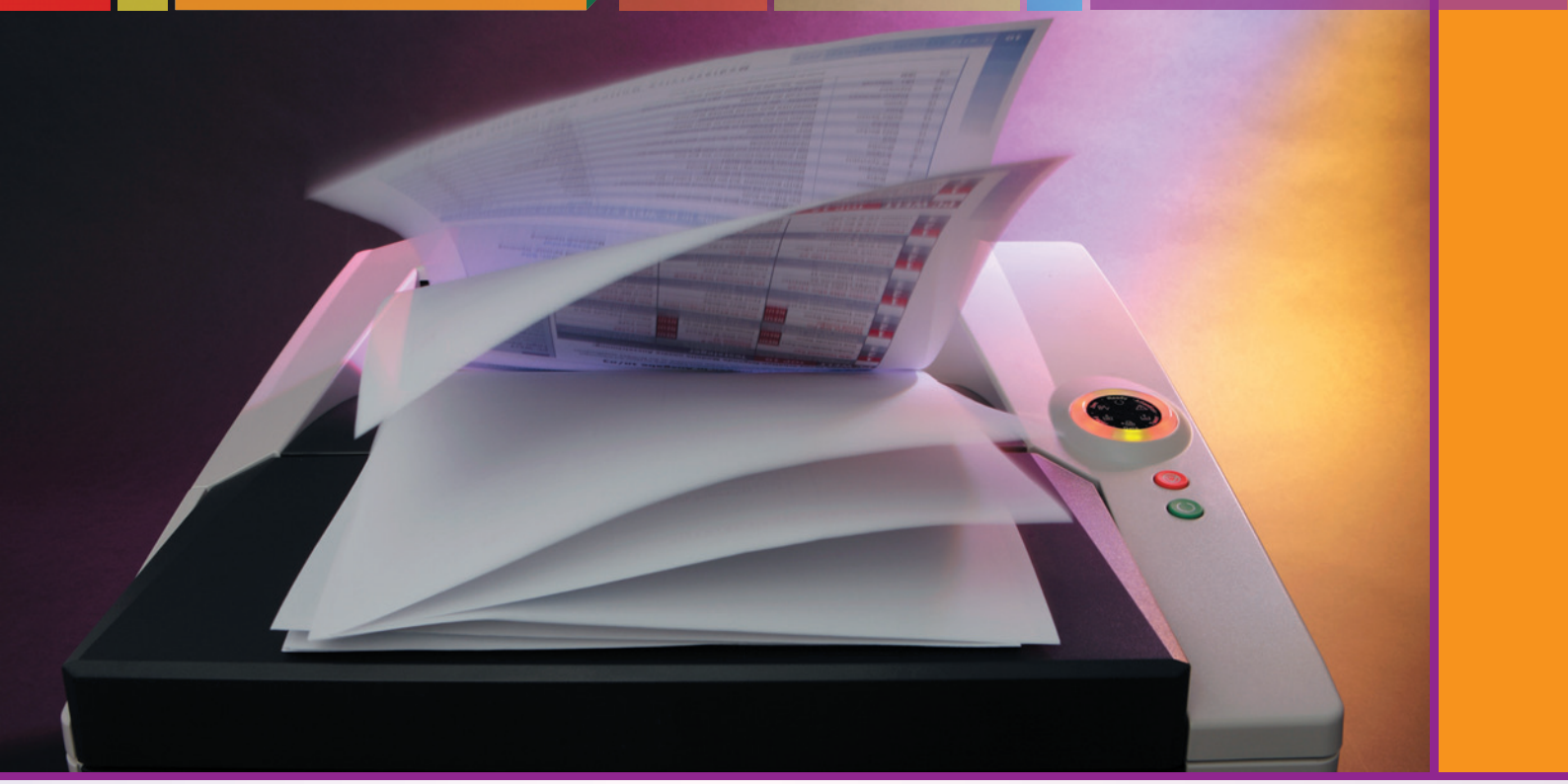
  Стоимость – 2390 руб.

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



Работаем в текстовом редакторе: оформление реферата

О.Б. Богомолова,
д. п. н., преподаватель Московского государственного гуманитарного университета (МГГУ) им. М.А. Шолохова, Москва

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

В настоящее время различные учебные работы — рефераты, тезисы докладов для учебных конференций и пр. — обычно создают и редактируют в текстовых редакторах Microsoft Word или OpenOffice.org Writer. Однако чтобы правильно оформить такую учебную работу, недостаточно только уметь “нажимать правильные кнопки”. Необходимо также знать правила оформления: как сделать для реферата титульный лист, как правильно пронумеровать страницы, сделать ссылки на литературные источники и пр.

Эти вопросы обычно вызывают сложности у учащихся, а в специальной литературе (например, в соответствующих ГОСТах и справочниках) трудно найти описание правил оформления письменных учебных работ понятным для учащихся языком. Ниже представлены рекомендации по оформлению учебного реферата, которые максимально учитывают действующие правила и нормы оформления печатных работ, но упрощены до уровня, достаточного для школьников.

Титульный лист и нумерация страниц

Титульный лист всегда располагается в самом начале документа и является его первой страницей. При распечатке страниц документа на двух сторонах листов бумаги оборотная сторона титульного листа, как правило, оставляется пустой[1]. Для этого можно между титульным листом и первой

[1] В книгах и других официальных печатных изданиях на обороте титула располагаются библиографическая информация, аннотация, указание индексов ББК, УДК и ISBN (по которым данную книгу можно разыскать в библиотечных и других каталогах), а также, в некоторых случаях, так называемые “выходные данные” (сведения о формате книги, ее тираже, дате подписания в печать и пр.). В школьных рефератах и других учебных работах эта информация обычно не приводится.

страницей собственно содержимого реферата добавить пустую страницу либо учесть это при распечатке документа (как — будет показано далее). Соответственно, нумерация страниц документа при его двухсторонней распечатке будет следующей:

• титульный лист — это страница под номером 1 (но сам этот номер страницы на титульном листе не печатается!);

• оборот титульного листа — страница под номером 2 (здесь номер страницы также не печатается);

• первая страница собственно содержимого реферата должна иметь номер 3 (который размещается в колонтитуле), и далее страницы нумеруются по порядку до конца документа. Если в самом конце документа добавлен пустой лист в качестве “обложки”, то на нем номер страницы тоже не печатается.

При распечатке документа только на одной стороне листов бумаги нумерация страниц делается следующим образом:

• титульный лист — это страница под номером 1 (и этот номер страницы на титульном листе не печатается);

• далее следует первая страница собственно содержимого реферата, которая имеет номер 2, и затем страницы нумеруются по порядку до конца документа.

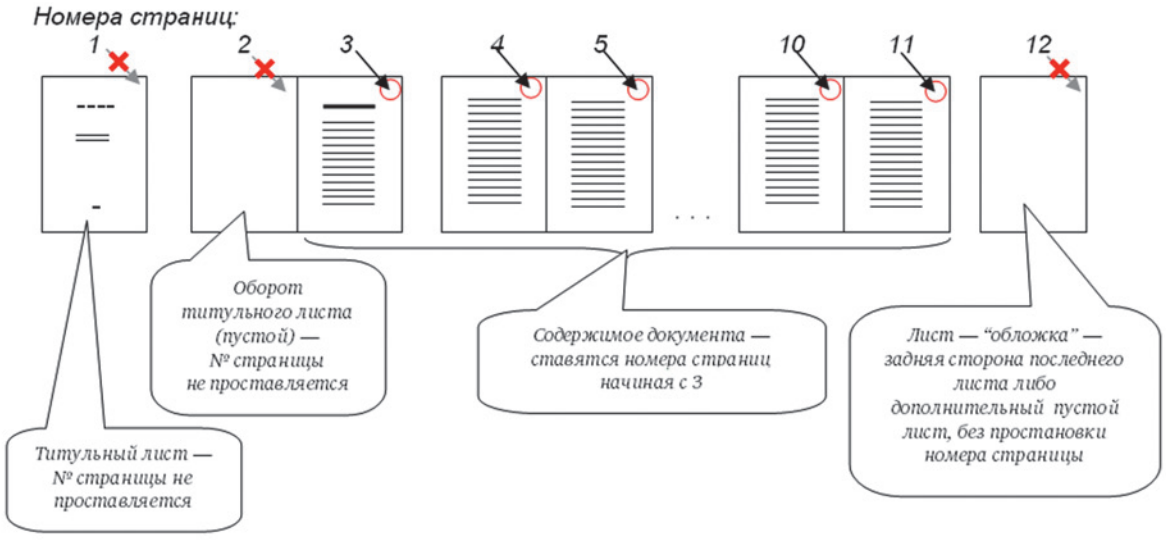
(Очевидно, поскольку в этом случае оборотные стороны листов остаются пустыми, добавлять в конце документа “обложку” из пустого листа уже не нужно.) См. *рис.* 1.

Оформление титульного листа производится по шаблону, представленному на *рис.* 2. Пример см. на *рис.* 3.

Оформление списка использованной литературы

Список использованной литературы оформляется в виде нумерованного (начиная с номера 1) списка *библиографических ссылок* на источники информации, использованные при написании реферата, и располагается на отдельной странице в самом конце документа (но перед содержанием) под заголовком “Литература”. Размер шрифта — такой же, как в основном тексте документа.

Нумерация страниц при двусторонней печати документа:



Нумерация страниц при односторонней печати документа (обратные стороны всех листов — пустые):

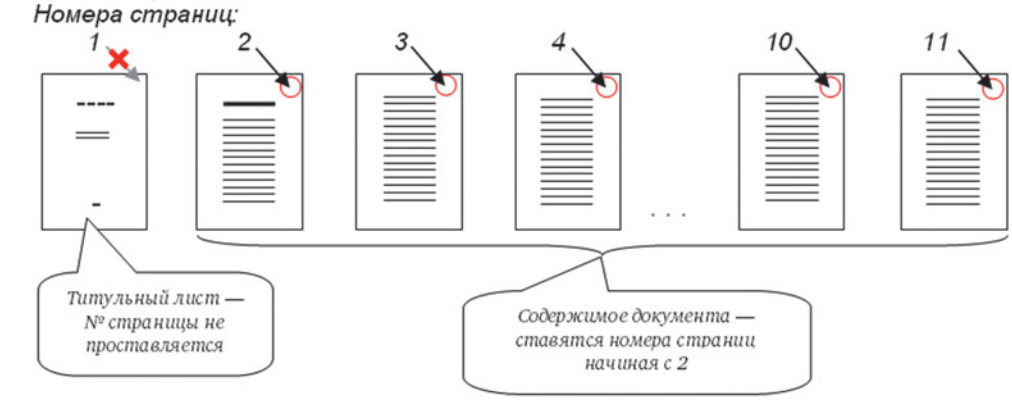


Рис. 1

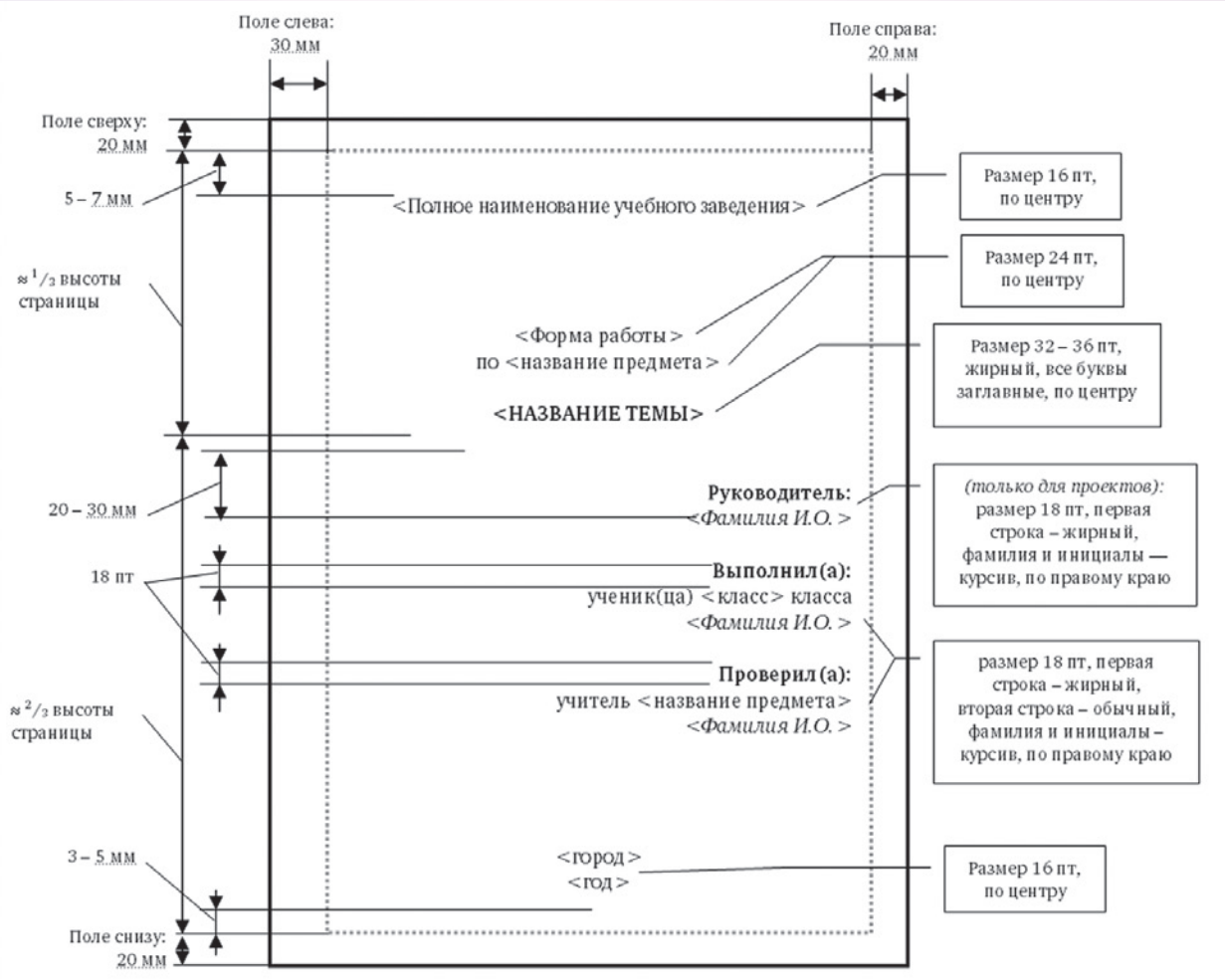
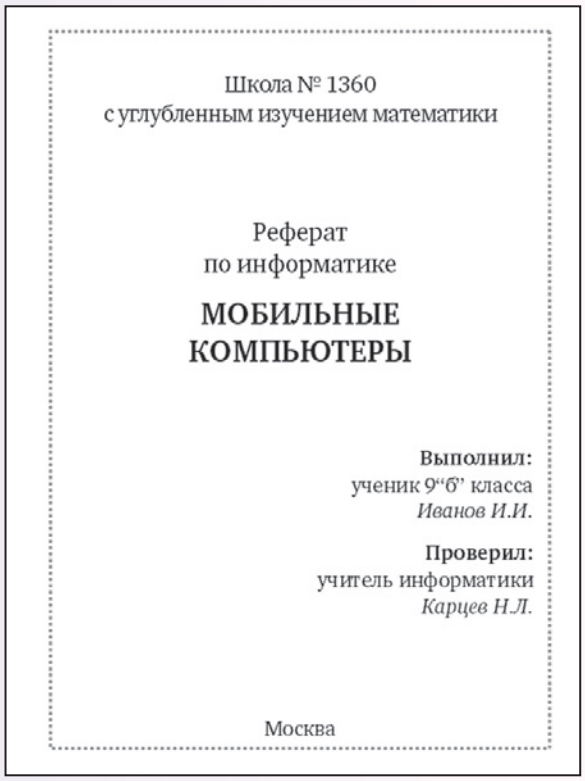


Рис. 2


Школа № 1360
с углубленным изучением математики

Реферат
по информатике

МОБИЛЬНЫЕ КОМПЬЮТЕРЫ

Выполнил:
ученик 9"б" класса
Иванов И.И.

