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

ИНФОРМАТИКА

4
Не хотите давать Нобелевскую премию по информатике?
Тогда мы идем к вам ☺

16
XYZ в WWW
Не “нативно”, так плагинами

22
Ударники производства,
или Колоссы индустрии программирования

внутри номера CD и код доступа к электронной версии

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

НА ОБЛОЖКЕ

Шумерские “шары-конверты”, которым более 5000 лет, возможно, являются первыми устройствами хранения информации в истории человечества. Причем весьма сложными устройствами. Внутри шаров запечатаны токены различной формы. Отпечатки токенов находятся на поверхности шаров. Таким образом, не было необходимости разбивать носители для того, чтобы узнать, что у них внутри. Очень интересное и технологичное решение!

В НОМЕРЕ

3 ПАРА СЛОВ
- Мы, люди, имеем право и не обязаны

4 ИСТОРИЯ НАУКИ
- За разработку разномасштабных моделей сложных химических систем

12 СЕМИНАР
- “Колокола” случайных чисел

16 ИНТЕРНЕТ-ТЕХНОЛОГИИ
- Трехмерный веб

22 ЯЗЫКИ ПРОГРАММИРОВАНИЯ
- Индустрия: черепаха или киты

36 МЕТОДИКА
- Навигационная панель — “лента” для учебного проекта

42 ВНЕКЛАССНАЯ РАБОТА
- Логика, творчество, интеллект. Об опыте проведения конкурса по математике и информатике

48 ЗАНИМАТЕЛЬНЫЕ МАТЕРИАЛЫ ДЛЯ ПЫТЛИВЫХ УЧЕНИКОВ И ИХ ТАЛАНТЛИВЫХ УЧИТЕЛЕЙ
- “В мир информатики” № 192

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

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

ИНФОРМАТИКА

ПОДПИСНЫЕ ИНДЕКСЫ
по каталогу “Почта России”: 79066 — бумажная версия, 12684 — электронная версия

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 20 195 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

ИЗДАТЕЛЬСКИЙ ДОМ “ПЕРВОЕ СЕНТЯБРЯ”
Главный редактор:
Артем Соловейчик (генеральный директор)
Коммерческая деятельность:
Константин Шмарковский (финансовый директор)
Развитие, IT и координация проектов:
Сергей Островский (исполнительный директор)
Реклама, конференции и техническое обеспечение Издательского дома:
Павел Кузнецов
Производство:
Станислав Савельев
Административно-хозяйственное обеспечение:
Андрей Ушков
Педагогический университет:
Валерия Арсланьян (ректор)

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА
Английский язык – А.Громушкина
Библиотека в школе – О.Громова
Биология – Н.Иванова
География – О.Коротова
Дошкольное образование – Д.Тюттерин
Здоровье детей – Н.Сёмина
Информатика – С.Островский
Искусство – О.Волкова
История – А.Савельев
Классное руководство и воспитание школьников – М.Битянова
Литература – С.Волков
Математика – Л.Рослова
Начальная школа – М.Соловейчик
Немецкий язык – М.Бузоева
ОБЖ – А.Митрофанов
Русский язык – Л.Гончар
Спорт в школе – О.Леонтьева
Технология – А.Митрофанов
Управление школой – Е.Рачевский
Физика – Н.Козлова
Французский язык – Г.Чесновицкая
Химия – О.Блохина
Школа для родителей – Д.Тюттерин
Школьный психолог – И.Вачков

УЧРЕДИТЕЛЬ:
ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011
в Министерстве РФ по делам печати
Подписано в печать:
по графику 15.10.2013,
фактически 15.10.2013
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

Мы, люди, имеем право и не обязаны

Случается, что в тех или иных ситуациях я не могу не оказать психологическую поддержку своим коллегам, иногда — хорошо знакомым. Делать этого, строго говоря, не следует. Есть регламент, поясняющий, как надо поступать в подобных случаях. Но жизнь не всегда вписывается в регламенты.

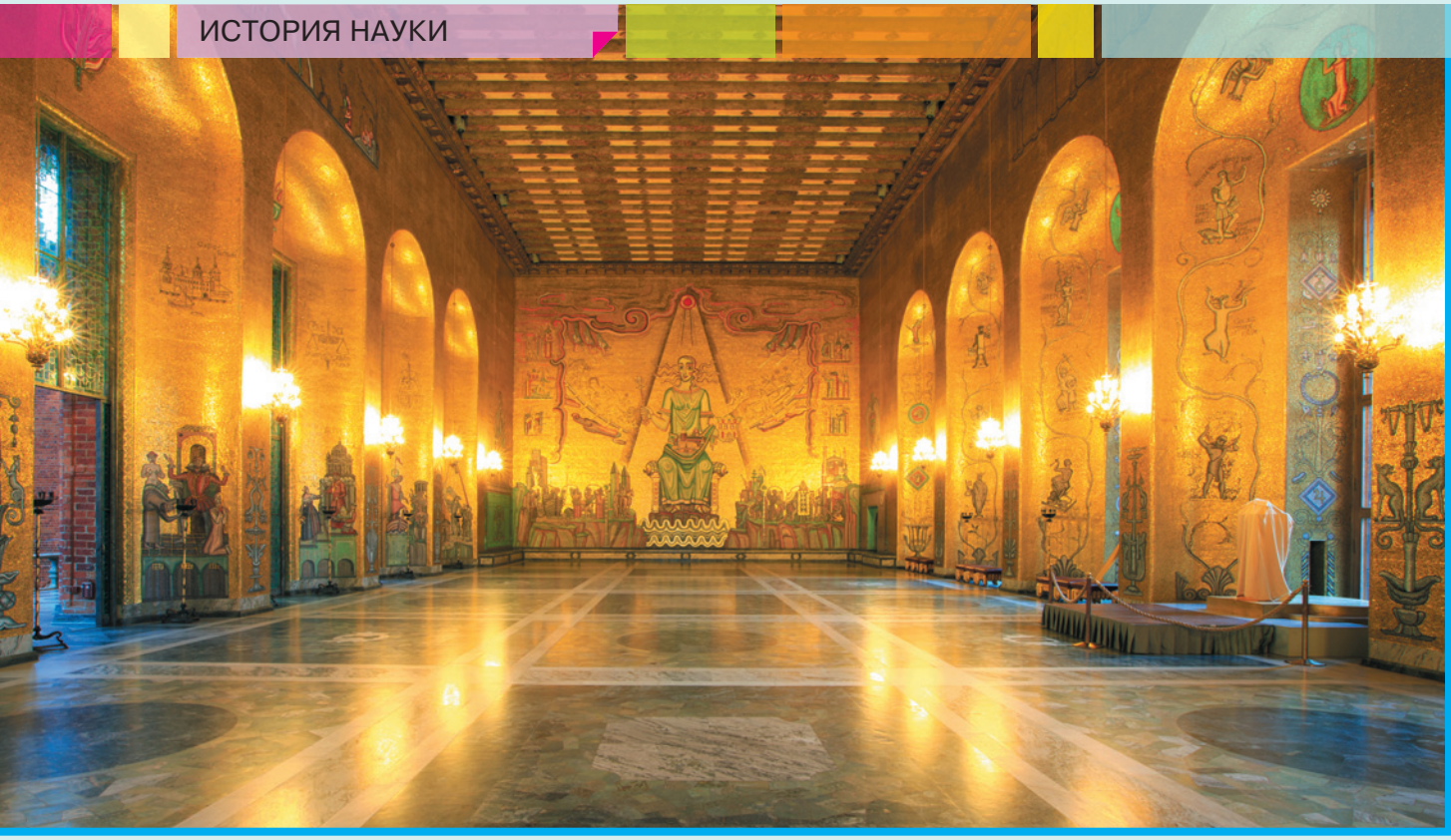
К счастью, есть один “листочек”, который нередко работает так, что после него и работы-то никакой не требуется. Его вообще полезно иметь с собой и периодически почитывать. Кстати, это сильно сокращенная версия оригинального документа — билля о правах личности, подготовленного американской ассоциацией психологов. Полный тест билля вы при желании легко найдете в Интернете в различных переводах.

Два замечания. Первое — это просто текст, который написали люди. Вы имеете право понимать и принимать его сколь угодно критично и по собственному разумению. Вы имеете право не принимать его вовсе. Оригинальный текст был написан на английском, и имеются различные переводы, в ряде случаев отражающие вкусы или личные особенности переводчика. Второе — обычно в конце этого текста содержится приписка: “Помните, что все другие люди имеют точно такие же права и «не обязанности»”. Мне видится, что эту фразу можно переместить в начало.

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

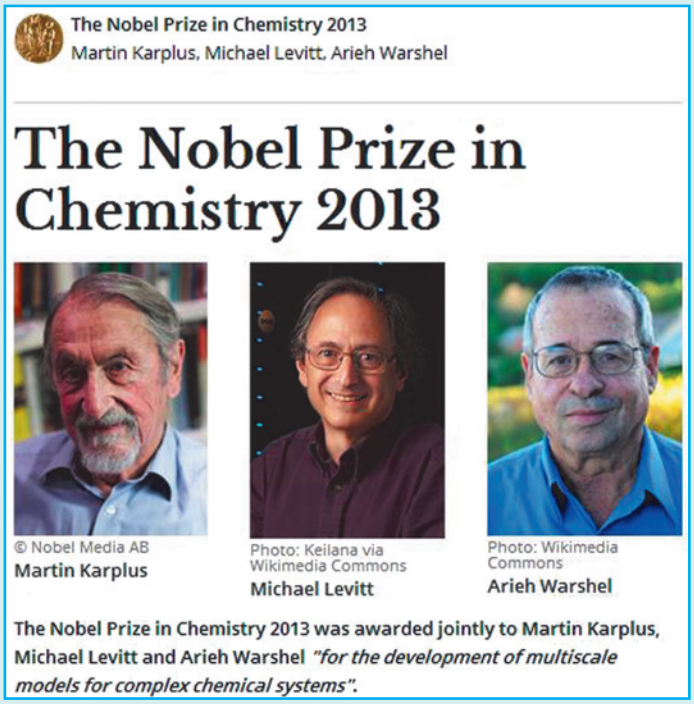
Вы имеете право:
…иногда ставить себя на первое место;
…просить о помощи и эмоциональной поддержке;
…протестовать против несправедливого обращения или критики;
…иметь свое собственное мнение или убеждения;
…совершать ошибки, пока не найдете правильный путь;
…предоставлять людям право самим решать свои проблемы;
…говорить: “Спасибо, нет”, “Извините, нет”;
…не обращать внимания на советы окружающих;
…побыть одному, даже если другим хочется общества;
…иметь свои собственные, какие угодно чувства, независимо от того, понимают ли их окружающие;
…менять свои решения или изменять образ действий;
…добиваться перемены договоренности, которая вас не устраивает.