Проверил:
учитель информатики
Карцев Н.Л.

Москва

Рис. 3

В списке литературы обычно сначала приводятся книжные издания (прежде всего энциклопедии, справочники и пр., а затем остальные книги), далее следуют статьи в печатных изданиях (журналах, газетах) и сборниках трудов различных конференций, а последними по списку идут электронные источники (например, web-страницы), и в пределах каждой из этих групп список литературных источников желательно отсортировать по алфавиту. Стандарты оформления библиографических ссылок на источники различного типа установлены соответствующим ГОСТом (ГОСТ Р 7.0.5-2008 "Библиографическая ссылка"). Ниже приведены упрощенные требования к оформлению списка литературы, достаточные для школьных учебных работ.

Стандарт оформления библиографической ссылки на книгу:

<Фамилия И.О.> <Название>:<тип издания>. <Город>: <Название издательства>, <год>.

Исходная информация о книге обычно содержится на обороте ее титульного листа.

В библиографической ссылке на книгу сначала записывается фамилия, а после нее — инициалы автора книги. Если авторов несколько, то они за-

писываются через запятую. Указывается не более трех авторов, если их больше, то после указания трех первых по порядку пишется “и др.”. Фамилии и инициалы авторов принято выделять курсивом.

После авторов записывается точное название книги. Для учебников, учебных или методических пособий после их названия ставится двоеточие, а затем пишется тип издания, например, “учебник для 8-го класса”, “учебное пособие” или “методическое пособие”.

Иногда фамилии авторов не указываются, — например, если при создании книги принимали участие очень много авторов или если это сборник каких-либо нормативных документов. Тогда библиографическая ссылка на такую книгу будет начинаться сразу с ее названия.

После названия книги ставится точка, а затем записывается название города, где была издана книга (для столиц — Москвы и Санкт-Петербурга — при этом используют сокращения: “М.” или “СПб.”; раньше подобное сокращение “Л.” записывали и для Ленинграда). После названия города (полного или сокращенного) ставится двоеточие, затем записывается точное название издательства, а после него, через запятую, — год издания.

Примеры:

1. *Усенков Д.Ю.* Уроки web-мастера. М.: БИНОМ. Лаборатория знаний, 2003.

2. *Иванов И.И., Петров П.П., Сидоров С.С. и др.* Программирование для КПК: учебное пособие. Волгоград: Программист-бук, 2008.

Стандарт оформления библиографической ссылки на статью в журнале (газете):

<*Фамилия И.О.*> <Название статьи> // <Название журнала>. <год>. № <номер>. С. <номера страниц>.

Здесь, так же как и для книг, сначала записывается фамилия, а после нее — инициалы автора книги. Если авторов несколько, то они записываются через запятую. Указывается не более трех авторов, если их больше, то после указания трех первых по порядку пишется “и др.”. Фамилии и инициалы авторов принято выделять курсивом.

После авторов записывается точное название статьи. После него точка не ставится, а через пробел записываются две подряд косые черты. Потом (тоже после пробела) записывается точное название журнала или газеты, после которого ставится точка. Далее записывается год выпуска журнала (газеты), тоже завершаемый точкой. Затем следует знак “№” и номер выпуска; если статья публиковалась с продолжением в нескольких номерах, то знак номера пишется дважды: “№№” и далее через запятую (или через тире, для более чем двух номеров подряд) перечисляются номера выпусков. Иногда журналы (точнее, сборники) не нумеруют, а просто пишут “Выпуск номер такой-то”, тогда вместо “№” надо записать сокращенное “Вып.”, а затем номер выпуска (или несколько номеров — для статьи с продолжением). Наконец, после номера записывается заглавная буква “С” (от слова “страницы”) с точкой после нее, а далее указывается номер страницы в журнале (газете), с которой начинается статья, а еще лучше — диапазон номеров страниц: номер страницы с началом статьи, потом тире, а затем номер страницы, на которой статья заканчивается.

Пример:

1. *Петрова П.П., Иванов И.И.* Нетбук — ноутбук, который всегда с вами // Информатика для всех. 2009, № 3. С. 15–18.

2. *Николайченко Т.В.* Планшетный компьютер — твой помощник // Вестник ПК, 2012. Вып. 3. С. 87–89.

Стандарт оформления библиографической ссылки на тезисы в сборнике конференции:

<*Фамилия И.О.*> <Название тезисов> // <Название конференции>: <Название сборника трудов>. Том <номер тома>. <Город>, <дата проведения>. С. <номера страниц>.

Фамилия автора (или фамилии нескольких авторов) и название тезисов оформляются так же, как для статей в журналах.

Далее после двух подряд символов косой черты (слева и справа от этой пары символов надо ставить пробелы) сначала записывается название конференции (например, “Информационные технологии в образовании”), а затем после двоеточия — название сборника трудов (например, “Сборник трудов XXIII Международного симпозиума”). Далее, если материалы конференции были разделены на несколько томов (или выпусков), то указывается номер тома (выпуска). После этого записывается название города, где проводилось мероприятие (с сокращениями для Москвы и Санкт-Петербурга, так же, как для книг), и через запятую — даты начала и окончания конференции. В конце библиографической ссылки записываются буква “С.” и номер начальной страницы тезисов в соответствующем сборнике (либо диапазон номеров страниц), так же, как это делалось для статей в журнале.

Пример:

1. *Королев А.С.* Опыт применения КПК в системе образования // Информационные технологии в школе: Сборник трудов XXI Всероссийской научно-практической конференции. Том 2. СПб., 28 июля — 3 августа 2010 г. С. 87–91.

Стандарт оформления библиографической ссылки на web-страницу или другой материал, опубликованный в Интернете:

<*Фамилия И.О.*> <Название статьи или иного материала> [электронный ресурс]. URL: <адрес web-страницы> (дата последнего обращения: <день месяц год>).

Для электронных ресурсов, представляющих собой опубликованные на web-сайте статьи или

“электронные книги”, фамилия автора (авторов) и название статьи оформляются так же, как и для статей в журналах. (В некоторых случаях, например для энциклопедических статей в “Википедии”, фамилий авторов может не быть, тогда записывается только название статьи.) Если же речь идет о ссылке просто на материалы, размещенные на какой-либо web-странице (скажем, на описание компакт-диска с учебной программой или на описание технических характеристик компьютеров), то вместо фамилии автора и названия статьи записывается название сайта.

Далее в квадратных скобках пишется указание “электронный ресурс”, ставится точка, записывается сокращение “URL” и после двоеточия — web-адрес данной страницы или сайта. После этого адреса в круглых скобках записывается фраза “дата последнего обращения” и после двоеточия записываются дата (день, месяц и год), когда вы в последний раз проверяли наличие доступа к данному ресурсу.

Примеры:

1. *Солнцедаров А.М.* UMPC идет в народ [электронный ресурс]. URL: *http://www.it-news.su* (дата последнего обращения: 8 августа 2010 г.).

2. “Ладошки к солнцу” (сайт для пользователей КПК) [электронный ресурс]. URL: *http://www.ladoshki.com* (дата последнего обращения: 19 сентября 2010 г.).

Ссылки из текста на список литературы

После составления списка использованной литературы необходимо расставить в основном тексте ссылки на каждый из литературных источников. Обычно такая ссылка представляет собой записанный в квадратных скобках номер литературного источника согласно списку литературы либо несколько таких номеров — при ссылке на несколько источников сразу. Если ссылка производится на весь источник в целом, то достаточно указать только его номер (при ссылке сразу на несколько источников их номера записываются через запятую или в виде диапазона номеров через тире). Если же нужно сослаться на конкретную страницу источника, то внутри квадратных скобок сразу после номера литературного источника через запятую записывается строчная буква “с.” и номер страницы (или диапазон номеров страниц), а записи о разных источниках разделяются уже точками с запятой.

Примеры: [3], [5, 8, 17–19], [6, с. 15], [23, с. 13–18], [1, с. 8; 3, с. 15–16].

Такие ссылки ставятся сразу после упоминания того или иного источника или фамилии его автора в тексте или же в конце цитируемого фрагмента текста из соответствующего источника.

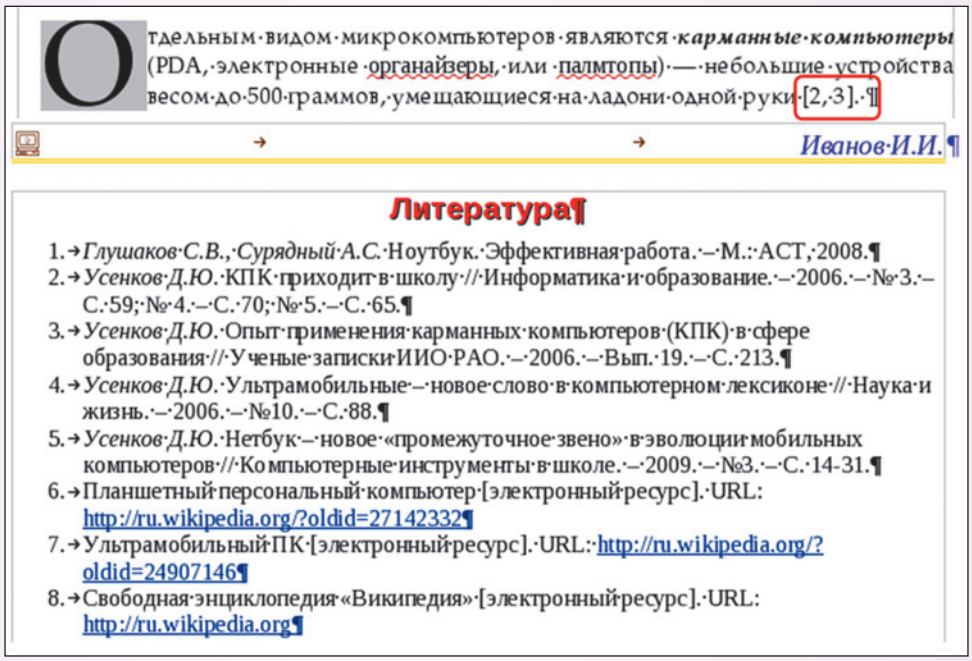
Пример списка литературы и ссылки на литературный источник из основного текста реферата представлены на рис. 4.

Распечатка реферата на двух сторонах листов

Чтобы распечатывать страницы подготовленного текстового документа на обеих сторонах листов бумаги, обычно используют принтеры, оснащенные специальным механизмом автоматического переворота листов для двухсторонней печати. Однако распечатать страницы на двух сторонах листов можно и вручную на любом обычном принтере.

1. Сначала отдельно распечатайте титульный лист (страницу с номером 1). Полученный лист отложите в сторону.

2. Выведите на печать *все нечетные (правые) страницы* документа — они будут напечатаны каж-


Отдельным видом микрокомпьютеров являются ***карманные компьютеры*** (PDA, электронные органайзеры, или палмтопы) — небольшие устройства весом до 500 граммов, умещающиеся на ладони одной руки [2, 3].

Иванов И.И.

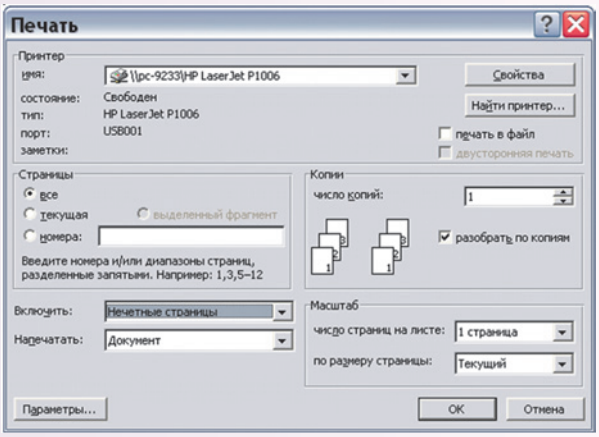
Литература

1. *Глушаков С.В., Сурядный А.С.* Ноутбук. Эффективная работа. – М.: АСТ, 2008.
2. *Усенков Д.Ю.* КПК приходит в школу // Информатика и образование. – 2006. – № 3. – С. 59; № 4. – С. 70; № 5. – С. 65.
3. *Усенков Д.Ю.* Опыт применения карманных компьютеров (КПК) в сфере образования // Ученые записки ИИО РАО. – 2006. – Вып. 19. – С. 213.
4. *Усенков Д.Ю.* Ультрамобильные – новое слово в компьютерном лексиконе // Наука и жизнь. – 2006. – №10. – С. 88.
5. *Усенков Д.Ю.* Нетбук – новое «промежуточное звено» в эволюции мобильных компьютеров // Компьютерные инструменты в школе. – 2009. – №3. – С. 14-31.
6. Планшетный персональный компьютер [электронный ресурс]. URL: http://ru.wikipedia.org/?oldid=27142332
7. Ультрамобильный ПК [электронный ресурс]. URL: http://ru.wikipedia.org/?oldid=24907146
8. Свободная энциклопедия «Википедия» [электронный ресурс]. URL: http://ru.wikipedia.org

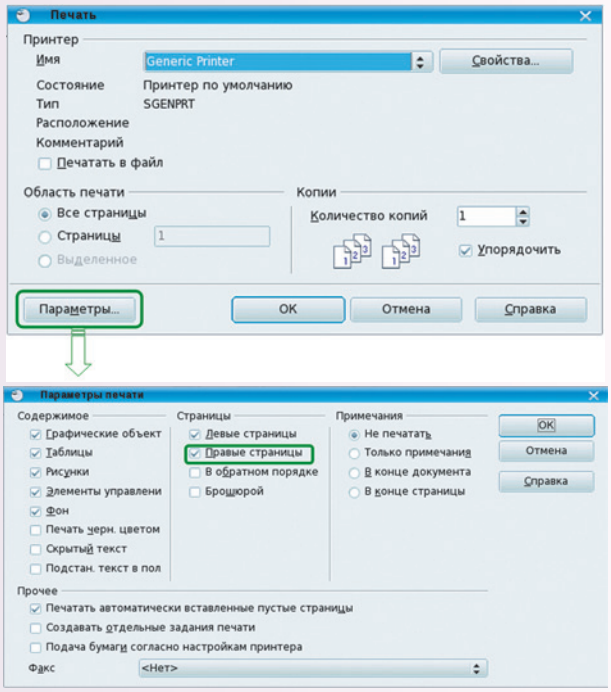
Рис. 4

дая на одной из сторон соответствующего листа бумаги. Чтобы сделать это, нужно в окне настройки параметров печати Microsoft Word 2003 в списке **Включить** выбрать пункт **Нечетные страницы**, а в OpenOffice.org Writer в окне настройки параметров печати нажать кнопку **Параметры**, а затем в отдельном окне **Параметры печати** выбрать переключатель **Правые страницы**:

В Microsoft Word:



В OpenOffice.org Writer:



1. После завершения работы принтера все листы с уже напечатанными правыми страницами нужно вынуть из принтера, перевернуть и заложить снова в его податчик бумаги[2], так чтобы при последующей распечатке вывод изображения производился на оставшуюся чистой обратную сторону этих листов. При этом проследите, чтобы верх обеих печатаемых на листе страниц располагался по одному и тому же краю листа.

2. Выведите на печать *все четные (левые) страницы* документа — они будут напечатаны на оборотных сторонах соответствующих листов бумаги (возможно, кроме самого последнего листа, если в документе было нечетное число страниц, — тогда этот лист надо потом забрать из податчика бумаги).

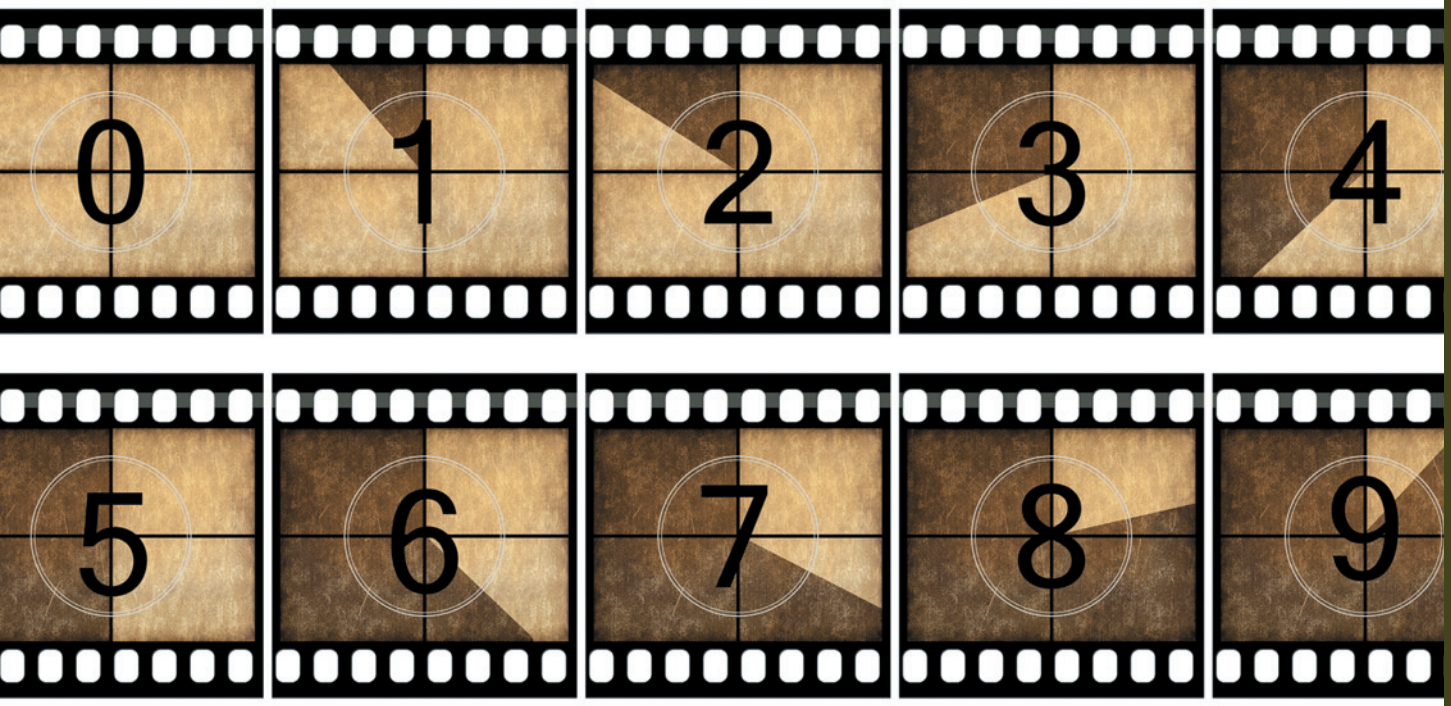
3. Проверьте: страницы документа должны следовать в правильном порядке и с обеих сторон листов не должны быть перевернуты вверх ногами друг относительно друга. После этого добавьте в начало распечатанных листов ранее напечатанный титульный лист. Если нужно, то добавьте в конце распечатки чистый лист — “обложку”.

4. Полученную распечатку документа скрепите (лучше всего степлером, поставив 3–4 скрепки вдоль левого края; можно также проклеить левый край, сделать термоклеевой переплет или переплет пластиковой спиралью). В крайнем случае можно собрать листы в папку типа “Дело…” при помощи дырокола или использовать обычные канцелярские скрепки (вверху и внизу, возле левого края).

Литература

1. *Богомолова О.Б.* Создание документов в OpenOffice.org Writer: практикум. М.: БИНОМ. Лаборатория знаний, 2011.

[2] Предварительно поэкспериментируйте со своим принтером, чтобы определить, как именно нужно закладывать листы для печати на их второй стороне: чистой стороной вниз или вверх, каким краем от себя (или к себе), чтобы верх обеих страниц располагался по одному краю листа, а также можно ли закладывать листы в податчик сразу всей пачкой в том порядке, в каком они перед этим были выданы принтером, или надо сначала переложить листы в обратном порядке. Для разных принтеров все это может быть по-разному. Лучше всего сначала попробовать выборочно напечатать пару страниц на двух сторонах только одного листа, а не пытаться выводить сразу весь документ.



GIF-анимация в OpenOffice.org Impress

О.Б. Богомолова,
д. п. н., преподаватель Московского государственного гуманитарного университета (МГГУ) им. М.А. Шолохова, Москва

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

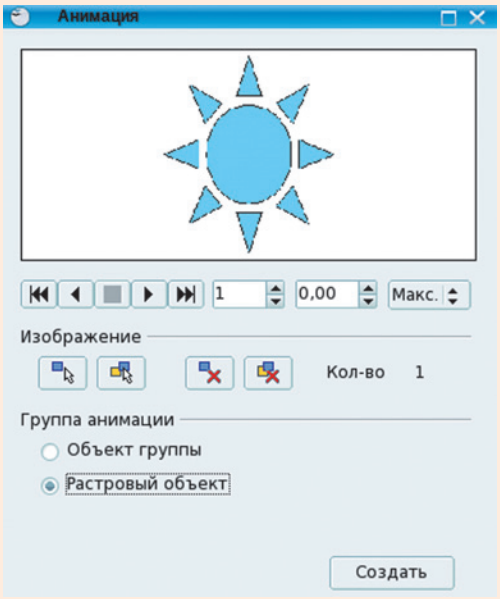
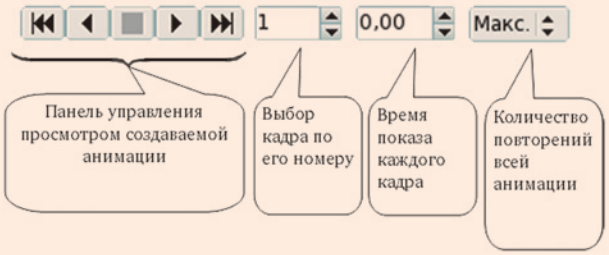
GIF-анимация — это довольно любопытный эффект, реализуемый в одной из разновидностей графического формата GIF: заранее заготовив отдельные рисунки — кадры, изображающие различные положения объекта или фазы движений персонажа и при помощи специальной программы объединив их в один GIF-рисунок, можно получить небольшой мультфильм. Секрет его работы очень прост: в GIF-файле хранится последовательность рисунков-кадров (как кадры на кинопленке), а при просмотре такого GIF-файла компьютер автоматически показывает эти кадры друг за другом с заданным интервалом времени между ними (обычно измеряемым в миллисекундах).

Чтобы создать свою собственную GIF-анимацию, требуется специальная программа (например, GIF Animator). Соответствующие функции также есть в составе пакета Adobe Photoshop. Однако Photoshop — это коммерческий графический редактор, а прочие программы — “GIF-аниматоры” нужно еще сначала где-то найти и скачать (если они бесплатные) или приобрести. Что же делать, если требуется в своей презентации быстро сделать в форме GIF-анимации какую-то схему или несложный рисунок?

Пользователи редактора презентаций Impress из комплекта свободного офисного пакета OpenOffice.org в этом случае оказываются в выигрыше (в отличие от пользователей Microsoft Power Point). Ведь в OpenOffice.org Impress имеется встроенный модуль GIF-анимации!

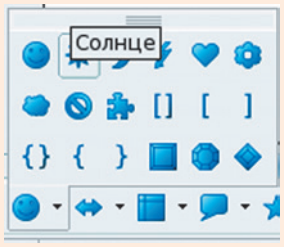
Предлагаемая ниже статья представляет собой материалы для проведения практического занятия у школьников, изучающих OpenOffice.org Impress (одного из цикла практических занятий, предлагаемых в книге [1]).

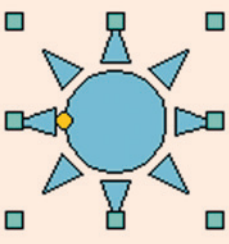
Краткий теоретический минимум

Вставка → Анимированное изображение (можно также вывести на панель инструментов **Рисование** кнопку)	Раскрывает окно **Анимация**, позволяющее конструировать GIF-анимации из заранее нарисованных графических объектов — отдельных кадров анимации (в виде кривых, либо преобразованных в кривые — пункт контекстного меню **Преобразовать → В кривую**)
	Элементы диалогового окна **Анимация**: • переключатель **Объект группы** — создается статичный сгруппированный набор объектов — кадров анимации; • переключатель **Растровый объект** — создается GIF-анимация из объектов — кадров анимации; • панель под полем образца создаваемой анимации:  • кнопка — добавляет один выбранный объект к уже имеющейся в Аниматоре последовательности кадров; • кнопка — добавляет сразу несколько объектов, выделенных одновременно (объекты-кадры должны быть размещены в нужной последовательности в ряд или в “стопку” друг поверх друга); • кнопка — удаляет из создаваемой анимации один текущий кадр; • кнопка — удаляет все ранее добавленные кадры; • текст “Кол-во <*число*>” — указывает текущее общее количество кадров в анимации; • кнопка **Создать** — вставляет на слайд созданную GIF-анимацию (если выбран переключатель **Растровый объект**).

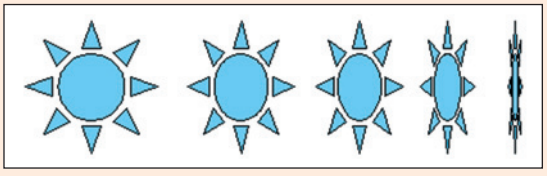
Практические задания

1. Создайте в презентации слайд с разметкой “Только заголовок”. В качестве заголовка введите строку “Создание анимации”. Воспользовавшись кнопкой **Фигуры-символы** в панели инструментов **Рисование**, добавьте на слайд фигуру **Солнце**.

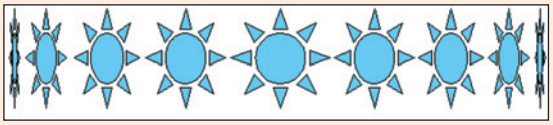




2. Копируя эту фигуру нужное количество раз и масштабируя ее (перетаскивая маркеры на вертикальных сторонах, чтобы менять ширину при неизменной высоте картинки), создайте на слайде следующий ряд фигур — кадров будущей анимации (левая фигура — исходная):

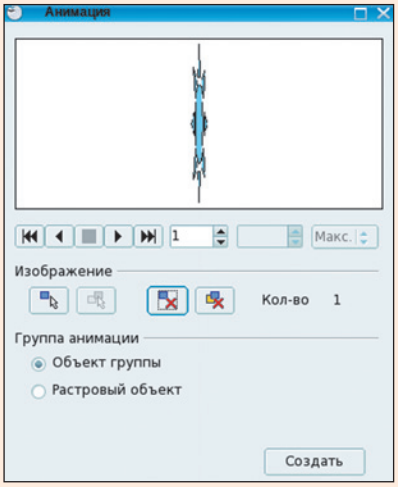


3. Скопируйте четыре правые фигуры и вставьте на слайд их копии. Расположите их в ряд следующим образом:

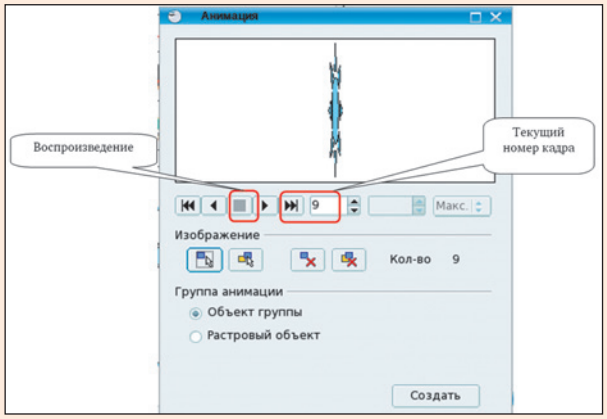


4. Щелкните мышью на кнопке **Анимированное изображение** в панели инструментов **Рисование** (либо выберите в меню пункт **Вставка → Анимированное изображение**). Появится окно **Анимация**.