Вы никогда не обязаны:
…быть безупречным на 100%;
…следовать за всеми;
…делать приятное неприятным вам людям;
…любить людей, приносящих вам вред;
…извиняться за то, что вы были самим собой;
…выбиваться из сил ради других;
…чувствовать себя виноватым за свои желания;
…мириться с неприятной вам ситуацией;
…жертвовать своим внутренним миром ради кого бы то ни было;
…сохранять отношения, ставшие оскорбительными;
…делать больше, чем вам позволяет время;
…делать что-то, что вы в самом деле не можете сделать;
…выполнять неразумные требования;
…отдавать что-то, что вам на самом деле отдавать не хочется;
…нести на себе тяжесть чьего-то неправильного поведения;
…отказываться от своего “Я” ради чего бы то ни было и кого бы то ни было.



Голубой зал ратуши Стокгольма, в котором проводится традиционный Нобелевский банкет. Фото с сайта *vichie81/Shutterstock.com*

За разработку разномасштабных моделей сложных химических систем


The Nobel Prize in Chemistry 2013
Martin Karplus, Michael Levitt, Arieh Warshel

The Nobel Prize in Chemistry 2013

© Nobel Media AB
Martin Karplus

Photo: Keilana via Wikimedia Commons
Michael Levitt

Photo: Wikimedia Commons
Arieh Warshel

The Nobel Prize in Chemistry 2013 was awarded jointly to Martin Karplus, Michael Levitt and Arieh Warshel *"for the development of multiscale models for complex chemical systems"*.

Рис. 1. С сайта *http://www.nobelprize.org/nobel_prizes/chemistry/laureates/2013/*

М.А. Ройтберг,
Институт математических проблем биологии РАН, заведующий лабораторией; редактор сайта *ege-go.ru*

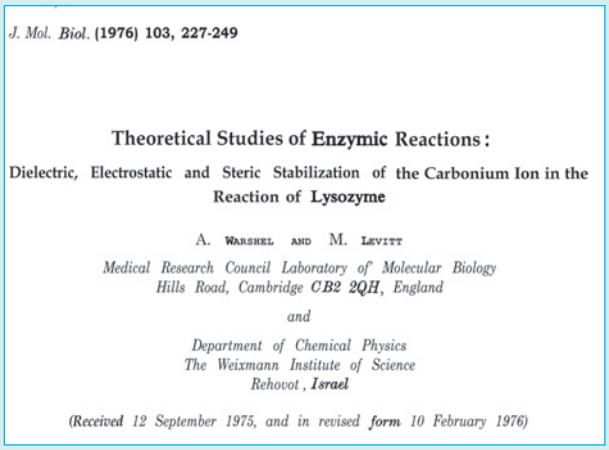
Нобелевская премия по химии 2013 года была присуждена Мартину Карплюсу (Martin Karplus), Майклу Левитту (Michael Levitt) и Ари Уоршелу (Arieh Warshel) за разработку разномасштабных моделей сложных химических систем (“for the development of multiscale models for complex chemical systems”).

Почему премия по химии (даже Нобелевская!) заинтересовала журнал “Информатика”? Потому что Нобелевский комитет, формулируя, за что присуждена премия, пропустил одно слово, которое и так очевидно всем ученым. Это слово — “компьютерный”. Речь идет о разработке *компьютерных* моделей, то есть алгоритмов, программ, математических моделей (формул, дифференциальных уравнений и т.п.).

Так как речь идет о моделировании *естественно-научных* систем (премия присуждена по химии, однако работы, за которые она присуждена, лежат на стыке химии, биологии и физики), то в основе моделей лежит глубокое изучение законов природы — как теоретическое, так и экспериментальное.

Присуждение Нобелевской премии отмечает не только заслуги лауреатов, но и роль определенного направления науки. Нобелевская премия 2013 года по химии подчеркивает огромную роль, которую играет компьютерное моделирование в современных научных исследованиях (к слову — не только в естественно-научных).

Поздравим лауреатов! Кстати, по крайней мере двое из них (М.Карплюс и М.Левитт) имеют прочные научные связи с нашей страной, в частности, с лабораторией физики белка Института белка РАН (эта лаборатория была создана проф. О.Б. Птицыным, сейчас ею руководит чл.-корр. РАН А.В. Финкельштейн). На *рис.* 2 верхняя часть 227-й страницы из журнала "Journal of Molecular Biology" за 1976 г. со статьей М.Левитта и А.Уоршела [1]. На эту статью за прошедшие почти 40 лет (данные о ссылках — по Google Scholar) было 2414 ссылок (несколько десятков ссылок — достойный результат для научной статьи, очень хорошие статьи могут иметь 100–200 ссылок).



J. Mol. Biol. (1976) 103, 227-249

Theoretical Studies of Enzymic Reactions:

Dielectric, Electrostatic and Steric Stabilization of the Carbonium Ion in the Reaction of Lysozyme

A. WARSHEL AND M. LEVITT

Medical Research Council Laboratory of Molecular Biology
Hills Road, Cambridge CB2 2QH, England

and

Department of Chemical Physics
The Weixmann Institute of Science
Rehovot, Israel

(*Received 12 September 1975, and in revised form 10 February 1976*)

Рис. 2.
Обратите внимание: в названиях лабораторий, в которых работают авторы, упоминаются медицина, биология, химия и физика. К слову: найдите опечатку. Ответ — в конце этой заметки. Подсказка — в Приложении 2

Воспользуемся этой статьей, чтобы пояснить, в чем же состоит "разработка разномасштабных *компьютерных* моделей сложных химических систем", а точнее — молекул, прежде всего биологических макромолекул — белков и нуклеиновых кислот, и молекулярных комплексов. Именно этими "сложными химическими системами" занимаются новые нобелевские лауреаты.

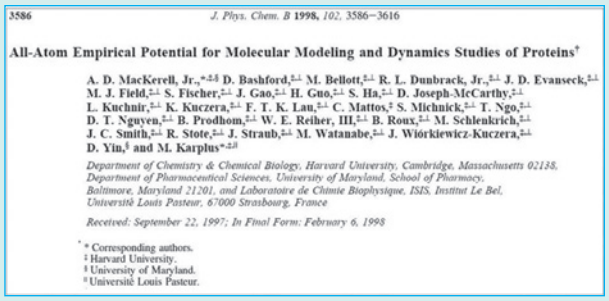
Название статьи переводится так: "Теоретическое изучение ферментативных реакций: диэлектрическая, электростатическая и пространственная стабилизация карбониевого иона при взаимодействии с лизоцимом". Начнем с конца. Лизоцим — это относительно небольшой белок, его роль — разрушение клеточных стенок бактерий. Лизоцима много в слюне и слезах — отсюда их антибактериальная роль. Подробнее — см. Приложение 1.

Лизоцим связывается с небольшими молекулами, входящими в стенки бактерий, и способствует их разрушению. Сама молекула лизоцима при этом не разрушается. Это и обозначает слово "ферментативная": фермент — это молекула, которая способствует определенной химической реакции (в данном случае — разрушению молекулы стенки бактериальной клетки), а сама при этом не меняется. М.Левитт и А.Уоршел изучали, как лизоцим стабилизирует карбониевый ион; это важно для ферментативной роли лизоцима. Карбониевым ионом в статье, говоря упрощенно, называется атом углерода, несущий положительный электрический заряд; этот атом входит в состав разрушаемой молекулы клеточной стенки. Оказывается, для стабилизации важны как "классические" эффекты, например, форма молекул и электростатика, так и квантово-механические, приводящие к расщеплению молекул клеточной стенки и перераспределению электрического заряда во взаимодействующих молекулах. Эти эффекты характеризуются совершенно различными масштабами энергий взаимодействий и расстояний, на которых эти взаимодействия происходят. Это и есть "разномасштабность", про которую говорится в решении Нобелевского комитета.

Кажется, с формулировкой решения ("за разработку разномасштабных моделей сложных химических систем") разобрались. В Приложении 2 — резюме (Abstract) из статьи Warshel, Levitt (1976). Прочитайте, будет понятно. В нем, мне кажется, чувствуется энергетика авторов — двух из трех лауреатов Нобелевской премии по химии 2013 г.

Третий лауреат, Мартин Карплюс, самый старший из трех, родился в 1930 г. в Австрии, а вырос в США: в 1938 г. его семья переехала в США, спасаясь от нацистов. К слову, сейчас все лауреаты работают в Америке, а родились и выросли они в разных частях света: М.Левитт родился в 1947 г. в Южной Африке, а А.Уоршел — в 1940 г. в Израиле. См. подробнее Приложение 3.

Основная работа М.Карплюса [2] (см. *рис.* 3, примерный перевод заглавия: "Межатомные эмпирические потенциалы для молекулярного моделирования и изучения динамики белков") вышла более чем на 20 лет позже работы А.Уоршела и М.Левитта — в 1998 году. У этой работы 29 соавторов и 6597 ссылок — почти в 3 раза больше, чем у работы А.Уоршела и М.Левитта. И то и другое имеет общую причину.



3586 *J. Phys. Chem. B* 1998, *102*, 3586−3616

All-Atom Empirical Potential for Molecular Modeling and Dynamics Studies of Proteins†

A. D. MacKerell, Jr.,*,‡,§ D. Bashford,‡,⊥ M. Bellott,‡,⊥ R. L. Dunbrack, Jr.,‡,⊥ J. D. Evanseck,‡,⊥ M. J. Field,‡,⊥ S. Fischer,‡,⊥ J. Gao,‡,⊥ H. Guo,‡,⊥ S. Ha,‡,⊥ D. Joseph-McCarthy,‡,⊥ L. Kuchnir,‡,⊥ K. Kuczera,‡,⊥ F. T. K. Lau,‡,⊥ C. Mattos,‡ S. Michnick,‡,⊥ T. Ngo,‡,⊥ D. T. Nguyen,‡,⊥ B. Prodhom,‡,⊥ W. E. Reiher, III,‡,⊥ B. Roux,‡,⊥ M. Schlenkrich,‡,⊥ J. C. Smith,‡,⊥ R. Stote,‡,⊥ J. Straub,‡,⊥ M. Watanabe,‡,⊥ J. Wiórkiewicz-Kuczera,‡,⊥ D. Yin,§ and M. Karplus*,‡,||

Department of Chemistry & Chemical Biology, Harvard University, Cambridge, Massachusetts 02138, Department of Pharmaceutical Sciences, University of Maryland, School of Pharmacy, Baltimore, Maryland 21201, and Laboratoire de Chimie Biophysique, ISIS, Institut Le Bel, Université Louis Pasteur, 67000 Strasbourg, France

Received: September 22, 1997; In Final Form: February 6, 1998

* Corresponding authors.
‡ Harvard University.
§ University of Maryland.
⊥ Université Louis Pasteur.

Рис. 3.
Работа выполнена в трех лабораториях, две из которых находятся в США, а одна — во Франции. Попробуйте разобраться, кто где работал ☺

Рис. 4а.
Схематическое изображение молекулы лизоцима. Показан ход полимерной цепи белка. Спирали (розовые) и стрелки (желтые) показывают по-особому организованные участки (так называемые “альфа-спиральные” и “бета-структурные участки”), благодаря которым лизоцим (как и многие другие белки) “упакован” в довольно плотную структуру

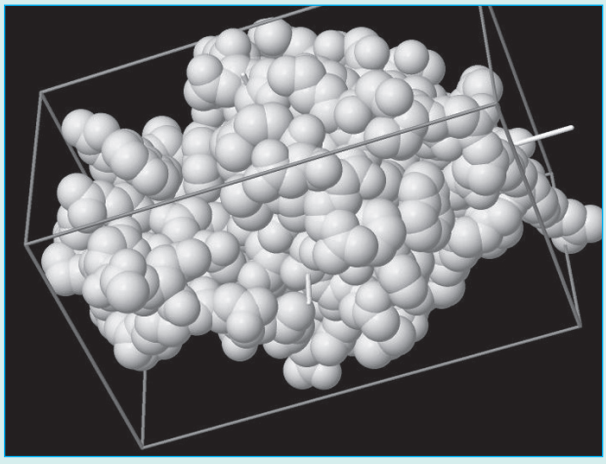


Рис. 4б.
Схематическое изображение поверхности молекулы (глобулы) лизоцима. В целом молекула имеет вид параллелепипеда со срезанными углами. Однако поверхность молекулы — не ровная. На ней есть “карманы”, благодаря которым молекула лизоцима и связывается с другими молекулами. Полусферы изображают атомы молекулы лизоцима, которые “видны” с поверхности. Многие атомы не видны — они внутри глобулы

Статья [1] посвящена теоретическому обоснованию подхода к расчету межатомных взаимодействий в макромолекулах и их комплексах и демонстрации возможностей этого подхода для одного белка (лизоцима). Подход Уоршела — Левитта, описанный в [1], позволил с достаточно хорошим приближением описать взаимодействие лизоцима с карбониевым ионом. При этом, несмотря на квантово-механическую природу процессов, авторы смогли обойтись без использования детальных квантово-механических расчетов. Но применим ли этот подход для других макромолекулярных комплексов? Существуют ли универсальные формулы и числовые параметры, применимые во всех случаях, или в каждом случае (классе случаев) потребуются свои варианты формул и параметров? Наконец, даже если мы знаем формулы и параметры, программная реализация подхода, применимая в достаточно широком классе случаев, — непростая задача. Статья [2], выполненная большим коллективом ученых (а иначе она и не могла быть выполнена), дает ответы на эти вопросы. Эта статья подвела итог работе, которая длилась более 15 лет и продолжается и сейчас, — работе по созданию комплекса программ CHARMM (***Chemistry at HARvard Macromolecular Mechanics***) и одноименного комплекса формул и параметров (подобные комплексы называют “силовые поля”, force field).

CHARMM позволяет работать с широким кругом макромолекул. Первая публикация по CHARMM [3] относится к 1983 году. Как и в случае работы [2], руководителем коллектива, в котором тогда было всего пять авторов, был М.Карплюс. Программа CHARMM, включая используемое силовое поле, постоянно совершенствуется. В работе [2] описана 22-я версия, а нынешняя версия — 31-я (см. *http://www.charmm.org/package/contents.html*). В настоящее время используются и другие силовые поля и соответствующие программные комплексы. Отметим AMBER и GROMACS; последний программный комплекс ориентирован на использование параллельных вычислений. По мере увеличения сложности изучаемых макромолекул увеличиваются требования и к точности приближения, которые дают силовые поля, и к скорости вычислений. Работа продолжается.

Замечание. Не следует думать, что Мартин Карплюс — это просто организатор, который “снимает сливки” с работ своих сотрудников. Он автор ряда основополагающих работ, написанных без соавторов или с небольшим количеством соавторов. Отметим его работу 1972 года [4], написанную вместе с А.Уоршелом и имеющую прямое отношение к тематике, за которую была дана Нобелевская премия этого года по химии. Современные масштабные научные проекты могут успешно возглавлять только люди, сочетающие и высокий научный уровень, и организаторские способности. В нашей стране такими были, например, математики Мстислав Всеволодович Келдыш и Иван Георгиевич Петровский, физики Игорь Васильевич Курчатов и Яков Борисович Зельдович и другие.

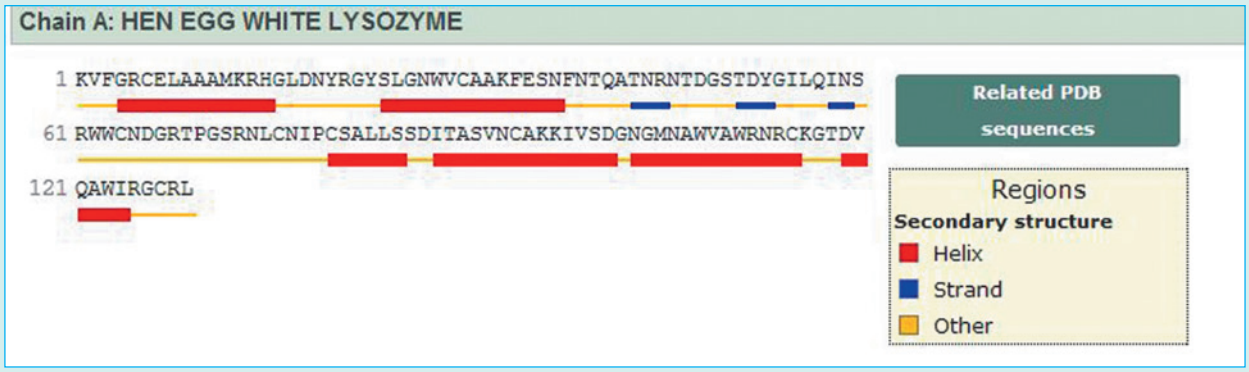


Рис. 5.
Вторичная структура лизоцима. Красным помечены альфа-спиральные, синим — бета-структурные участки. С сайта *http://www.ebi.ac.uk/pdbe-srv/view/entry/2lyz/secondary.html*

В заключение на нескольких примерах покажем, как компьютеры применяются в молекулярно-биологических исследованиях.

На *рис.* 4*а* и 4*б* изображена молекула лизоцима. Изображения получены с помощью программы Jmol и взяты с сайта международной базы данных по пространственным структурам биологических макромолекул PDB. Расшифровка подобных структур производится с помощью кристаллизации белков и рентгенографического анализа кристаллов. Обработка результатов такого эксперимента — сложная математическая задача, соответствующие расчеты выполнить без компьютеров нельзя. Получение изображений (программа Jmol и ее аналоги позволяют вращать изображения, раскрашивать их удобным способом, измерять расстояния между атомами и др.) требует использования средств компьютерной графики.

Так как экспериментальное определение пространственной структуры белка во много раз более сложная задача, чем определение последовательности аминокислот, то важно уметь предсказывать пространственную (“третичную”) структуру белка по его последовательности (“первичной структуре”). Методы расчета взаимодействий между атомами, предложенные в работах Карплюса, Уоршела, Левитта и других ученых, позволяют рассчитывать пространственную структуру. Однако в настоящее время точность предсказаний *de novo*, то есть не использующих ничего, кроме последовательности белка, невысока. Ситуация заметно улучшается, если использовать данные об экспериментально определенных структурах похожих белков.

Внимательный читатель может обратить внимание на слова “первичная” и “третичная” структуры и поинтересоваться — а что такое “вторичная структура”? Вторичная структура — это разметка на белковой цепи альфа-спиральных и бета-структурных участков (см. *рис.* 5, а также *рис.* 4*а*). На *рис.* 4*а* альфа-спиральные участки показаны розовым, а бета-структурные участки — желтыми стрелками.