5. Выделите первую (самую левую) фигуру, щелкните на ней правой кнопкой мыши и выберите в контекстном меню пункт **Преобразовать → В кривую** (голубые квадратики-маркеры сменятся на зеленые). Сразу же (не снимая с фигуры выделение) щелкните мышью на кнопке **Принять объект** в окне **Анимация** — в его верхнем поле появится выбранная фигура:

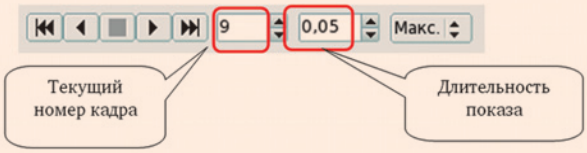


6. Аналогичным способом преобразуйте в кривые и добавьте в окно **Анимация** поочередно остальные заготовленные фигуры (слева направо). В итоге окно **Анимация** примет вид:

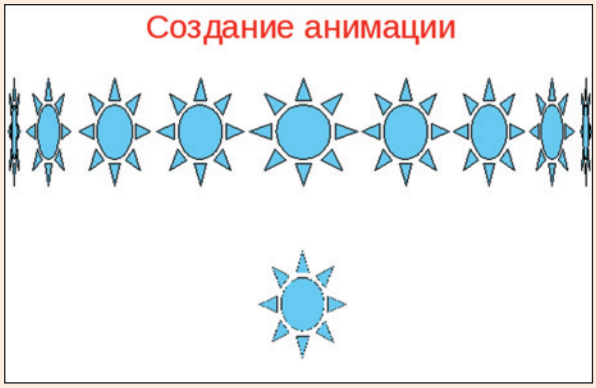


В нем указано, что собранная анимация состоит из 9 кадров. Меняя текущий кадр с 1-го по 9-й и обратно (щелчками мыши на мини-кнопках ▲ и ▼ справа от поля с текущим номером кадра), проследите за изменениями изображения в поле окна **Анимация**. Просмотрите созданную анимацию в режиме автоматической смены кадров, щелкнув мышью на кнопке **Воспроизведение** (▶).

7. Выберите радиокнопку **Растровый объект** в нижней части окна **Анимация**. В ставшем доступным поле **Длительность показа** установите длительность 0,05. Проверьте, чтобы в правом списке (правее поля **Длительность показа**) был выбран пункт **Макс.** — непрерывное воспроизведение анимации.

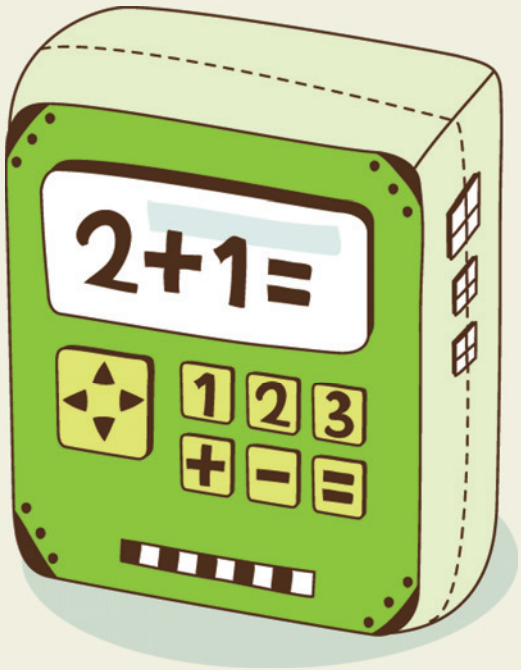


8. Щелкните мышью на кнопке **Создать** в нижнем правом углу окна **Анимация** — на слайде появится анимированная картинка, изображающая вращающееся солнце. Разместите ее на свободном участке слайда под рядом ранее заготовленных фигур — кадров анимации.



Литература

1. *Богомолова О.Б.* Создание презентаций в OpenOffice.org Impress: практикум. М.: БИНОМ. Лаборатория знаний, 2011.



“Каверзная” задача про автомат

О.Б. Богомолова,
д. п. н., преподаватель Московского государственного гуманитарного университета (МГГУ) им. М.А. Шолохова, Москва

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

Очередная тренировочная работа в формате ГИА принесла школьникам новые впечатления ☺. И в их числе — уже вроде бы хорошо знакомая задача № 14 про автомат (исполнитель).

Ее типовое условие:

“У некоторого исполнителя — две команды, обозначенные номерами 1 и 2 и реализующие некоторые математические действия (“прибавить число”, “умножить на число” и т.д.). Требуется составить алгоритм, позволяющий получить из одного числа другое и состоящий не более чем из пяти команд. Ответом является записанная без разделителей (запятых, пробелов и пр.) последовательность номеров команд”.

Например, пусть исполнитель имеет команды:

1) прибавить 1

2) умножить на 2

и требуется получить из числа 3 число 19.

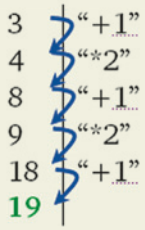
Решение проводится достаточно просто, — только удобнее вести его “от противного”: пытаться найти алгоритм получения из числа 19 (конечного) числа 3 (исходного) за не более чем пять действий, обратных к заданным (т.е. в нашем случае — “вычесть 1” и “разделить на 2”). При этом рассуждения основаны на двух основных идеях: во-первых, надо прийти от одного числа к другому как можно быстрее (ведь количество команд в алгоритме ограничено), а во-вторых, если на каком-то этапе число не делится нацело на 2, то это подсказка: надо применять другую команду (“вычесть 1”).

Такое “обратное” решение записываем “в столбик”, где операции выполняются сверху вниз:

19	“−1”	*Не делится на 2*
18	“/2”	*Делится на 2*
9	“−1”	*Не делится на 2*
8	“/2”	*Делится на 2*
4	“−1”	*Не делится на 2*
3		

А теперь, когда найден “обратный путь”, надо записать тот же алгоритм наоборот — уже от числа 3 к числу 19 и

с использованием исходных команд “прибавить 1” и “умножить на 2”:



Вот и все! Остается только записать последовательность номеров команд (помня, что “+1” имеет номер 1, а “*2” — номер 2): **12121**.

Вроде бы все очень просто. Но в очередной тренировочной работе встретилась задача, в которой такой “механизм решения” дает сбой. (Поскольку копировать задания тренировочных работ их создатели запрещают, даем аналогичную задачу с такой же “подковыркой”.)

Исполнитель имеет две команды:

1) вычесть 2

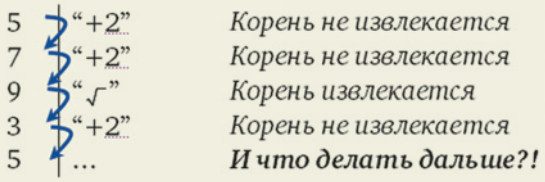
2) возвести в квадрат

Требуется получить из числа 1 число 5 ровно за пять команд.

Попробуем решать эту задачу так же, как и предыдущую.

“Обратные” команды: “прибавить 2” и “извлечь квадратный корень”.

“Обратное” решение:



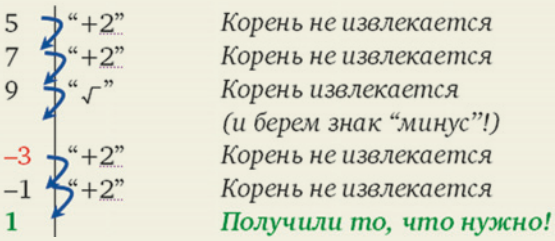
То есть за четыре команды мы не только не приблизились к исходному числу, но и вообще вернулись “на исходные позиции”!

Впрочем, многие ученики (но не все!) все же догадались, в чем тут секрет. Просто здесь в качестве команды исполнителя выбрана “неоднозначная” команда “возвести в квадрат”. Почему “неоднозначная”? Да потому, что она дает один и тот же результат для двух *разных* исходных чисел, *равных по модулю*, но *различающихся знаком*! Поэтому для этой команды надо (в противоречие всем правилам математики) брать “обратную” команду в виде $\pm\sqrt{\ }$. (Впрочем, если вдуматься, то противоречия нет, — ведь мы не просто механически заменяем действие на обратное, а решаем, например, уравнение $x^2 = 9$.)

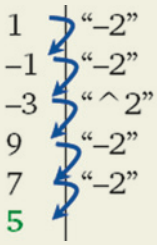
Учитывая это, “общепринятый” способ решения такой задачи с командой “возвести в квадрат” (либо с другой “неоднозначной” командой) нужно модифицировать следующим образом:

- сначала пытаться решать “обратную” задачу как обычно;
- если не получается, то для результата “вычисления квадратного корня” поменять знак результата с плюса на минус;
- если операция “взять квадратный корень” используется несколько раз, то сначала пытаться поменять знак на “минус” у одной такой операции; если не получается, то попробовать у другой, потом — у обеих вместе и так перебрать все возможные варианты.

Вернувшись к нашей задаче, решение “обратной” задачи записываем так:



Остается записать уже решение исходной задачи в исходных командах:



А вот и ответ: 11211.

Задачи С3: “быстрый путь к выигрышу”

О.Б. Богомолова,
д. п. н., преподаватель Московского государственного гуманитарного университета (МГГУ) им. М.А. Шолохова, Москва

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

Задачи С3 на теорию игр, которые пугали школьников в прошлые годы и вроде бы были на какое-то время забыты, с 2012 года вернулись вновь, хотя и несколько в другом облике. Но они по-прежнему вызывают значительные трудности у учащихся.

Напомним, о чем идет речь.

В современной задаче С3 рассматривается игра (обычно — “в камешки”), где в куче изначально содержится некоторое (заранее неизвестное, дан лишь возможный диапазон значений) число камней, а игроки могут поочередными ходами забирать из кучи (либо, наоборот, добавлять в нее) оговоренное число камней. Игра завершается, когда кто-то возьмет последний камень (если камни забираются из кучи) либо когда количество камней в куче превысит указанное значение. Требуется же, проанализировав ход игры, определить, какое изначальное число камней должно быть в куче, чтобы выиграл первый или второй игрок.

Таким образом, эта задача в каком-то смысле обратная по отношению к тем, что предлагались в качестве С3 ранее (там требовалось при известном изначальном числе камней найти, кто из игроков выиграет, и показать его выигрышную стратегию).

Решать такую задачу можно разными способами:

- составляя полную таблицу ходов игроков при каждом возможном исходном количестве камней в куче — вариант “беспроигрышный” в смысле нахождения всех возможных решений, но очень “муторный” и занимающий огромное количество времени;
- путем логических рассуждений — способ достаточно быстрый, но, увы, не всем школьникам по силам, — даже в наглядном варианте с одномерным массивом ячеек, как в статье К.Ю. Полякова “ЕГЭ: новые стратегии (задача С3)” (“Информатика” № 1 за 2013 год) — см. *рис.* 1.

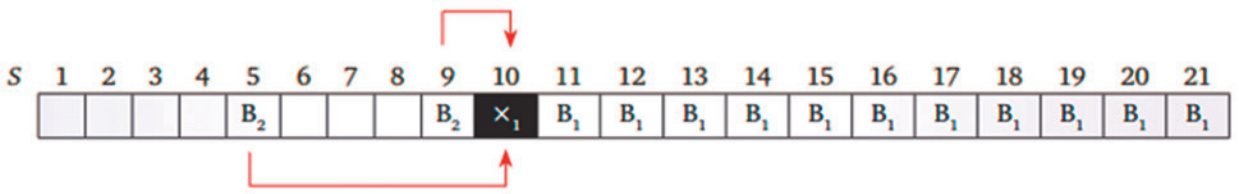


Рис. 1. Наглядно-графический способ логических рассуждений при решении задачи С3, рассмотренный в статье К.Ю. Полякова

Мы же хотим предложить еще один, быстрый алгоритм поиска решения задач С3, достаточно простой, легко усваиваемый учащимися и позволяющий найти решение по крайней мере "типовой" задачи С3 буквально за полминуты!

В качестве примера рассмотрим задачу из демонстрационного варианта ЕГЭ за 2013 год:

Задача 1. Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один** камень или увеличить количество камней в куче в **два** раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 22. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 22 или больше камней.

В начальный момент в куче было S камней, $1 \le S \le 21$.

Будем говорить, что игрок имеет *выигрышную стратегию*, если он может выиграть при любых ходах противника. Описать стратегию игрока – значит описать, какой ход он должен сделать в любой ситуации, которая ему может встретиться при различной игре противника.

Выполните следующие задания. Во всех случаях обосновывайте свой ответ.

1. а) Укажите все такие значения числа S, при которых Петя может выиграть в один ход. Обоснуйте, что найдены все нужные значения S, и укажите выигрывающий ход для каждого указанного значения S.

б) Укажите такое значение S, при котором Петя не может выиграть за один ход, но при любом ходе Пети Ваня может выиграть своим первым ходом. Опишите выигрышную стратегию Вани.

2. Укажите два таких значения S, при которых у Пети есть выигрышная стратегия, причем

– Петя не может выиграть за один ход, и

– Петя может выиграть своим вторым ходом, независимо от того, как будет ходить Ваня.

Для каждого указанного значения S опишите выигрышную стратегию Пети.

3. Укажите значение S, при котором:

– у Вани есть выигрышная стратегия, позволяющая ему выиграть первым или вторым ходом при любой игре Пети, и

– у Вани нет стратегии, которая позволит ему гарантированно выиграть первым ходом.

Для указанного значения S опишите выигрышную стратегию Вани.

Постройте дерево всех партий, возможных при этой выигрышной стратегии Вани (в виде рисунка или таблицы). На ребрах дерева указывайте, кто делает ход, в узлах – количество камней в куче.

Решение

Шаг 1. Анализируем исходные данные в условии:

Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может добавить в кучу **один** камень или увеличить количество камней в куче в **два** раза. Например, имея кучу из 15 камней, за один ход можно получить кучу из 16 или 30 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней. Игра завершается в тот момент, когда количество камней в куче становится не менее 22. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 22 или больше камней.

В начальный момент в куче было S камней, $1 \le S \le 21$.

Требуемые "ключевые" данные выделены выше зеленым цветом. Это описание возможных ходов: "плюс 1" и "умножить на 2", а также граничное количество камней, которое считается выигрышным (22).

Шаг 2. Отвечаем на вопрос 1*а*.

Берем граничное количество камней (22) и применяем к нему "обратные ходы" по отношению к возможным ходам игрока, т.е. "минус 1" и "делить на 2". Получаем: $22 - 1 = 21$, $22 / 2 = 11$. *Вывод:* для выигрыша первого игрока (Пети) первым ходом надо, чтобы в куче изначально было от 11 до 21 камня.

Шаг 3. Отвечаем на вопрос 1*б*.

Из двух ранее полученных значений (выигрышных для первого игрока) берем значение, *самое далекое от выигрыша* (в данном случае — 11) и к нему применяем ход, "обратный" такому ходу игрока, который приводит к выигрышу медленнее всего, т.е. в данном случае — "минус 1". Получаем: $11 - 1 = 10$. *Вывод:* для выигрыша второго игрока (Вани) первым ходом надо, чтобы в куче изначально было 10 камней.

Шаг 4. Отвечаем на вопрос 2.

Берем полученное на предыдущем шаге значение (10) и к нему снова применяем "обратные"

ходы “минус 1” и “делить на 2”. Получаем: 10 – 1 = = 9, 10 / 2 = 5. *Вывод:* для выигрыша первого игрока (Пети) только вторым ходом надо, чтобы в куче изначально было 5 или 9 камней.

Шаг 5. Отвечаем на вопрос 3.

Берем полученные на предыдущем шаге значения (5 и 9) и к ним применяем ход, “обратный” такому ходу игрока, который приводит к выигрышу медленнее всего, т.е. “минус 1”. Получаем: 9 – 1 = = 8, 5 – 1 = 4. *Вывод:* для выигрыша второго игрока (Вани) первым (не гарантированно) или вторым (гарантированно) ходом надо, чтобы в куче изначально было 8 или 4 камня.

Шаг 6. Нужно не забывать, что данный метод — “прикидочный”. Поэтому для ответов на вопросы 2 и 3 нужно все-таки выполнить построение таблицы ходов, подобной следующей (приведена таблица для ответа на вопрос 3), тем более что в решении задачи С3 на ЕГЭ все равно надо представить развернутый ответ.

По таблице внизу страницы мы видим, например, что если в куче изначально было 8 камней, то при первом ходе первого игрока “плюс 1” второй игрок может выбрать свой первый ход “плюс 1” и тем самым при любом втором ходе первого игрока обеспечить себе выигрышный ход, а если первым ходом первого игрока было “умножить на 2”, то второй игрок выигрывает уже своим первым ходом.

То же можно отобразить диаграммой (см. *рис.* 2).

При этом, если для какого-то из найденных “быстрым способом” количеств камней решение не достигнуто, то надо “пробить” в этой таблице соседние значения. Но в любом случае такое построение таблицы и диаграммы можно выполнить гораздо быстрее, чем построение полной таблицы для всех возможных значений (даже с учетом отбрасывания значений исходного количества камней, найденных при ответе на вопрос 1*а*, которые сравнительно нетрудно определить).

Для закрепления рассмотрим решение еще пары задач С3 (для краткости будем приводить только

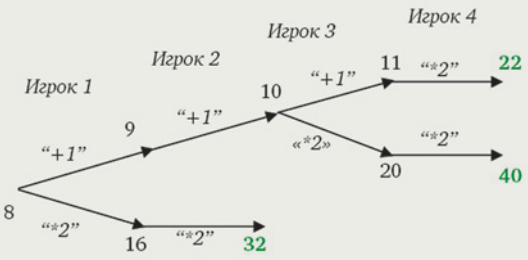


Рис. 2

“смысловую” часть условия, без “типовых” вопросов и комментариев, и сразу выполнять по отношению к нему шаг 1 нашего алгоритма — выделять цветом “ключевые” данные).

Задача 2[1]**.** Два игрока, Петя и Ваня, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает Петя. За один ход игрок может **добавить в кучу один камень** или **увеличить количество камней в куче в пять раз**. Например, имея кучу из 10 камней, за один ход можно получить кучу из 11 или 50 камней. У каждого игрока, чтобы делать ходы, есть неограниченное количество камней.

Игра завершается в тот момент, когда количество камней в куче становится более 100. Победителем считается игрок, сделавший последний ход, то есть первым получивший кучу, в которой будет 101 или больше камней. В начальный момент в куче было S камней, $1 \le S \le 100$.

Решение

Шаг 1. Нужно внимательно проследить, как выполняется вычисление, если деление количества камней невозможно выполнить нацело.

Шаг 2. Отвечаем на вопрос 1*а*.

Берем граничное количество камней (101) и применяем к нему “обратные ходы” — “минус 1” и “делить на 5”. В первом случае получаем: 101 – 1 = 100. А вот во втором случае разделить число 101 нацело на 5 не удается. Поэтому нужно

[1] Задача из тренировочной работы по информатике (февраль 2013 г.).

Изначальное число камней в куче	Первый ход первого игрока	Первый ход второго игрока	Второй ход первого игрока	Второй ход второго игрока
8	9 (“+ 1”)	10 (“+ 1”)	11 (“+ 1”)	12 (“+ 1”)
				22 (“* 2”) — **Выигрыш**
			20 (“* 2”)	21 (“+ 1”)
				40 (“* 2”) — **Выигрыш**
		18 (“* 2”)	19 (“+ 1”)	20 (“+ 1”)
				38 (“* 2”) — **Выигрыш**
			36 (“* 2”) — **Выигрыш**	–
	16 (“* 2”)	17 (“+ 1”)	18 (“+ 1”)	19 (“+ 1”)
				36 (“* 2”) — **Выигрыш**
			34 (“* 2”) — **Выигрыш**	–
		32 (“* 2”) — **Выигрыш**	–	–

делить нацело с округлением вверх (20 + 1 = 21). *Вывод:* для выигрыша первого игрока (Пети) первым ходом надо, чтобы в куче изначально было от 21 до 100 камней.

Шаг 3. Отвечаем на вопрос 1*б*.

Из двух ранее полученных значений (выигрышных для первого игрока) берем значение, *самое далекое от выигрыша* (в данном случае — 21) и к нему применяем ход, "обратный" такому ходу игрока, который приводит к выигрышу медленнее всего, т.е. "минус 1". Получаем: 21 – 1 = 20. *Вывод:* для выигрыша второго игрока (Вани) первым ходом надо, чтобы в куче изначально было 20 камней.

Шаг 4. Отвечаем на вопрос 2.

Берем полученное на предыдущем шаге значение (20) и к нему снова применяем "обратные" ходы "минус 1" и "делить на 5". Получаем: 20 – 1 = 19, 20 / 5 = 4 (в данном случае деление выполняется нацело, поэтому никаких операций с остатком производить не нужно). *Вывод:* для выигрыша первого игрока (Пети) только вторым ходом надо, чтобы в куче изначально было 4 или 19 камней.

Шаг 5. Отвечаем на вопрос 3.

Берем полученные на предыдущем шаге значения (4 и 19) и к ним применяем ход, "обратный" такому ходу игрока, который приводит к выигрышу медленнее всего, т.е. "минус 1". Получаем: 4 – 1 = 3, 19 – 1 = 18. Однако, проверяя эти значения на шаге 6 (построением таблицы), получаем, что значение 3 является ложным ответом. *Вывод:* для выигрыша второго игрока (Вани) первым (не гарантированно) или вторым (гарантированно) ходом надо, чтобы в куче изначально было 18 камней.

Из таблицы внизу страницы видно, что если исходное число камней в куче равно 3, то на первом ходе выигрыш второго игрока невозможен ни в какой ситуации, а выигрыш второго игрока вторым ходом не гарантирован (возможен только при первом ходе первого игрока "+1", однако первый игрок, заведомо зная стратегию игры, выберет первый ход "* 5").

Задача 3[2]**.** Два игрока, обозначенных буквами А и В, играют в следующую игру. Перед игроками лежит куча камней. Игроки ходят по очереди, первый ход делает игрок А. За один ход игрок может взять из кучи **два** или **пять** камней. При последнем ходе игрок может взять и меньшее количество камней (1, 3 или 4), если в куче больше камней не осталось. Например, имея кучу из 10 камней, за один ход можно получить кучу из 7 или 5 камней. Победителем считается игрок, забравший последний камень.

В начальный момент в куче имеется N камней, $1 \le N \le 20$.

Решение

Шаг 1. Сразу заметим: здесь ход решения чуть-чуть отличается от предыдущего на шагах 3 и 4; это отличие касается модификации задачи С3, где выигрывает забирающий последний камень.

Шаг 2. Отвечаем на вопрос 1*а*.

Берем граничное количество камней (0, поскольку в задаче для выигрыша достаточно забрать последний камень) и применяем к нему "обратные ходы" — "плюс 2" и "плюс 5". Получаем: 0 + 2 = 2, 0 + 5 = 5. *Вывод:* для выигрыша первого игрока (А) первым ходом надо, чтобы в куче изначально было от 1 до 5 камней (поскольку он может забирать и менее 2 или 5 камней, если они — последние в куче).

Шаг 3. Отвечаем на вопрос 1*б*.

Берем *оба* ранее полученных значения (выигрышных для первого игрока) — в данном случае 1 и 5. К значению, наиболее *далекому* от выигрыша (5), применяем ход, "обратный" такому ходу игрока, который приводит к выигрышу *медленнее* всего, т.е. "плюс 2". Получаем: 5 + 2 = 7. К значению же, более *близкому* к выигрышу, наоборот, применяем ход, "обратный" ходу, приводящему к выигрышу *быстрее* всего, т.е. "плюс 5". Получаем: 1 + 5 = 6. *Вывод:* для выигрыша второго игрока (В) первым ходом надо, чтобы в куче изначально было 6 или 7 камней.

[2] Авторская задача. Опубликована в составе подборки из 10 задач на CD "ЕГЭ-2013. Информатика. Подготовка к экзамену". М.: ООО "Физикон", 2013.

Изначальное число камней в куче	Первый ход первого игрока	Первый ход второго игрока	Второй ход первого игрока	Второй ход второго игрока
3	4 ("+ 1")	5 ("+ 1")	6 ("+ 1")	7 ("+ 1")
				30 ("* 5")
			25 ("* 5")	26 ("+ 1")
				125 ("* 5") — **Выигрыш**
		20 ("* 5")	21 ("+ 1")	22 ("+ 1")
				105 ("* 5") — **Выигрыш**
			100 ("* 5")	101 ("+ 1") **Выигрыш**
				500 ("* 5") — **Выигрыш**
	15 ("* 5")	16 ("+ 1")	17 ("+ 1")	18 ("+ 1")
				85 ("* 5")
			80 ("* 5")	81 ("+ 1")
				400 ("* 5") — **Выигрыш**
		75 ("* 5")	76 ("+ 1")	77 ("+ 1")
			375 ("* 5") — **Выигрыш**	–

Шаг 4. Отвечаем на вопрос 2.

Берем оба полученных ранее значения (6 и 7) и к ним снова применяем “обратные” ходы “плюс 2” и “плюс 5”. Получаем: 6 + 2 = 8, 6 + 5 = 11, 7 + 2 = = 9, 7 + 5 = 12. *Вывод:* для выигрыша первого игрока (А) только вторым ходом надо, чтобы в куче изначально было 8, 9, 11 или 12 камней.

Шаг 5. Отвечаем на вопрос 3.

Берем полученные на предыдущем шаге значения (8, 9, 11 и 12) и к ним применяем ход, “обратный” такому ходу игрока, который приводит к выигрышу медленнее всего, т.е. “плюс 2”. Получаем: 8 + 2 = 10, 9 + 2 = 11, 11 + 2 = 13, 12 + 2 = 14. Но значение 11, полученное таким “механическим” способом, придется исключить, так как ранее мы уже обнаружили, что это — выигрышное значение для игрока А на втором ходе. *Вывод:* для выигрыша второго игрока (В) первым (не гарантированно) или вторым (гарантированно) ходом надо, чтобы в куче изначально было 10, 13 или 14 камней.

РОБОТЛАНДСКИЙ УНИВЕРСИТЕТ

Роботландский сетевой университет (руководитель — Дуванов Александр Александрович) — работает с октября по май каждого учебного года.

В университет принимаются коллективные, а на курс “42. Web-конструирование” — коллективные и индивидуальные ученики.

Коллективный студент — это группа детей, работающая под руководством одного или нескольких наставников (наставник, как правило, школьный учитель).

Индивидуальный студент — это учитель, желающий пройти обучение индивидуально, без группы детей.

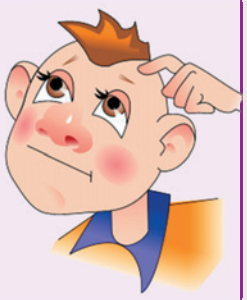
В конце года обучаемые получают удостоверения от негосударственного образовательного учреждения “Роботландия” и грамоты за победы в курсовых конкурсах.

Занятия в университете платные (цены — на странице *www.botik.ru/~robot/ru/price.htm*). Они начинаются 10 октября текущего года и продолжаются в течение двух семестров до мая следующего года.

Для подписчиков “Информатики” предусмотрена 10%-ная скидка за обучение.

Характерные черты Роботландской школы:

1. Совместное обучение учителя и школьников в рамках одной команды.
2. Турнирный цикл обучения.
3. Моделирование коллективной деятельности.
4. Реальная практическая польза детских проектов.
5. Перекрестные проверки работ.
6. Развитые горизонтальные связи.



Заявки принимаются по адресу: *kurs@robotland.pereslavl.ru* **в сентябре.**

Краткое описание курсов

Номер и название курса	Возраст детей	Куратор курса	Описание курса
10. Азбука Роботландии. Компьютер	2–3-й классы	Первин Юрий Абрамович	Введение в информатику на базе курса “Азбука Роботландии”, часть I “Компьютер”
11. Азбука Роботландии. Информация	3–4-й классы	Первин Юрий Абрамович	Продолжение курса “Азбука Роботландии”, часть II “Компьютер”
12. Азы информатики-I. Знакомимся с компьютером. Работаем с информацией	3–5-й классы	Кацай Ирина Ивановна	Начала информатики для малышей. Коллективная работа в социальных сетях, ориентированных на обучение детей
14. Азы информатики-II. Пишем на компьютере	4–6-й классы	Кацай Ирина Ивановна	В рамках обозначенной темы курс связывает пять контентов: познавательный, инструментальный, концептуальный, дизайнерский и творческий. Коллективная работа в социальных сетях, ориентированных на обучение детей
31. Азы программирования-I. Плюсик и Кукарача	6–8-й классы	Садовая Ирина Владимировна	Для детей, знакомых с понятиями “исполнитель”, “алгоритм”, “программа” и желающих поближе познакомиться с программированием
32. Азы программирования-II. Корректор	7–9-й классы	Садовая Ирина Владимировна	Продолжение курса 31 на базе исполнителя Корректор
42. Web-конструирование	старшие классы	Дуванов Александр Александрович	Создание сайтов на базе HTML+CSS+JavaScript (последнее факультативно). Основы проектирования, веб-дизайна и юзабилити. Планируется кружок по теме, связанной с JavaScript+jQuery

ежемесячный журнал
ШКОЛА
ДЛЯ РОДИТЕЛЕЙ

ШКОЛА
ДЛЯ РОДИТЕЛЕЙ
объявляет набор

ДОРОГИЕ УЧИТЕЛЯ,

приглашайте родителей в новый журнал!

- На страницах журнала родители встретятся с педагогами, детскими психологами, педиатрами, семейными консультантами
- Первый номер журнала появится в вашем Личном кабинете 1 сентября вместе с кодами доступа для родителей
- Если журнал вам понравится, коды доступа вы сможете раздать уже на первом родительском собрании

РАБОТА С РОДИТЕЛЯМИ — НАШЕ ОБЩЕЕ ДЕЛО

Объединим усилия!

ГРАФИКА

Постижение гириха

А.И. Азевич,
Москва

В Средние века на мусульманском Востоке было распространено мнение, что геометрия очищает и совершенствует ум. Считалось, что не может совершить ошибку человек, постоянно занимающийся геометрией. Эта мысль прививалась с раннего детства.

В исламской традиции Кораном запрещено изображать людей и животных. В связи с этим в мусульманских странах при оформлении культовых сооружений особую популярность приобрел геометрический орнамент. В его построении средневековые мастера достигли невероятного совершенства. Они передавали свои знания из поколения в поколение.