Предсказывать вторичную структуру значительно проще, чем третичную, и качество предсказания выше — как для предсказаний *de novo*, так и для предсказаний с использованием сведений о сходных белках. При предсказании вторичной структуры белков наряду с другими подходами используется метод динамического программирования, предложенный Ричардом Беллманом. Этот метод применяется при решении многих алгоритмических задач, в том числе — задач вычислительной молекулярной биологии. Вариант этого метода (построение таблицы промежуточных результатов) используется при решении задач ЕГЭ по информатике последних лет (задача B13, см. *http://ege-go.ru/zadania/grb/b13/*).

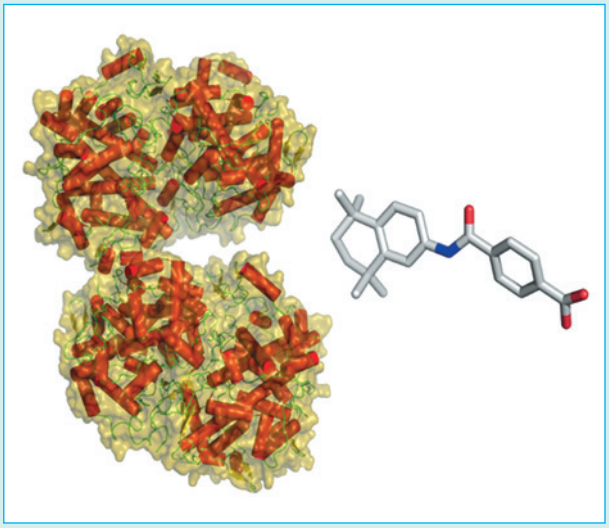


Рис. 6.
На рисунке схематически изображен процесс докинга (docking) — моделирование взаимодействия двух молекул, чаще всего белка (слева) и низкомолекулярного вещества — предполагаемого лиганда (справа). Цель докинга — найти оптимальную конформацию и расположение лиганда в месте связывания и оценить силу взаимодействия

Методы расчетов межмолекулярных взаимодействий, основа которых была заложена в работах новых нобелевских лауреатов и развита многими другими учеными, сейчас активно применяются при разработке новых лекарств (“разработка лекарств с помощью компьютеров” — computer aided drug de-

sign). Лекарство должно на молекулярном уровне воздействовать на один из основных механизмов, поддерживающих рост и развитие болезнетворных бактерий и вирусов. В качестве лекарства обычно используется небольшая молекула (“лиганд”), которая связывается с активным центром (“карманом”) целевого белка и, тем самым, блокирует его и не дает белку выполнять свою функцию. Используя современные модели межмолекулярных взаимодействий, можно попытаться рассчитать, какое вещество лучше всего будет связываться с известным белком, или как можно улучшить уже известный лиганд (см. *рис.* 6 на с. 7).

К сожалению, современное состояние науки не позволяет точно определить наилучший лиганд. На практике применяется “виртуальный скрининг” — предварительный отбор веществ-кандидатов, которые затем анализируются экспериментально. Этот отбор позволяет многократно сократить время разработки. Есть и другие применения “разномасштабных моделей сложных химических систем”, все они требуют применения компьютеров, а часто и суперкомпьютеров, в которых задача “распараллеливается” между тысячами процессоров. Есть много ученых, вклад которых в компьютерные и математические методы анализа биологических макромолекул стоит отметить. Можно попробовать рассказать и об отдельных методах. Но это — другая история.

Автор благодарит А.С. Карягину, Ю.Г. Колягина и А.В. Финкельштейна за помощь при подготовке статьи.

Ответ на загадку на *рис.* 2. Опечатка в слове “Weixmann”. Правильно: “WeiZmann” (в Приложении 2 написано правильно). Weizmann Institute, расположенный в г. Реховот, — один из ведущих научных центров Израиля. Ха́им Азриэль Ве́йцман, англ. Chaim Azriel Weizmann (1874–1952) — ученый-химик, первый президент государства Израиль.

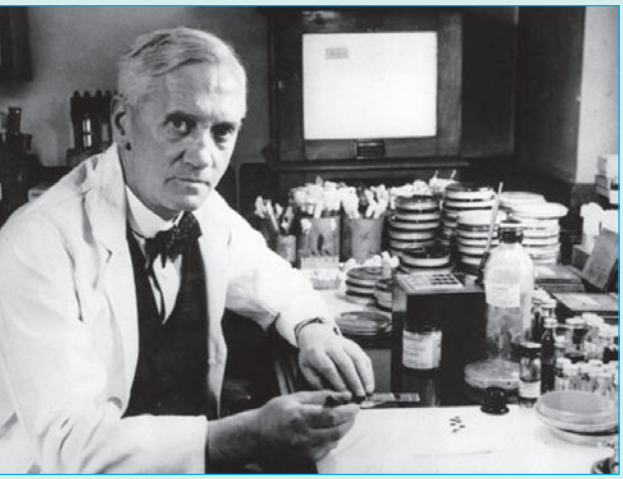
Литература

1. *Warshel A., Levitt M.* Theoretical studies of enzymic reactions: dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. J Mol Biol. 1976 May 15; 103(2): 227–49.

2. *MacKerell, Jr. AD, et al.* (1998). “All-atom empirical potential for molecular modeling and dynamics studies of proteins”. J Phys Chem B 102 (18): 3586–3616.

3. *Brooks B.R., Bruccoleri R.E., Olafson B.D., States D.J., Swaminathan S., Karplus M.* (1983). “CHARMM: A program for macromolecular energy, minimization, and dynamics calculations”. J Comp Chem 4 (2): 187–217.

4. *Warshel A., Karplus M.* (1972). “Calculation of ground and excited state potential surfaces of conjugated molecules. I. Formulation and parametrization”. Journal of the American Chemical Society 94 (16): 5612–5625.



Александр Флеминг. Фото из Википедии

Приложение 1

Лизоци́м (англ. *lysozyme*) — фермент класса гидролаз, разрушающий клеточные стенки бактерий путем разрушения пептидогликанов — небольших молекул, входящих в клеточные стенки бактерий. Главным образом, лизоцим получают из белка куриных яиц. Также аналогичные ферменты содержатся в организмах животных, в первую очередь в местах соприкосновения с окружающей средой — в слизистой оболочке желудочно-кишечного тракта, слезной жидкости, грудном молоке, слюне, слизи носоглотки и т.д. Лизоцим из белка куриных яиц представляет собой небольшой фермент (14,5 килодальтона), состоящий из 129 аминокислотных остатков.

История. В 1909 г. П.Л. Лащенко открыл в курином белке фермент, который избирательно повреждал клеточные стенки, содержащие пептидогликаны. Выделил в чистом виде, описал и дал название “лизоцим” в 1922 г. Александр Флеминг (тот самый, который позже открыл пенициллин).

Трехмерная структура лизоцима впервые была получена Дэвидом Чилтоном Филлипсом в 1965 году, когда он получил первую модель с помощью рентгеновской кристаллографии. Структура была публично представлена на лекции Королевского института. Лизоцим стал второй белковой структурой и первой ферментной структурой, которая была получена с помощью рентгеновской кристаллографии.

На работу Филлипса ссылаются в своей статье [1] М.Левитт и А.Уоршел.

[По материалам Википедии *http://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%B7%D0%BE%D1%86%D0%B8%D0%BC*]

Journal of Molecular Biology
Volume 103, Issue 2, 15 May 1976, Pages 227–249

Theoretical studies of enzymic reactions: Dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme

A. Warshel[a, b], M. Levitt[a, b]
[a] Medical Research Council Laboratory of Molecular Biology Hills Road, Cambridge CB2 2QH, England
[b] Department of Chemical Physics The Weizmann Institute of Science Rehovot, Israel

A general method for detailed study of enzymic reactions is presented. The method considers the complete enzyme-substrate complex together with the surrounding solvent and evaluates all the different quantum mechanical and classical energy factors that can affect the reaction pathway. These factors include the quantum mechanical energies associated with bond cleavage and charge redistribution of the substrate and the classical energies of steric and electrostatic interactions between the substrate and the enzyme. The electrostatic polarization of the enzyme atoms and the orientation of the dipoles of the surrounding water molecules is simulated by a microscopic dielectric model. The solvation energy resulting from this polarization is considerable and must be included in any realistic calculation of chemical reactions involving anything more than an isolated molecule *in vacua*. Without it, acidic groups can never become ionized and the charge distribution on the substrate will not be reasonable. The same dielectric model can also be used to study the reaction of the substrate in solution. In this way the reaction in solution can be compared with the enzymic reaction.

In this paper we study the stability of the carbonium ion intermediate formed in the cleavage of a glycosidic bond by lysozyme. It is found that electrostatic stabilization is an important factor in increasing the rate of the reaction step that leads to the formation of the carbonium ion intermediate. Steric factors, such as the strain of the substrate on binding to lysozyme, do not seem to contribute significantly.

Перевод [*пояснения переводчика выделены курсивом*]:

Предложен общий подход к детальному изучению ферментативных реакций. В рамках этого подхода комплекс энзим-субстрат [*комплекс, состоящий из молекулы энзима, т.е. белка-фермента, и молекулы субстрата, т.е. вещества, с которым взаимодействует энзим*] рассматривается вместе с окружающим его растворителем; при этом оцениваются все разнообразные квантово-механические и классические факторы, которые могут влиять на пути протекания [*химической*] реакции. Эти факторы включают квантово-механические энергии, связанные с разрушением связей и перераспределением заряда в субстрате, и классические энергии стерических [*то есть связанных с геометрией молекул*] и электростатических взаимодействий между субстратом и энзимом. Электростатическая поляризация атомов энзима и ориентация диполей окружающих молекул воды представляется с помощью микроскопической диэлектрической модели [*если нужно — перечитайте школьные учебники физики и химии, спросите учителей по этим предметам или посмотрите в Интернете и книгах* ☺]. Энергия растворения, порождаемая этой поляризацией, достаточно велика и обязательно должна учитываться в расчетах протекания любых химических реакций, более сложных, чем реакция двух изолированых молекул *in vacua* [то есть *в вакууме*]. Без учета энергии растворения боковые группы аминокислотных остатков никогда не будут ионизированными, а распределение заряда на субстрате будет далеко от реального. Эта же диэлектрическая модель может использоваться при изучении реакции субстрата в растворе. В этом смысле реакцию в растворе можно сравнить с ферментативной реакцией.

В статье мы изучаем стабильность карбониевого иона — интермедиата, который образуется при разрушении гликозидной связи лизоцимом [интермедиат (лат. *intermedius* — средний) — вещество с коротким временем жизни, образующееся в ходе химической реакции и затем распадающееся]. Стерические факторы, такие, как напряжение в структуре субстрата при связывании с лизоцимом, как выяснилось, не играют существенной роли.

НАУКИ И МИР

Все лауреаты Нобелевской премии 2013 года по химии сейчас работают в Америке. Но родились и выросли они в разных частях света: М.Левитт родился в 1947 г. в Южной Африке, А.Уоршел — в 1940 г. в Израиле. Мартин Карплюс, самый старший из трех, родился в 1930 г. в Австрии, но вырос в США: в 1938 году его семья переехала в США, спасаясь от преследования евреев нацистами.

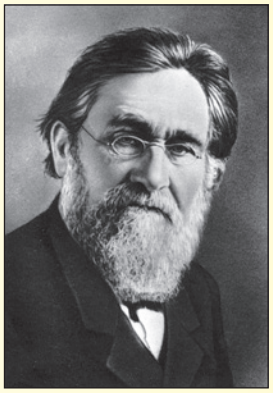
Чтобы ни один континент не был забыт, назовем великого предшественника нынешних лауреатов, создателя планетарной модели атома, лауреата Нобелевской (1908 года) премии по химии (!) Эрнеста Резерфорда — “за проведенные им исследования в области распада элементов в химии радиоактивных веществ”. Э.Резерфорд родился в Новой Зеландии и большую часть жизни работал в Англии. Кстати, в том же году, что и Э.Резерфорд, Нобелевскую премию по медицине получил наш великий соотечественник Илья Ильич Мечников (ешьте простоквашу — будете здоровы!).

Как видим, Нобелевские премии по химии и раньше присуждались за исследования в пограничных с химией областях. В первой половине XX века это была граница с физикой; прежде всего — физикой радиоактивных веществ. Кроме Э.Резерфорда, Нобелевские премии по химии присуждались Марии Склодовской-Кюри (1911), Фредерику Жолио и Ирен Жолио-Кюри (1935), Энрико Ферми (1938). Во второй половине XX века повышается роль биологии, и это находит свое отражение в списке Нобелевских лауреатов по химии. По-видимому, первым Нобелевскую премию по химии за работы, связанные с биологией, получил в 1938 году выросший в Австрии и большую часть жизни проработавший в Германии Рихард Кун — “в знак признания проделанной им работы по каротиноидам и витаминам”. Кун одним из первых начал исследовать биологические ферменты (энзимы), которым посвящена статья М.Левитта и А.Уоршела*.

Задержимся на цифре 38**. В 1938 году гитлеровская Германия присоединила Австрию (“аншлюс”), семья М.Карплюса была вынуждена бежать, спасаясь от нацистов. А лауреат Нобелевской премии по химии 1938 года (присуждение состоялось в 1939 году), профессор Гейдельбергского университета Рихард Кун был вынужден отказаться от получения Нобелевской премии, поскольку Гитлер запретил немецким ученым получать эту премию. Позже, так же, как Р.Кун, от премии были вынуждены отказаться химик Адольф Фридрих Иоганн Бутенандт и выдающийся бактериолог Герхард Домагк (после разгрома гитлеровской Германии все они получили Нобелевские медали).

Запрет Гитлера был связан с тем, что лауреатом Нобелевской премии мира 1935 года (присуждена в 1936 году) стал немецкий журналист и общественный деятель Карл фон Осецкий “за борьбу с милитаризмом в Германии”. Фон Осецкий был арестован вскоре после прихода Гитлера к власти (в 1933 г.) и помещен в концлагерь. В момент присуждения Нобелевской премии мира фон Осецкий находился в тюремной больнице. Карл фон Осецкий умер от туберкулеза 4 мая 1938 года.

Ученые и общество, гражданская позиция ученых (вспомним Галилея, Нильса Бора, Альберта Эйнштейна, Андрея Дмитриевича Сахарова), жизнь ученых в переломные моменты истории, например в нацистской Германии, — важная тема. Но это отдельная тема.



Илья Ильич Мечников (1845–1916)

Карл фон Осецкий (1889–1938)

Рихард Кун (1900–1967), фото ETH Zürich

Эрнест Резерфорд (1871–1937)

---

* Менее известно, что, работая по армейскому заказу, Р.Кун в 1944 году синтезировал боевое отравляющее вещество нервно-паралитического действия зоман. К счастью, гитлеровская армия его не применила в боевых условиях. После войны зоман состоял на вооружении армий всех ведущих держав.

** Это скрытая цитата из В.С. Высоцкого: “Задержимся на цифре 37...”.

Интерактивный
короткофокусный проектор

MimioProjector

Легкий и доступный способ
внедрения интерактивного обучения
в классах, не оборудованных проектором

Кто сказал, что нельзя получить все сразу? Установите MimioProjector, подключите к компьютеру и оживите ваши уроки, используя все возможности интерактивного обучения на маркерной доске.

В вашем классе уже есть маркерная доска, но нет проектора. А еще вы мечтаете об интерактивной доске, но на все это не хватает средств? Наш новый интерактивный **MimioProjector** даст вам возможность внедрить интерактивное обучение, избежав чрезмерных расходов, — ведь он совмещает в себе функции отличного короткофокусного проектора и полноценной интерактивной доски!

Благодаря функции использования двух интерактивных ручек сразу двое учащихся могут одновременно выполнять манипуляции с объектами на доске, используя все преимущества группового обучения. А ультракороткофокусная модель проектора позволяет снизить количество теней на экране. Проектор легко подключается к компьютеру и позволяет начать использовать имеющуюся у вас маркерную доску или даже светлую стену вашего класса в качестве интерактивной доски. Поставляемое в комплекте с проектором программное обеспечение **MimioStudio** позволяет вам создавать и проводить увлекательные уроки и управляет всем оборудованием семейства **MimioClassroom**.

Проектор: широкоформатный (16:10) с разрешением 1280 x 800 точек (WXGA). Коэффициент контрастности до 3000:1. Размер экрана от 70 до 100 дюймов.
Интерактивные возможности: рабочая область от 75 до 115 дюймов по диагонали. Одна или две интерактивные ручки в зависимости от комплекта поставки. Проектор можно заказать в комплектации с одной или двумя интерактивными ручками либо как традиционный проектор без интерактивных функций.

Продажа оборудования, консультации и обучение:

http://www.mimioclass.ru

8 (800) 5555-33-0

Звонок по России бесплатный

ООО «Рене» — генеральный дистрибьютор Mimio в России

mimio
a better way to learn

“Колокола” случайных чисел

А.И. Азевич,
Москва

Мир разнообразен. На каком бы уровне его не рассматривать. Взять хотя бы человека. У каждого свой вес, рост, возраст, цвет глаз и т.д. Очень трудно найти похожих людей. Производя всевозможные измерения, ученые научились приводить в порядок величины, относящиеся к самым разным областям. Они открыли законы, которые помогают наглядно представить разные и совсем не связанные друг с другом данные.

Чтобы понять некоторые математические законы, проведем эксперимент. Представим, что есть группа исследователей. Им дано задание — узнать рост довольно большого количества мужчин. Среди испытуемых найдутся “карлики”, рост которых меньше 140 см, и “гиганты”, возвышающиеся над землей более чем на 200 см. Но это скорее исключение из правил. Большинство имеет рост, находящийся в пределах некоторого интервала. Но об этом чуть позже. А пока составим табл. 1, в которую внесем данные двух видов: рост мужчин, сгруппированный по интервалам, и частоту появления того или иного роста.

Пусть рост мужчин варьируется от 140 см (хотя эта величина может быть и меньше) до 208 см (есть, правда, и более высокие люди). У нас вырисовывается некая шкала от 140 до 208. Разобьем ее на 68 мелких шагов-интервалов по 1 см. Первый из них от 140 до 141, второй — от 141 до 142, и т.д. Последним будет интервал от 207 до 208. Получили первый набор величин — длин интервалов, в которые попадет случайная величина — рост мужчины.

Предположим, что количество мужчин с ростом 140 см равно 10. Тех, у кого рост 141, насчитали 12. Дальше пойдет 142 см и т.д. В ходе эксперимента можно заметить, что частота появления

Таблица 1

Интервалы роста	140–141	141–142	…	172–173	173–174	174–175	…	206–207	207–208
Частота появления	10	12	…	1010	1300	1004	…	13	11

мужчин очень маленького роста (около 140 см) и очень большого роста (свыше 200 см) весьма мала. Зато мужчин, рост которых приближается к 174 см, больше всего. Поэтому и частота их появления в этом случайном ряде чисел наибольшая.

Следует сказать, что величины, приведенные в таблице, весьма условны и недостаточно полно отражают реальную картину. Таких измерений должно быть очень много. Интересно понять, как будут связаны две группы данных в результате такого эксперимента.

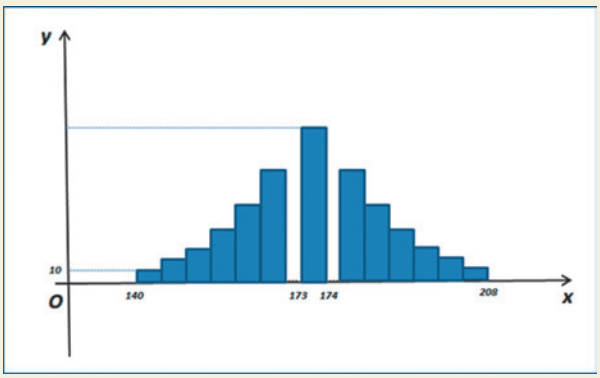


Рис. 1

Чтобы узнать это, введем систему координат *XOУ*. По оси *OX* будем откладывать интервалы попадания случайных величин роста, по оси *OУ* — соответствующие им частоты. В результате получим конструкцию, состоящую из двухсторонних ступенек (*рис.* 1). Мы умышленно не отметили на оси *OУ* частоту, соответствующую интервалу (173;174), так как ряд частот подчиняется довольно сложному закону. Для нас важно понять, как выстраиваются столбики, каждый из которых связывает две величины: рост и частоту его появления для данного интервала.