Орнаментальные рисунки называли “музыкой для глаз”. Для древних художников геометрические композиции были основополагающим мотивом в прикладном творчестве. Восторгаясь красотой восточных мечетей и минаретов, удивляешься математической точности построений, скрытой от непосвященного наблюдателя. Разгадка технических приемов — дело непростое! Попытаемся узнать хотя бы некоторые из них.

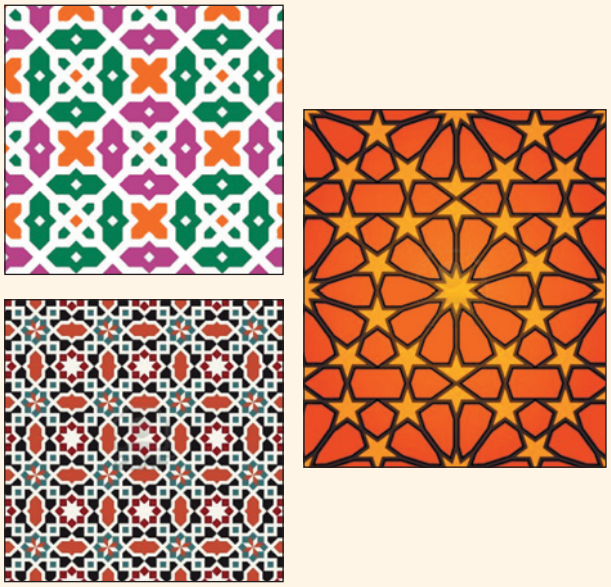


Рис. 1

В орнаментальном искусстве Востока часто встречается необычный геометрический орнамент — так называемый “гирих”. Он представляет собой причудливое переплетение многоугольников и звезд за счет вариации нескольких повторяющихся элементов (слово *гирих* в переводе с персидского языка означает *узел* или *пучок*). Вот несколько примеров[1] (*рис.* 1).

На компьютере удобнее всего строить гирихи в векторном графическом редакторе.

Опишем порядок построения гириха, изображенного на *рис.* 2, в самой известной программе указанного типа — редакторе CorelDraw. Следуя намеченному плану, попытаемся ощутить ритм, симметрию и композицию гириха.

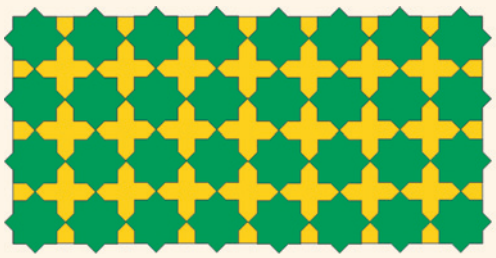


Рис. 2

План построения следующий.

1. Открываем программу CorelDraw и создаем новый документ.

2. Выбираем инструмент прямоугольник и, удерживая нажатой клавишу Ctrl, строим квадрат.

3. Создаем его копию. Для этого выделяем исходную фигуру указателем. Затем нажимаем по-

[1] Примеры гирихов можно также найти на странице *http://ru.123rf.com/search.php?word=islamic&start=840&imgtype=2&searchopts=&itemsperpage=60*.

следовательно на кнопки **Копировать** и **Вставить** ( ).

4. Выделяем копию квадрата и поворачиваем ее на 135°. Для этого внутри квадрата дважды щелкаем мышкой. Появляются стрелочки (*рис.* 3). Тянем за одну из них, например правую верхнюю, до тех пор, пока в окошке “Угол поворота” (135,0 °) не появится число 135. Это значение можно ввести и вручную.

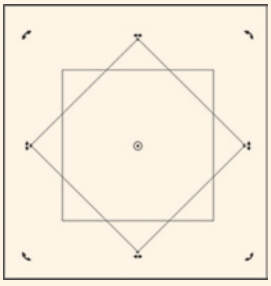


Рис. 3

5. Следующий шаг — объединение квадратов. Выделяем оба квадрата. В строке меню выбираем пункт **Упорядочить** (*рис.* 4), а в нем — команды **Формирование | Объединить**.

В результате получим новый рисунок:

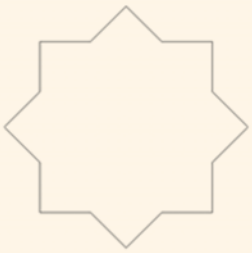


Рис. 5

6. Выделим полученную восьмиконечную звезду и зададим толщину абриса (границы) 1 мм

.

7. Проведем заливку звезды. Для этого сначала ее выделим, а потом нажмем на зеленый квадратик палитры цветов (*рис.* 6). Результат показан на *рис.* 7.

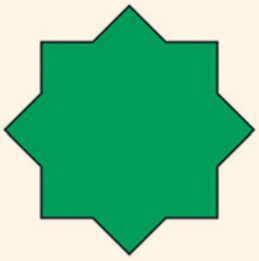


Рис. 7

8. Скопируем последний рисунок (**Копировать** и **Вставить**). Переместим копию так, чтобы звезды имели общую вершину (*рис.* 8).

Рис. 8

9. Выделим звезды. Нажмем на кнопку **Упорядочить**, а затем **Выровнять и распределить** и **Выровнять центры по горизонтали**.

10. Сгруппируем объекты — нажимаем последовательно **Упорядочить | Сгруппировать**.

11. Вновь создадим копию теперь уже сгруппированных объектов и расположим их так, как показано на *рис.* 9.

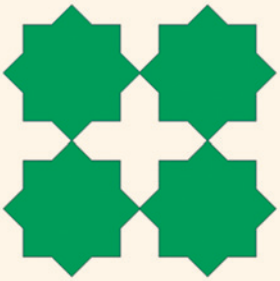


Рис. 9

В результате всех построений получилась композиция, состоящая из четырех восьмиконечных звезд.

12. Зальем желтым цветом центральную часть гириха — своеобразный крест. Для этого начертим поверх восьмиконечных звезд квадрат и закрасим его желтым цветом. Вершины квадрата — центры звезд (*рис.* 10).

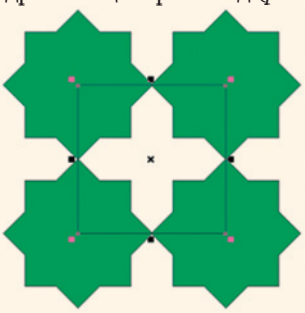


Рис. 10

13. На последнем шаге поместим квадрат на задний план. Для этого выбираем **Упорядочить | Порядок | На задний план страницы**. В результате получим изображение, показанное на *рис.* 11.

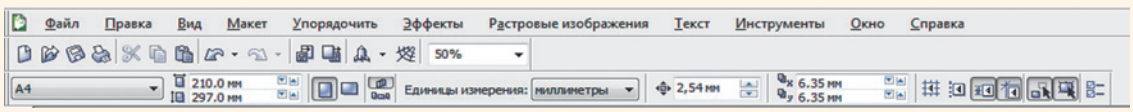


Рис. 4

Рис. 6

14. Чтобы получить еще один такой гирих, скопируем его и сместим вправо так, чтобы восьмиконечные звезды имели общие вершины. А потом вновь произведем заливку “крестов”.

Рис. 11

Этими изображениями можно заполнить всю плоскость. Так говорят математики, имея в виду бесконечность процесса построения. Практически же это сделать невозможно, поскольку плоскость не ограничена. Правда, малую часть плоскости все же заполнить можно (см. *рис.* 2 на с. 48).

Построенный гирих — самый простой среди тех, что встречаются в памятниках исламской культуры. Если посмотреть на причудливые орнаменты, украшающие мечети и минареты, то обнаружатся еще более сложные переплетения геометрических фигур. Все изображения выглядят “ритмическим” орнаментом — картинкой, обладающей трансляционной симметрией (см. *рис.* 1 на с. 48). Такой тип симметрии означает, что в орнаменте выбирается определенный фрагмент, который можно копировать, а затем совмещать “дубликаты” друг с другом параллельным переносом или поворотом. Рассматривая замысловатые восточные орнаменты, можно бесконечно открывать все новые и новые закономерности, спрятанные в них.

Построив один из гирихов, мы убедились, что это — сложный рисунок, образованный многократным наложением друг на друга геометрических фигур. Гирих выглядит как своеобразная геометрическая сетка, покрывающая архитектурную поверхность.

Подобными орнаментами в мусульманских странах оформлялись фасады и внутреннее убранство культовых зданий. В мечетях ими расписывался *михраб* — ниша, в которой имам читал молитву. Наложенные друг на друга геометрические фигуры в композиции гириха имели не только религиозное, но и мистическое значение. Они внушали верующим благоговение и трепет.

Не меньший трепет испытываем и мы, анализируя уникальные творения древних строителей. Кропотливое изучение гирихов открывает уникальные геометрические приемы, таинственные символы, неповторимую рукотворную красоту. А еще — историю целых народов, для которых орнамент был одним из способов выражения религиозных идей и национальных традиций.

От редакции

1. Гирих, изображение которого показано на *рис.* 2 в статье, можно получить также средствами текстового редактора Microsoft Word и растрового графического редактора Microsoft Paint или аналогичных программ. Перечислим основные этапы построения.

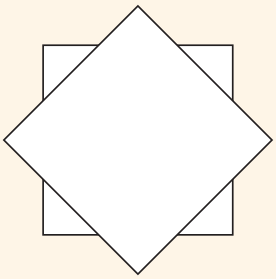
В текстовом редакторе:

1) нарисовать квадрат (удерживая нажатой клавишу Shift);

2) получить его копию (**Правка** | **Копировать** | **Вставить**);

3) повернуть копию на 45°;

4) совместить центры двух квадратов:



5) сгруппировать оба квадрата;

6) скопировать сгруппированное изображение (**Правка** | **Копировать**).

В графическом редакторе:

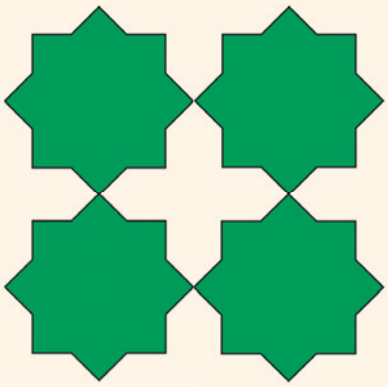
1) вставить изображение из буфера обмена (**Вставить**);

2) изменить размеры до минимума (по размерам вставленного изображения);

3) удалить внутренние наклонные линии (инструмент **Ластик**);

4) закрасить полученную восьмиконечную звезду (инструмент **Заливка**);

5–6) получить сначала две, а затем четыре звезды:



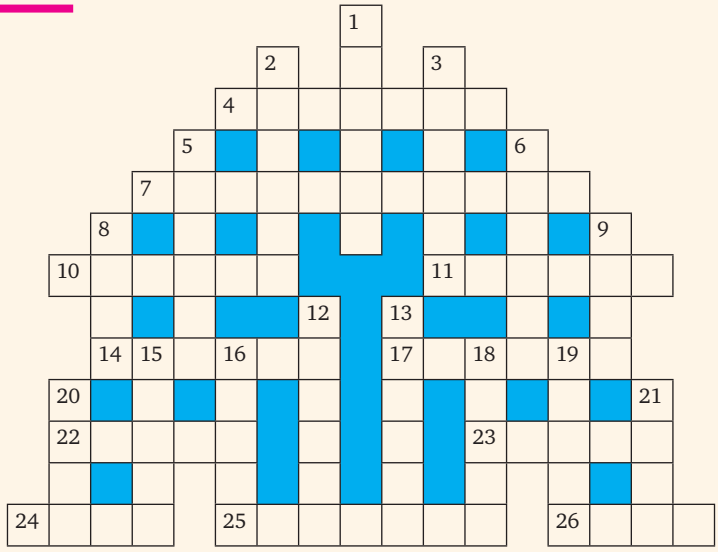
7) закрасить внутреннюю часть;

8) заполнить холст несколькими копиями полученного изображения.

2. Предлагаем читателям построить собственные графические изображения типа гириха и прислать их в редакцию.

“ЛОМАЕМ” ГОЛОВУ

Кроссворд



По горизонтали:

4. Накопитель на магнитной ленте в картридже.

7. Наука о законах, методах и способах накопления, обработки и передачи информации.

10. Один из двух режимов ввода текста в текстовых редакторах.

11. Порядок расстановки людей в одну шеренгу после упорядочения их по росту.

14. Инструмент графического редактора.

17. Одна из стадий компиляции программы.

22. Сторона прямоугольного треугольника.

23. Одна из разновидностей мультимедиа.

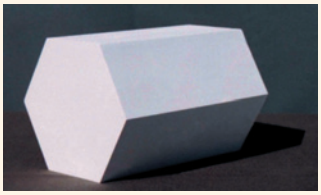
24.

25. Разновидность, модификация, версия.

26. Семейство совместимых друг с другом компьютеров.

По вертикали:

1.



2. Часть электронной таблицы.

3. Участок дорожки магнитного диска.

4. Номер элемента массива.

5. Устройство для ввода информации в персональный компьютер.

8. Совокупность данных на носителе, снабженная именем.

9. Группа линий электрических соединений, обеспечивающих передачу данных и управляющих сигналов между компонентами компьютера.

12. Физическая величина, имеющая только значение и не имеющая направления.

13. Значение переменной величины или константы логического типа (русский вариант).

15. Разработчик программы.

16. Фамилия второго космонавта.

18. Реакция объекта на воздействие или запрос.

19. Шифратор/дешифратор, устройство или программа, способная выполнять преобразование данных или сигнала.

20. Часть экрана, занимаемая приложением или документом Windows.

21. Язык программирования.

Бальзам

Три человека купили 24-литровый сосуд, полностью заполненный бальзамом. Позже они приобрели три пустых сосуда объемом 5, 11 и 13 л. Как они могли бы поделить бальзам на равные части, используя эти четыре сосуда?

“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).

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.

ЯПОНСКИЙ УГОЛОК

Два судоку

Решите, пожалуйста, две японские головоломки “судоку”:

1) простую:

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

2) сложную:

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

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

ДЛЯ ЭРУДИТОВ

Викторина “Олимпийские игры”

На каждый вопрос выберите один из четырех правильных ответов (но не наугад, а используя Интернет или другие источники информации).

1. Согласно легендам, Олимпийские игры организовал:

А. Зевс.

Б. Геракл.

В. Прометей.

Г. Аполлон.

2. Олимпиадой звали:

А. Любимую лошадь Александра Македонского.

Б. Жену Александра Македонского.

В. Сестру Александра Македонского.

Г. Мать Александра Македонского.

3. В античности Олимпийские игры проходили, естественно, в Олимпии. А располагалась они:

А. На острове Евбея.

Б. На полуострове Пелопоннес.

В. В Фессалии.

Г. В Македонии.

4. Участвовать в античных Олимпийских играх могли:

А. Только греки.

Б. Сначала только греки, а потом еще и римляне.

В. Все свободные (не рабы) мужчины.

Г. Любой приехавший на них.

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

ЗАДАЧНИК

Нашего полку прибыло!

Прежде всего редакция хочет представить новых читателей, приславших ответы на задания наших конкурсов:

— учащихся гимназии № 11 г. Норильска, Красноярский край, учитель **Береза Любовь Антоновна**;

— учеников средней школы села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Раушания Нурлимановна;**

— учащихся Центра образования “Школа Сотрудничества”, Москва, учитель **Денисенко Петр Леонидович**;

— учеников школы № 1108 г. Москвы, учитель **Маркова Ольга Валериевна**;

— учащихся школы № 9 г. Златоуста, Челябинская обл., учитель **Мусатова Ирина Борисовна**.

Желаем им успехов!

Игрушки

У одного мальчика-инопланетянина было 12 разных игрушек. Когда ему подарили еще 5, то общее число игрушек стало равно 21. Как такое могло быть?

Культурные развлечения

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.

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.

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?

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.

Литература

1. На досуге. Сборник занимательных задач. Ростов-на-Дону: Ростовское книжное изд-во, 1959.

Ответы, решения, разъяснения к заданиям, опубликованным в разделе “В мир информатики” ранее

14 ребусов

Ответы (соответствуют номерам ребусов): 1. ТАБЛИЦА. 2. ЗАПРОС. 3. ФИЛЬТР. 4. ОТЧЁТ. 5. ОТЧЁТ. 6. ОБЪЕКТ. 7. КОНСТРУКТОР. 8. МАСТЕР. 9. МАСТЕР. 10. ПОЛЕ. 11. ФОРМА. 12. СВЯЗИ. 13. СПИСОК. 14. СОРТИРОВКА.

Ответы представили:

— Аджоян Кристина и Бобровская Диана, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Аксененко Сергей, Перова Валентина, Иванов Иван и Яковлева Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Антоненко Кирилл, Беспалый Сергей, Блощук Александр, Истомина Полина и Мусатов Максим, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

— Антонов Денис, Губарев Максим, Зорин Сергей и Фадеев Петр, г. Рязань, школа № 44, учитель **Марцинкевич Е.Е.**;

— Ахматгалиева Диана и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Бабаян Безар, Игошев Константин, Краснова Диана и Смирнягина Александра, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

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

— Бобков Максим, Горшкова Алина, Воскресенский Денис, Карташова Виктория, Кротов Олег, Куркова Елизавета, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Грибанов Владлен, Дукач Светлана, Кабанов Вадим, Лысенко Екатерина, Овчинникова Елизавета, Соболев Иван и Яндушкин Виталий, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Гурьянова Дарья, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Иванова Полина, Кашпырева Алена, Олешова Дарья, Станкевич Александра и Шугакина Кира, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Калугин Руслан, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**;

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

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

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

— Крысанов Виктор, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Мальцева Виктория и Масленников Владислав, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Надельный Артем и Петров Юрий, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Новиков Сергей и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Токарева Алина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Трептау Татьяна, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**

Кроссворд (опубликованный в февральском выпуске)

Ответы

По горизонтали: 5. Операнд. 7. Цилиндр. 8. ИЛИ. 13. Калькулятор. 15. База. 16. Овал. 17. Теорема. 18. Замена. 19. Зрение. 21. Отвод. 27. Яркость. 28. Фортран. 29. Закладка. 30. Страница. 31. Дек. 32. Скан. 33. Стяг.

По вертикали: 1. Логика. 2. Три. 3. Бит. 4. Правка. 6. Диск. 7. Цикл. 9. Сканер. 10. Плата. 11. Отказ. 12. Пробел. 14. Управление. 18. Запятая. 20. Единица. 21. Ольга. 22. Дефис. 23. Шкала. 24. Осадок. 25. Прерия. 26. Крона.

Ответы прислали:

— Аксененко Сергей и Яковлева Ирина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Алейникова Анастасия, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Алейникова Г.Н.**;

— Ахматгалиева Диана и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Березин Василий, Сапыгина Александра и Хомякова Анна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Бобровская Диана, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Выгина Ксения и Литовченко Яна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Грибанов Владлен, Дукач Светлана, Кабанов Вадим, Лысенко Екатерина, Овчинникова Елизавета, Соболев Иван и Яндушкин Виталий, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Гуцал Анна и Любимова Ирина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Дегтярь Анатолий и Новиченко Владимир, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Ермакова Мария и Семенюк Евгений, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Землянская Надежда, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

— Иванов Владислав, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

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

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

— Костикова Татьяна, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

— Леоненко Степан, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Новиков Сергей, Полосина Анна и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Рахимова Алина, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Смирнягина Александра, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Трептау Татьяна, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**

Задача “Четыре гнома”

Напомним, что следовало определить, правдивым (всегда говорит правду) или вруном (всегда врет) является каждый из четырех гномов — Беня, Веня, Женя и Сеня, если между ними состоялся такой разговор: Беня — Вене: “Ты врун”; Женя — Бене: “Сам ты врун”; Сеня — Жене: “Да оба они вруны, — (подумав), — впрочем, ты тоже”.

Решение

Пронумеруем реплики гномов:

Беня — Вене: “Ты врун” — № 1;

Женя — Бене: “Сам ты врун” — № 2;

Сеня — Жене (про Беню и Веню): “Да оба они вруны, впрочем, ты тоже” — № 3.

Рассмотрим варианты.

1. Допустим, что правду говорит Беня. Тогда Веня — врун (реплика № 1) и Женя — врун (реплика № 2). Тогда слова Сени о Жене (“Ты тоже”) являются правдой, а о Бене, что тот врун, но Сеня не может быть правдивым, так как он сказал про Беню то, что противоречит допущению.

2. Предположим, что правду говорит Женя. Тогда Беня — врун (реплика № 2), а Веня — правдивый (реплика № 1). Сеня же, сказав Жене, что тот врун, врун сам. Но тогда он сказал правду про врун Беню, чего быть не может.

3. Допустим, что правду говорит Сеня (три остальных гнома — вруны). Но тогда врун Беня сказал про Веню правду, что приводит к противоречию.

4. Предположим, что правдивый — Веня. Тогда Беня — врун (реплика № 1), а Женя — правдивый (реплика № 2). Тогда слова Сени говорят о том, что он врун.

Итак, ответ: правду всегда говорят Веня и Женя, остальные гномы — вруны.

Ответы прислали:

— Абрамов Игорь, Полосина Анна и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Аджоян Кристина, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Аллахвердиев Николай, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Антипов Владимир, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Артюхина Татьяна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Ахматгалиева Диана, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Букина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Воскресенский Денис и Манукян Григорий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Вяткин Владимир, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Гурьянова Дарья, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

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

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

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

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

— Шамаева Наталья, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Читателей, представивших правильный ответ (их фамилии мы оставляем в тайне ☺), мы решили наградить дипломом. Молодцы!

Викторина (опубликованная в февральском выпуске)

Напомним, что требовалось, используя информацию из Интернета или из других источников, выбрать правильный ответ из трех предложенных вариантов.

Ответы

1. Лайнер, останки которого приобрели защиту Конвенции ЮНЕСКО 2001 года об охране подводного культурного наследия, — “Титаник” (вариант *в*).

2. Бытовое прозвище, которое получил один из небоскребов, ставших визитной карточкой исторического ядра Нью-Йорка — Манхэттена, — “Утюг” (вариант *в*).

3. Северный город, о котором шла речь в вопросе, — Воркута (вариант *а*).

4. Сейф, о котором говорилось в вопросе, принадлежал Свердлову (вариант *а*).

5. Ирландскому актеру Ричарду Харрису, сыгравшему Альбуса Дамблдора в фильме “Гарри Поттер и философский камень”, пригрозила бессрочным бойкотом его внучка, после чего он согласился на съемки (вариант *в*).

Ответы прислали:

— Абрамов Игорь и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Алейникова Анастасия, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Алейникова Г.Н.**;

— Ахматгалиева Диана и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Воронова Анжелика и Хомутов Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Воскресенский Денис, Кротов Олег, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Глазова Елизавета, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Гурьянова Дарья, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Дубинин Владимир и Мякишева Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Заева Кристина, Республика Башкортостан, г. Уфа, школа № 54 (Центр дистанционного обучения), учитель **Искандарова А.Р.**;

— Игошев Константин, Краснова Диана и Смирнягина Александра, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Кабанова Нина и Самсонова Ксения, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

— Лабкова Валерия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Мальцева Виктория, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Рыбакова Мария, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

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

— Ширяев Денис, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**

Задача “Что начертил учитель?”

Напомним условие: “Учитель начертил на классной доске четырехугольник. Вася утверждал, что это квадрат. Таня считала, что четырехугольник — трапеция. Армен думал, что на доске изображен ромб. Дима назвал четырехугольник параллелограммом. Выслушав каждого и обстоятельно изучив свойства четырехугольника, учитель установил, что ровно три из четырех утверждений истинны и ровно одно утверждение ложно. Какой четырехугольник начертил учитель на классной доске?”

Решение

Составим таблицу, в которой рассмотрим 4 варианта четырехугольника, который мог быть начерчен на доске, и истинность или ложность высказываний учеников для каждого варианта.

Примечание. Ромб является трапецией и параллелограммом, но не является квадратом, а квадрат ромбом является (см. мнение Армена).

Ответ — учитель нарисовал ромб.

Ответы представили:

— Аллахвердиев Николай, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Артюхина Татьяна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Ахматгалиева Диана и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Бобровская Диана, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Букина Елена, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Волков Владимир и Глушаков Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Вяткин Владимир, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Грибанов Владлен, Дукач Светлана, Кабанов Вадим, Лысенко Екатерина, Овчинникова Елизавета, Соболев Иван и Яндушкин Виталий, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Гурьянова Дарья, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

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

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

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

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

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

— Туголукова Ольга, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Цыплаков Евгений, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Заметим, что в нескольких ответах ошибочно утверждалось, что “квадрат не является трапецией”. В одном из ответов использовано “оригинальное” понятие — “равносторонний прямоугольник” ☺.

Если на доске начерчен:	то мнение Васи (“квадрат”)	то мнение Тани (“трапеция”)	то мнение Армена (“ромб”)	то мнение Димы (“параллелограмм”)	Число истинных мнений
1. Квадрат □	истинно	истинно	истинно	истинно	4
2. Трапеция ⏢	ложно	истинно	ложно	ложно	1
3. Ромб ◊	ложно	истинно	истинно	истинно	3
4. Параллелограмм ▱	ложно	истинно	ложно	истинно	2

Задача “Запутанный ответ”

Напомним, что следовало определить, как называется книга, которую прочитал ученик, если он ответил так: “В название книги входят четыре буквы. Если эти буквы заменить числами, которые они по порядку занимают в русском алфавите, то эти числа будут иметь следующие свойства: сумма их будет равна 40; первое число равно третьему; сумма второго и четвертого составляет половину третьего, а разность между четвертым и вторым равна одной восьмой первого числа”.

Ответ: “Овод” (автор романа — Этель Лилиан Войнич).

Правильный ответ прислали:

— Бобков Максим, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Волков Владимир и Глушаков Андрей, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Глазова Елизавета, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Иванова Полина, Кашпырева Алена, Олешова Дарья, Станкевич Александра и Шугакина Кира, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Карасева Евгения, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

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

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

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

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

— Мякишева Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Рухтин Дмитрий, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Самсонова Ксения и Харламов Виталий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Туголукова Ольга и Хутилашвили Виктория, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Хорькова Анна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задача “Лицеисты”

Напомним условие: “В лицее информационного профиля, где учатся больше 225, но меньше 245 учеников, часть учеников являются отличниками, а остальные хорошистами. После контрольной работы по информатике 2/7 отличников стали хорошистами, а хорошисты так и остались хорошистами за исключением одного человека, который стал троечником. При этом хорошистов и отличников стало поровну. Сколько учеников могло быть в лицее?”

Решение

Пусть V — общее количество лицеистов,

OT — количество отличников до контрольной работы,

X — количество хорошистов до контрольной.

После контрольной количество отличников стало равно $OT - 2/7 \cdot OT$, а количество хорошистов — $V - (OT + 2/7 \cdot OT) - 1$, причем эти значения равны:

$$OT - 2/7 \cdot OT = V - OT + 2/7 \cdot OT - 1.$$

Последнее равенство можно записать так:

$$7OT - 2OT = 7V - 7OT + 2OT - 7,$$

откуда

$$OT = \frac{7(V-1)}{10}.$$

Так как значение OT должно быть целым, то числитель дроби в правой части должен быть кратен 10, или значение V — на 1 больше некоторого количества десятков. Из интервала, указанного в условии, этому требованию удовлетворяют значения 231 и 241.

Ответ: общее число лицеистов составляло 231 или 241.

Ответы прислали:

— Бобков Максим, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Волков Владимир, Глушаков Андрей и Савинцева Светлана, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Глазова Елизавета, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Живило Андрей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Карасева Евгения, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

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

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

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

— Мякишева Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Полосина Анна и Хромченкова Елизавета, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Хорькова Анна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Задача “Рубик рубит кубик”

Напомним, что следовало определить, какое минимальное число раз изобретатель Рубик должен ударить топором, чтобы разрубить кубик размером 3 × 3 × 3 на маленькие кубики размером 1 × 1 × 1 (при разрубании разрешены наложения кусков кубика друг на друга).

Ответ

Потребуется шесть разрубаний — по плоскостям, которые соответствуют рядам кубиков размером 1 × 1 × 1 (на рисунке ниже они выглядят черными линиями между кубиками).

Менее чем шестью разрубаниями это сделать нельзя по следующим причинам. У куба шесть граней. Каждое разрубание означает проведение плоскости, то есть при каждом разрубании появляется не более одной новой грани куба. Следовательно, чтобы вырубить маленький кубик в самом центре большого куба (это единственный кубик, у которого вначале нет ни одной готовой грани), потребуются шесть разрубаний.

Ответы прислали:

— Абрамова Наталья и Бойкова Ксения, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Аджоян Кристина, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Александрова Ирина и Смольякова Светлана, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Антипов Владимир, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Ахматгалиева Диана и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Барановская Наталья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Бобков Максим, Воскресенский Денис, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Васильева Тамара и Мякишева Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Евграфов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

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

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

— Стороженко Степан, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Новая задача

Какое минимальное число разрубаний потребуется, чтобы разрубить на маленькие кубики размером 1 × 1 × 1 кубик размером 4 × 4 × 4 (при разрубании разрешены наложения кусков кубика друг на друга)?

Задача “Четыре девочки”

Ответ

Имена девочек в порядке возрастания роста Оля, Маша/Лена, Таня (по условию Маша и Лена имеют одинаковый рост).