Рассматривая рисунок, глаз мысленно пытается провести некоторую линию, проходящую по вершинам столбиков. Действительно, если бесконечно уменьшать ширину столбиков, то получится линия, похожая на очертание колокола. Сейчас ширина столбика равна 1 см. А можно взять 0,1 см, потом

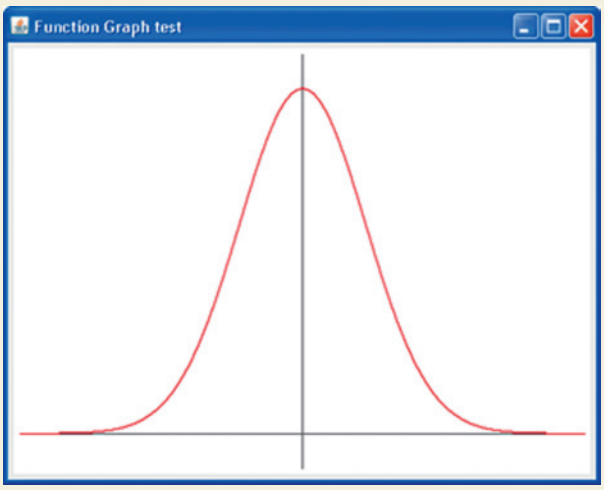


Рис. 2

0,01 см и т.д. Столбиков станет все больше и больше. Чем точнее и больше будет измерений, тем четче будет выглядеть кривая-колокол. Она носит название кривой Гаусса — в честь немецкого математика, первым открывшего ее. Вот как она выглядит на компьютере (*рис.* 2).

Кривая показывает плотность распределения вероятности появления случайной величины. Ей соответствует довольно сложная функция, имеющая вид:

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-\mu)^2}{2\sigma^2}},$$

где μ — математическое ожидание[1],
σ — стандартное отклонение,
σ^2 — дисперсия[2] распределения.
Эта функция выражает так называемое “нормальное распределение”.

Если результат наблюдения является суммой многих случайных слабо взаимозависимых величин, каждая из которых вносит малый вклад относительно общей суммы, то при увеличении числа слагаемых распределение стремится к нормальному. Так звучит один из важнейших законов теории вероятностей. Это обусловлено тем, что он проявляется во всех случаях, когда случайная величина является результатом действия большого числа различных факторов.

Чтобы построить одну из кривых Гаусса, воспользуемся программой MS Excel.

Рассмотрим конкретную задачу. Построить график нормальной функции распределения $f(x)$ при x, меняющихся от 19,8 до 28,8 с шагом 0,5, математическим ожиданием 24,3 и стандартным отклонением 1,5.

Откроем программу MS Excel. В ячейку A1 внесем переменную x, в ячейку B1 — $f(x)$. Выделим ячейки A2:A3, потянем за крестик, нажав на левую кнопку мыши. Протащим его вдоль столбца A до ячейки A21. Увидим, что в столбце выстроились значения величин с шагом 0,5 (см. табл. 2 на с. 8).

Выделяем ячейку B2. Нажимаем формулы. Выбираем последовательно *Другие функции, Статистические*. Затем в открывшемся списке НОРМ.РАСП. Нажав на нее, переходим к окну “Аргументы функции”. Заполняем пустые ячейки так, как показано на *рис.* 3.

Поясним, что означает каждый аргумент, входящий в функцию нормального распределения. *Среднее* μ — среднее арифметическое распределения; чем дальше X от среднего, тем ниже вероятность наступления такого события.

---

[1] *Математическим ожиданием* дискретной случайной величины называется сумма произведений всех возможных значений случайной величины на их вероятности.

[2] *Дисперсией (рассеиванием)* дискретной случайной величины называется математическое ожидание квадрата отклонения случайной величины от ее математического ожидания.

*Стандартное_откл*σ — стандартное отклонение распределения; мера кучности; чем меньше σ, тем выше вероятность у тех *X*, которые расположены ближе к *среднему*. *Интегральная* — логическое значение, определяющее форму функции. Если “интегральная” имеет значение ИСТИНА, функция НОРМРАСП возвращает интегральную функцию распределения, то есть суммарную вероятность всех событий для аргументов от –∞ до *X*; если “интегральная” имеет значение ЛОЖЬ, возвращается вероятность реализации события *X*, точнее говоря, вероятность событий, находящихся в некотором диапазоне вокруг *X*.

После того как внесены все аргументы, функция нормального распределения примет вид:

=НОРМ.РАСП(A21;24,3;1,5;0).

В ячейке B2 видим вероятность появления числа 19,8. Копируем последнюю функцию по столбцу B в ячейки B3:B21. Для этого выделим ячейку B2 и протащим за крестик, расположенный в правом нижнем углу. При этом надо обязательно удерживать левую кнопку мыши. В ячейках B2:B21 появятся значения функции $f(x)$.

Таблица 2

	A	B
1	*x*	*f(x)*
2	19,8	0,002955
3	20,3	0,007597
4	20,8	0,017481
5	21,3	0,035994
6	21,8	0,066318
7	22,3	0,10934
8	22,8	0,161314
9	23,3	0,212965
10	23,8	0,251589
11	24,3	0,265962
12	24,8	0,251589
13	25,3	0,212965
14	25,8	0,161314
15	26,3	0,10934
16	26,8	0,066318
17	27,3	0,035994
18	27,8	0,017481
19	28,3	0,007597
20	28,8	0,002955
21	29,3	0,001028

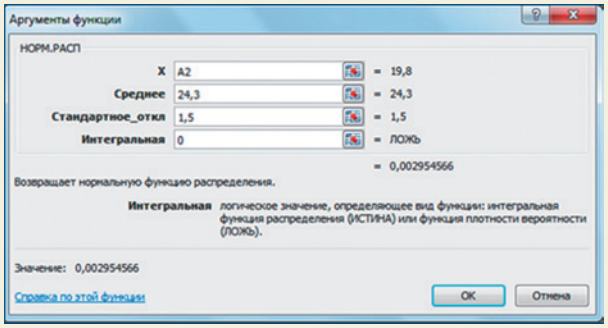


Рис. 3

Осталось построить кривую нормального распределения (плотность вероятности), воспользовавшись *Мастером диаграмм* программы MS Excel. Выделяем ячейки B1:B21. Затем последовательно кликаем *Вставка*, *График* (рассматривается версия MS Excel 2010). Выбираем тип *График с маркером*. Нажимаем на него и получаем кривую Гаусса (*рис.* 4).

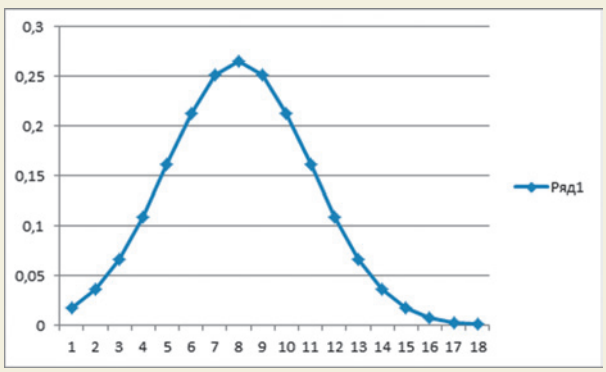


Рис. 4

Положение кривой Гаусса в декартовой системе координат зависит от математического ожидания и дисперсии распределения. Меняя значения этих величин, получим различные кривые — “колокола” (*рис.* 5). Одни из них крутые, другие пологие. И вместе с тем каждая кривая — это визитная карточка закона нормального распределения случайных величин.

Подобных распределений в природе немало. Можно сказать, что это *норма* для природы, которой “управляет” Его Величество Случай. Ясно, что у каждого нормального распределения будет свой “колокол” (своя “картинка”), свои параметры распределения — математическое ожидание и дисперсия.

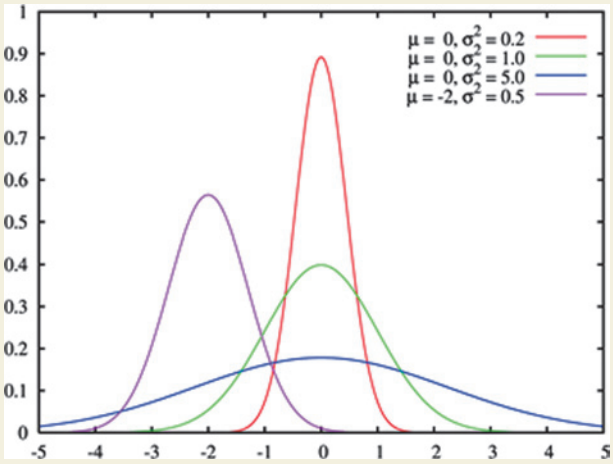


Рис. 5

Некоторые процессы кучнее группируются возле среднего значения, другие более разбросаны. Например, рост собак и рост домашних кошек имеют разный разброс значений, их кривые нормального распределения будут выглядеть по-разному. Так, кривая для роста кошек будет более узкой и высокой, а для роста собак — ниже и шире.

Рассмотрим еще один показательный пример, приводящий к нормальному распределению случайных величин. Он связан с доской Гальтона. Ее изобрел английский ученый Фрэнсис Гальтон в 1873 году.

Доска Гальтона представляет собой ящик, на задней стенке которого укреплены: вверху две наклонные планки, образующие воронку; в середине несколько рядов вбитых в стенку и расположенных в шахматном порядке гвоздей; внизу система одинаковых вертикальных ячеек. Передняя стенка ящика стеклянная (*рис.* 6).

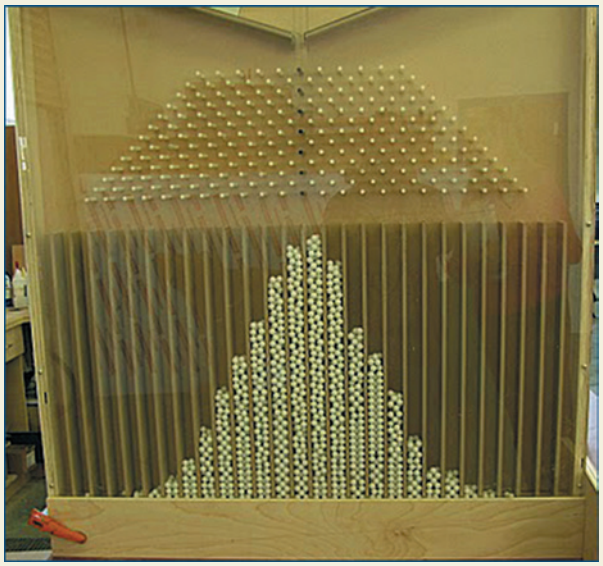


Рис. 6

Бросим в воронку одну горошину и проследим за ее движением. Горошина, претерпев ряд столкновений с гвоздями, попадет в какую-то ячейку. Предсказать заранее, с какими гвоздями столкнется горошина и в какую ячейку попадет, невозможно. Это случайный процесс. Так будет происходить и с любой другой горошиной. Будем сыпать горох в воронку непрерывным потоком. Заранее можно сказать, что в центральные ячейки попадет большее число горошин, чем в воронки, расположенные по бокам. Неизвестно, как будет вести себя отдельная частица-горошина, но зато экспериментально установлено, какое положение займет множество всех высыпанных в ящик горошин. На *рис.* 6 мы вновь видим нормальное распределение случайных величин, которое выражает кривая Гаусса.

На сайте университета Колорадо можно увидеть динамическую модель доски Гальтона. Вот его адрес: *http://phet.colorado.edu/sims/plinko-probability/plinko-probability_en.html*.

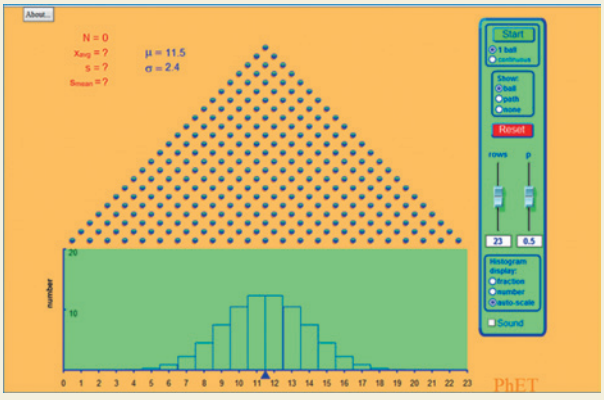


Рис. 7

Что мы видим на экране? В центре — треугольник с гвоздиками (нам видны только их шляпки) (*рис.* 7). Под ним — система координат, в которой построено множество столбиков. Это гистограмма. Правее — пульт с бегунками. Он называется Histogramdisplay. По-русски — выставка гистограмм.

Количество гвоздиков в этой модели можно изменять. Для этого надо подвигать бегунок **rows** (ряды). Если уменьшить количество гвоздиков, изменится и гистограмма. Столбики станут шире, а гистограмма — ниже. Если вращать ползунок **p**, гистограмма будет смещаться вдоль оси *OX*. Вместе с движениями бегунков будут меняться и главные параметры, входящие в формулу нормального распределения, — математическое ожидание и дисперсия. Их показатели расположены слева от треугольника гвоздиков. Попробуйте поэкспериментировать с виртуальной доской Гальтона. Определите, как меняются показатели формулы нормального распределения, форма и положение гистограммы.

Как мы уже выяснили, нормальное распределение случайных величин — один из фундаментальных законов теории вероятностей. Его действие можно встретить повсюду: в показателях роста человека и животных, в погрешности измерений, во многих физических явлениях, в давлении крови и в экзаменационных оценках. Да что говорить, даже в метро можно увидеть кривую Гаусса. Число людей, ожидающих поезда и желающих попасть в средние вагоны, значительно больше, чем стоящих в начале и конце платформы. А если посмотреть на это сверху, то вновь увидим гистограмму — “колокол”. Закон нормального распределения — повсюду! И как тут поспоришь с известной фразой “Миром правят числа”?!

Источники сети Интернет

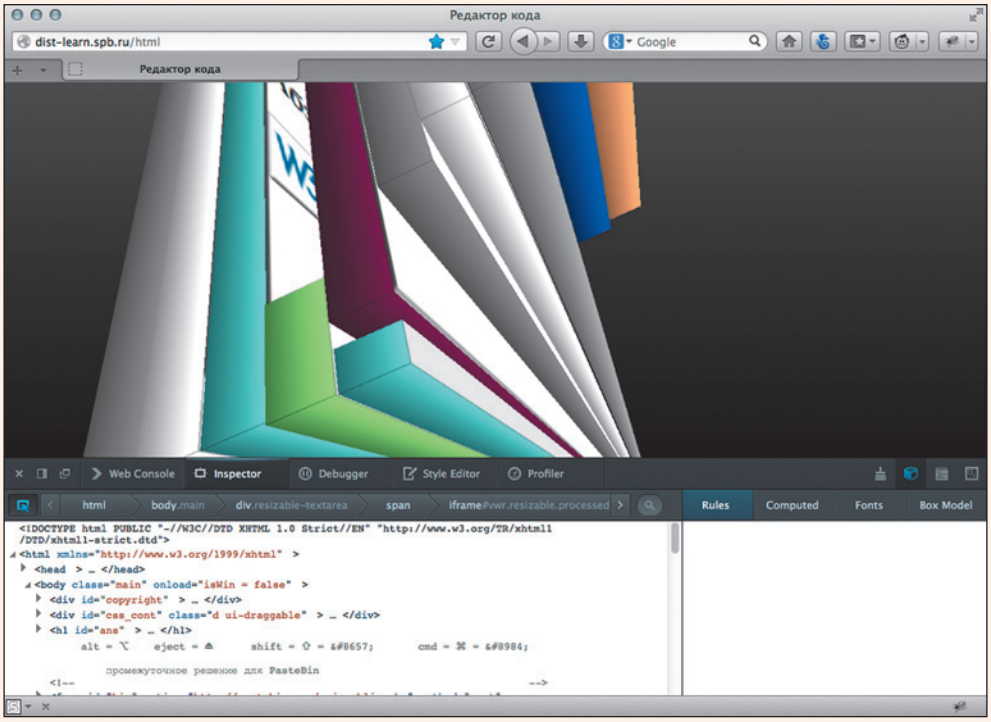
1. *http://web-in-math.blogspot.ru/2012/02/blog-post_27.html*.
2. *http://phet.colorado.edu/sims/plinko-probability/plinko-probability_en.html*.
3. *http://www.youtube.com/watch?v=M9tw6xC9ZPk*.
4. *http://hsetube.skillopedia.ru/material.php?id=4186*.
5. *http://baguzin.ru/wp/?p=1170*.
6. *http://maxpark.com/community/2516/content/831912*.
7. *http://zhurnal.lib.ru/i/isaew_aleksandr_wasilxewich/number55-2.shtml*.
8. *http://fenix.vn.ua/raznoe/normalnoe_raspredelenie/*.
9. *http://metallovedeniye.ru/analiz-dannyx-v-excel/postroenie-gistogramm-raspredeleniya-v-excel.html*.
10. *http://www.mathelp.spb.ru/book2/tv11.htm*.
11. *http://data-machine.ru/statistic/normalnoe-raspredelenie-eto-prosto.html*.
12. *http://www.obman.su/zakon-normalnogo-raspredeleniya-v-sociologii.htm*.

Трехмерный веб

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

Если в браузере Firefox открыть инструменты разработчика (WebDeveloper) и нажать на кнопку с изображением кубика, то откроется трехмерное представление структуры страницы.

Как это достигается? В предыдущем выпуске “Информатики” описывались основные подходы к созданию простых “трехмерных” изображений с



помощью доступного и легкоосваиваемого программного обеспечения (Google SketchUp). На самом деле все эти изображения псевдотрехмерны, ибо призваны создавать у зрителя иллюзию объемных объектов на плоском экране. Рендеринг трехмерных сцен, то есть превращение некоторого трехмерного геометрического описания в двумерную иллюзию, представляет собой сложный ресурсоемкий вычислительный процесс, и поэтому качественная 3D-графика — удел десктоп-программирования.

Однако все продолжающаяся тенденция перемещения документов и программ в Веб, в облака означает, что и трехмерные иллюзии должны как-то обрести воплощение на веб-страницах.

Такие попытки делались очень давно. Оставляя в стороне безнадежно оставшиеся в прошлом Java-апплеты и реализации Direct 3D в виде ActiveX-элемента в браузере Microsoft Internet Explorer, упомянем главный способ привнесения медиа, анимации и 3D в Интернет.