Ответы прислали:

— Абрамов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Авдеева Любовь, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Аджоян Кристина и Бобровская Диана, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Васнин Иван, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Тарасова Г.А.**;

— Голубева Ирина и Кузина Елена, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Грибанов Владлен, Дукач Светлана, Кабанов Вадим, Лысенко Екатерина, Овчинникова Елизавета, Соболев Иван и Яндушкин Виталий, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Гурьянова Дарья, МБОУ “Кадетская школа”, Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Дымова Ксения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Киселев Владислав, Никулин Данила, Печников Максим (в прошлом учебном году ученик 1-го класса) и Строкин Константин, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

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

— Кошкина Екатерина, средняя школа села Чистопольские Выселки, Республика Татарстан, Чистопольский р-н, учитель **Валиева Р.Н.**;

— Ливанова Алина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Митрохина Клавдия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Шацкая Елена, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Юнцова Маргарита, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**

Задача "В санатории"

Напомним условие: "В санатории на берегу моря отдыхают отец, мать, сын и две дочери. До завтрака члены семьи купаются в море, причем известно, что если отец утром отправляется купаться, то с ним обязательно идут мать и сын; если сын идет купаться, то первая дочь идет с ним; вторая дочь купается тогда и только тогда, когда купается мать; каждое утро купается по крайней мере один из родителей. Если в воскресенье утром среди купающихся членов семьи была лишь одна из дочерей, то кто в это утро ходил на море?"

Решение

Обозначим мать, отца, сына, первую и вторую дочерей, соответственно, М, О, С, Д1 и Д2. С использованием этих обозначений на основе имеющей информации можно составить схему:

$$\begin{array}{c} О \to М + С \\ \downarrow\downarrow \\ Д1\ Д2 \end{array}$$

Из нее следует, что так как по условию среди купающихся членов семьи была лишь одна из дочерей, то мать и сын одновременно в воскресенье не купались (то есть на море в этот день не было и отца).

Предположим, что на море сын был без матери. Тогда с ним в то утро купалась только первая дочь, а родителей не было, что противоречит условию (каждое утро купается по крайней мере один из родителей).

Остается, что на море ходила мать без сына, а с ней обязательно должна была быть вторая дочь.

Ответ: в воскресенье ходили купаться мать и вторая дочь.

Ответы представили:

— Аджоян Кристина и Бобровская Диана, средняя школа рабочего поселка Пинеровка, Саратовская обл., Балашовский р-н, учитель **Пичугин В.В.**;

— Алтухова Людмила и Сотникова Мария, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Антипов Владимир, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

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

— Воскресенский Денис, Тананаева Анастасия, Тананаева Ксения и Телегин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

— Гурьянова Дарья, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Донцов Юрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Иванова Полина, Кашпырева Алена, Олешова Дарья, Станкевич Александра и Шугакина Кира, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Казакова Татьяна, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

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

— Митина Наталья, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

Задачи "Две банки с бактериями"

Задача 1

Ответ: можно — банка будет заполнена наполовину через 59 минут 59 секунд (каждую секунду число бактерий удваивается; по условию через 60 минут банка была полной).

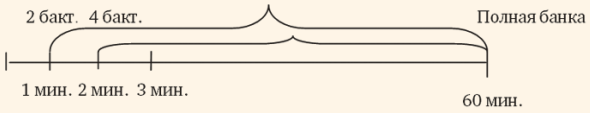
Задача 2

Ответы:

а) через 59 минут;

б) через 58 минут.

Обоснование показано на схеме:



Ответы прислали:

— Авдеева Любовь, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

— Голубева Ирина и Кузина Елена, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Кабардин Демид, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Казакова Ульяна и Мащенко Наталья, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

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

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

— Ливанова Алина, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Лякина Мария, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

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

— Митрохина Клавдия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

В одном из ответов утверждалось, что задачу 1 решить нельзя, так как “неизвестно количество бактерий” ☺.

Числовой ребус

Напомним, что следовало решить числовой ребус:

ABBA + A + B = CDDA

Решение

Так как числа **ABBA** и **CDDA** оканчиваются на одинаковую цифру, то на 0 оканчивается сумма различных цифр **A** и **B**. Поэтому **A** = **B** = 10 и **ABBA** + 10 = **CDDA**. Прибавление единицы во втором разряде изменило цифры в двух высших разрядах, следовательно, **B** = 9, **A** = 1, **D** = 0 и **C** = 2.

Ответ: 1991 + 1 + 9 = 2001.

Правильный ответ прислали:

— Аристов Петр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Бобков Максим, Воскресенский Денис, Кротов Олег, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Грибанов Владлен, Дукач Светлана, Кабанов Вадим, Лысенко Екатерина, Овчинникова Елизавета, Соболев Иван и Яндушкин Виталий, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дерюгина Мария, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Донцов Юрий, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Зорина Елена, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Иванова Анна, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Игошев Константин и Краснова Диана, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Калинина Марина, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Китова Мария, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

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

— Лизунов Павел, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

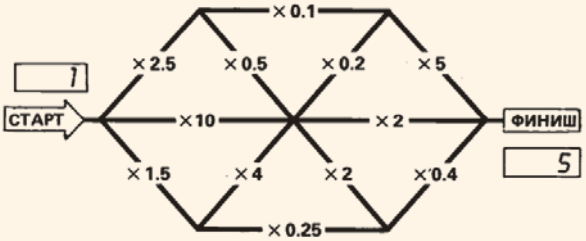
— Наделяев Денис, средняя школа поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**;

— Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Тупицына Мария, г. Пенза, школа № 512, учитель **Гаврилова М.И.**

Головоломка “Лабиринт из чисел”

Напомним, что нужно было пройти по ребрам графа, выполняя расчеты так, чтобы в результате получить число 5.



Ответы

Головоломка имеет два варианта решения

1)

0. Начальное значение 1.
1. × 2,5 = 2,5.
2. × 0,5 = 1,25.
3. × 4 = 5.
4. × 0,25 = 1,25.
5. × 2 = 2,5.
6. × 2 = 5.

2)

0. Начальное значение 1.
1. × 10 = 10.
2. × 4 = 40.
3. × 0,25 = 10.
4. × 2 = 20.
5. × 0.5 = 10.
6. × 0.1 = 1.
7. × 5 = 5.

Правильные ответы прислали:

— Аристов Петр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Воскресенский Денис, Кротов Олег, Тананаева Анастасия, Тананаева Ксения и Телегин Дмитрий, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Дерюгина Мария, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Лизунов Павел, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Мацишина Валерия, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Новожилова Светлана, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Хутилашвили Виктория, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Яковлева Марина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**

Две японские головоломки "судоку" (опубликованные в февральском выпуске)

Правильные ответы представили:

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

— Ахматгалиева Диана, Мацишина Валерия и Танасюк Артем, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Ахметова Маргарита, Берсенев Федор, Богомолов Георгий, Гайсина Галия, Зорина Елена, Прусакова Евгения и Рахимова Алина, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Беляева Мария, Костромская обл., Буйский р-н, г.п.п. Чистые Боры, школа № 1, учитель **Васнина О.В.**;

— Бобков Максим, Воскресенский Денис, Кротов Олег, Тананаева Анастасия и Тананаева Ксения, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Галиева Карина, Мещерякова Виктория, Сафиуллин Марат, Шавалеева Виктория и Шагинуров Эмиль, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**;

— Гурьянова Дарья, МБОУ "Кадетская школа", Республика Татарстан, г. Чистополь, пгт Крутая Гора, учитель **Валиева Р.Н.**;

— Директорова Анастасия, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Иванова Анна и Фоменко Анастасия, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

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

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

— Краснова Диана и Смирнягина Александра, Свердловская обл., г. Ревда, школа № 10, учитель **Игошева А.А.**;

— Кужелев Александр, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Кузьменко Иван и Полев Владислав, Красноярский край, г. Норильск, гимназия № 11, учитель **Береза Л.А.**;

— Лещенко Сергей и Корнилов Виктор, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Марков Федор и Пилипчук Иван, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Массольд Валерия, средняя школа села Ириновка, Новобурасский р-н Саратовской обл., учитель **Брунов А.С.**;

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

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

— Шигаев Никита, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

— Юмашев Константин, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**

Задача "Чемпионат школы"

Напомним, что в задаче шла речь о первенстве школы среди 11-х классов в десяти видах спорта, которое разыграли две команды. За победу команда получала 4 очка, за ничью — 2, за проигрыш — 1 очко. Победила команда 11-го "Б" класса, причем вместе обе команды набрали 46 очков. Предлагалось ответить на вопросы:

1) сколько было ничьих?

2) можно ли определить, сколько очков набрала каждая из двух команд?

Решение

Когда в каком-то виде спорта одна из команд побеждает, то оба соперника набирают вместе $4 + 1 = 5$ очков, если же соревнование закончилось вничью — $2 + 2 = 4$ очка, то есть на 1 меньше.

Если бы все 10 соревнований закончились победой одной из команд, то общая сумма набранных очков составила бы $10 \cdot 5 = 50$. Так как каждая ничья уменьшает это число на 1, то при общей сумме 46 очков количество ничьих равно 4. Значит, победой одной из команд закончились соревнования по шести видам спорта. Но сколько раз побеждала та или иная команда (то есть сколько очков она набрала) установить не представляется возможным.

Ответы прислали:

— Аристов Петр, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Бобков Максим, Владимирская обл., г. Струнино, школа № 11, учитель **Волков Ю.П.**;

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

— Марков Федор и Пилипчук Иван, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

— Новожилова Светлана, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**

Разбор решений числового ребуса «“СТАЙКА” из шести “ПТИЧЕК”» и задачи “Кучки монет” будет проведен в следующем выпуске “В мир информатики”.

В заключение заметим, что редакция публикует списки всех приславших ответы, в том числе и с неточностями и ошибками. В связи с этим предлагаем читателям сравнивать публикуемые ответы с собственными и анализировать возможные ошибки.

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.

Благодаря читателей, представивших ответ:

— Коростелева Иннокентия и Маруна Виталия, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Кузнецову Юлию, г. Ярославль, школа № 33, учитель **Ярцева О.В.**;

— Стороженко Степана, г. Пенза, школа № 512, учитель **Гаврилова М.И.**,

приведем начало решения.

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:

$$r = a\ ?\ b;$$
$$p = a\ ?\ b,$$

где знаками вопроса обозначены логические операции, которые применяются к числам (см. приложение 1 к статье “Системы счисления и электронные таблицы” в “Информатике” № 3/2013):

1) **AND** (русский вариант — **И**);

2) **OR** (**ИЛИ**);

3) **XOR** (от англ. *eXclusive* ***OR*** — **исключающее ИЛИ**).

Эти операции выполняются в процессоре компьютера (поэтому их называют также логическими командами) над числами, представленным 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.

Предлагаем читателям найти соответствующие команды и прислать ответ в редакцию. Решите также 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, то $r_1 = 0$, $r_2 = 1$, если 11 — то $r_1 = 1$, $r_2 = 0$.

Операции сравнения (условия) не использовать.

ПРИ1Ы

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?

ПОИСК ИНФОРМАЦИИ

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?

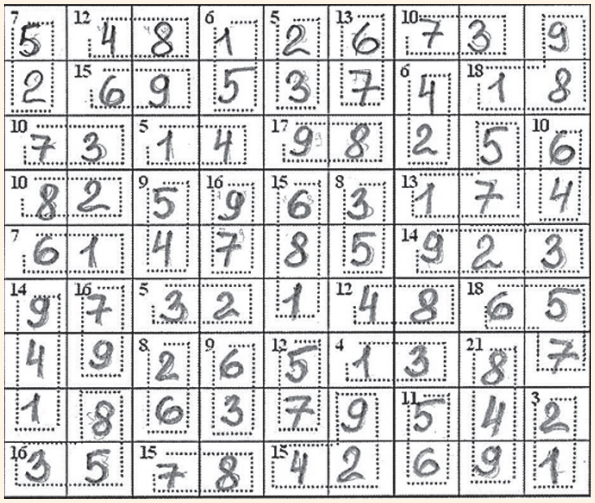
Ответы присылайте в редакцию (можно отвечать не на все вопросы).

ВНИМАНИЕ! КОНКУРС

Итоги конкурса № 100 “Юбилейный”

Напомним, что предлагалось решить японскую головоломку “судоку”, в которой 2, 3 или 4 клетки были объединены в блоки, и сумма цифр в клетках блока должна быть равна числу, указанному в углу блока.

Решение показано на рисунке:



Победителями конкурса признаны:

— Абрамов Алексей, средняя школа поселка Новопетровский Московской обл., учитель **Артамонова В.В.**;

— Гайсина Галия, Республика Башкортостан, г. Уфа, школа № 18, учитель **Искандарова А.Р.**;

— Галиева Карина, Сафиуллин Марат и Шагинуров Эмиль, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**;

— Костина Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Кузьменко Иван и Полев Владислав, Красноярский край, г. Норильск, гимназия № 11, учитель **Береза Л.А.**;

— Угольникова Анна, Москва, Центр образования “Школа сотрудничества”, учитель **Денисенко П.Л.**;

— Фоменко Анастасия, г. Воронеж, школа № 5 им. К.П. Феоктистова, учитель **Чернышева И.А.**;

— Хорькова Анна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Шигаев Никита, Челябинская обл., г. Златоуст, школа № 9, учитель **Мусатова И.Б.**;

— Якушев Сергей, Москва, школа № 1108, учитель **Маркова О.В.**

Все они будут награждены дипломами. Поздравляем!

Конкурс № 101

В качестве задания этого конкурса предлагаем решить две задачи:

1) задачу “Найти количество единиц”, опубликованную в рубрике “Крепкий орешек”;

2) задачу “Сложение трех двоичных цифр”:

Сложение трех двоичных цифр

При сложении трех двоичных цифр в общем случае получается двузначное значение *pr* (возможно, с начальным нулем или состоящее из двух нулей). Запишите формулы, определяющие значение цифр *r* и *p* в зависимости от значения двоичных цифр *a*, *b* и *c*. Операции сравнения (условия) не использовать.

Конкурс № 102

В качестве задания этого конкурса предлагаются к решению числовые ребусы, опубликованные в рубрике «“Ломаем” голову» (можно решать не все ребусы).

Ответы отправьте в редакцию до 25 сентября 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.

Срок представления ответов на другие задания, предложенные в этом выпуске, — 15 сентября. Мы ждем также решение задачи “Восемь семейных пар” (“Информатика” № 5/2013) и ответов на задания, предложенные в статье “О чем рассказали старые кассеты”, опубликованной в июньском выпуске журнала.

2013/14 учебный год

Школа цифрового века
открыта!

- **Выбирайте журналы**

  Теперь их двадцать четыре. Новые издания, «ОБЖ» и «Школа для родителей», ждут своих читателей

- **Выбирайте курсы**

  Программа «Навыки профессиональной и личной эффективности» каждый месяц пополняется новым модулем

- **Выбирайте вебинары**

  Авторы модульных курсов — в прямом эфире каждый месяц

- **Выбирайте литературу по специальности**

  В книжной программе больше трех тысяч наименований. Ассортимент бесплатной литературы увеличивается каждый месяц

digital.1september.ru

Прием заявок на участие в проекте продолжается

С НОВЫМ УЧЕБНЫМ ГОДОМ

Оргкомитет общероссийского проекта «Школа цифрового века»