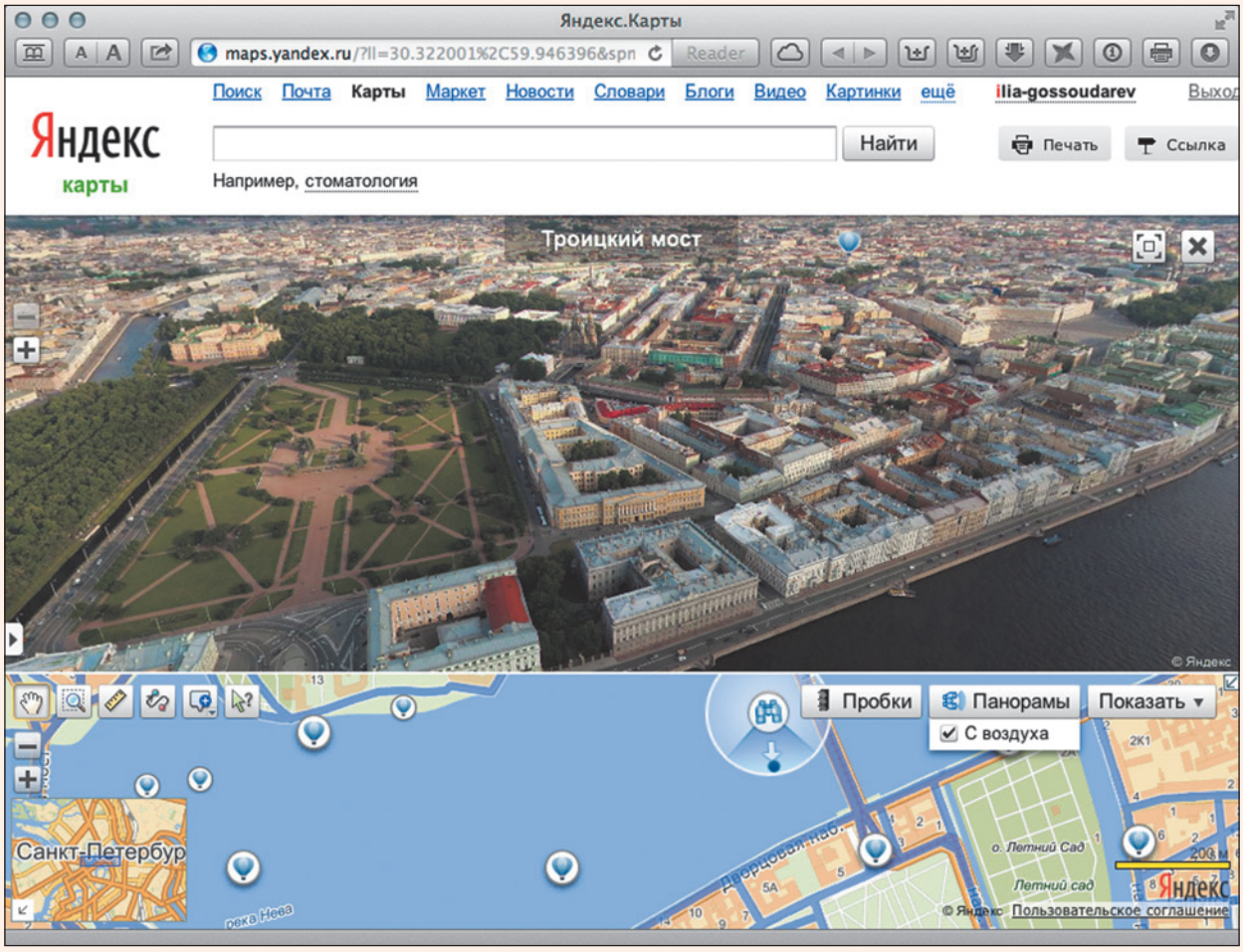
Flash

Для языка ActionScript (с помощью которого осуществляется программирование поведения объектов в Flash-роликах) разработаны движки, позволяющие создавать трехмерные сцены и программировать анимацию. Фактически они реализуют некоторый интерфейс прикладного программирования (API = ***A**pplication **P**rogramming **I**nterface*), то есть предоставляют определенный набор данных и функций, которые вызывает в своей программе разработчик. Одним из таких открытых движков является Alternativa3D *http://alternativaplatform.com/en/*. В разделе Showcase на сайте представлены примеры; в основном это игры, но встречаются и другие варианты, такие, как трехмерные карты: *http://gkb40.ur.ru/web/map40a3d.swf*.

Псевдотрехмерность применяется как элемент привнесения реалистичности в карты. Карты Яндекса (*maps.yandex.ru*) содержат режим “Панорамы” (в том числе “с воздуха”), который реализован во Flash путем соединения определенного количества фотографий. Это снижает нагрузку на процессор и память, так как производить рендеринг не нужно. Но, естественно, посмотреть объект можно только под теми углами, которые заранее предусмотрены разработчиками.

Отметим, что технология создания псевдотрехмерных панорамных изображений достаточно давно используется в специальном плагине Apple Quicktime, который может быть подключен к браузеру. Он применяется для создания виртуальных экскурсий, например, *http://www.chem.ox.ac.uk/oxfordtour/brasenosecollege/map.html*.

Автор этой статьи руководил дипломной работой студента Ивана Конакова, создавшего виртуальную

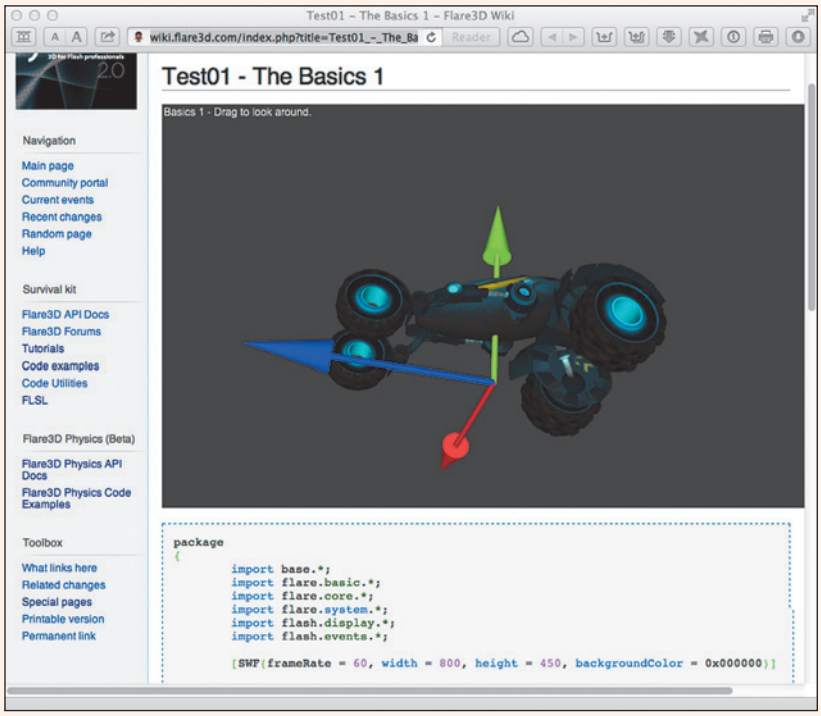


панораму кампуса Герценовского университета: *http://dist-learn.spb.ru/Konakov_Ivan/#*.

Соединение движения и 360-градусной панорамы дает 3D-видео, при проигрывании которого зритель управляет точкой обзора. Впечатляющая коллекция таких видеороликов может быть найдена по адресу *http://immersivemedia.com/demos/*.

Однако, повторимся, склеивание снимков, хотя и создает иллюзию обзора объемного вида, на самом деле является изначально плоским, так как в "исходниках" не содержится информация о координатах объектов в пространстве.

Контрпримером является еще одна библиотека, или движок, под названием Flare3D (*http://www.flare3d.com/downloads/*). Она ориентирована в том числе на импорт моделей из 3DMax и содержит команды, позволяющие создавать сцены, объекты из различных материалов, изменять положение "камеры", создавая иллюзию перемещения вокруг объектов. Результатом компиляции является файл *.swf, для проигрывания которого необходим Adobe Flash Player: *http://wiki.flare3d.com/index.php?title=Test01_-_The_Basics_1*.



Но Flash неуклонно уступает свои позиции, во многом из-за того, что производители популярных мобильных устройств (включая iPhone, iPad) не лицензируют Flash для своих операционных систем. При этом доля просмотра веб-страниц с мобильных устройств растет. В связи с этим развиваются альтернативные технологии внедрения медиа, анимации и 3D в веб-документы.

CSS

Одним из самых примитивных вариантов является использование каскадных стилей (CSS) для создания псевдотрехмерных эффектов (теней, бликов, преобразований). Как известно читателю, каскадные стили — это технология описания форматирования веб-страниц, позволяющая сопоставить тегам на языке разметки правила. Например, чтобы задать фоновое изображение для веб-страницы, необходимо определить соответствующее правило для тега `body`:

```
body {background-image:url(//dist-learn.spb.ru/cube.gif)}
```

Правила лучше всего определять в отдельном файле, который подключается к веб-странице с помощью тега `link`.

Напомним, что приводимые здесь примеры можно посмотреть в действии с помощью какого-либо онлайнового (облачного) инструмента, например, редактора *http://dist-learn.spb.ru/html?5*.

В предыдущей статье, посвященной векторной и динамической графике на веб-страницах [1], мы использовали облачный редактор для генерирования SVG-кода по вручную созданному изображению и помещали полученный код в указанный выше редактор. Также использовался сайт *caniuse.com* для получения информации о поддержке браузерами рассматриваемых технологий. Соответственно, к этому сайту полезно обратиться и теперь. При этом следует учитывать, что все эти технологии находятся в процессе становления, и некоторые браузеры более гибко реагируют на происходящие изменения, чем прочие. Целесообразно порекомендовать использовать браузер Chrome для всех экспериментов с веб-страницами, поскольку он обновляется быстрее всех и, как правило, содержит поддержку самых свежих нововведений. Некоторые новые свойства каскадных стилей по-разному реализованы в разных браузерах. Рассмотрим свойство border-radius, позволяющее скруглять углы прямоугольных элементов, в том числе превращать их в круги. На определенном этапе существования этого свойства его нужно было написать несколько раз в различных вариациях, так чтобы браузер пользователя отобрал нужную:

-moz-border-radius для Firefox;
-o-border-radius для Opera;
-ms-border-radius для Internet Explorer;
-webkit-border-radius для Chrome и Safari.

Однако на момент написания этой статьи для современных версий всех браузеров годится написание **border-radius** без префиксов.

То же касается свойства **box-shadow**, которое управляет отображением тени. А вот новое свой-

ство, задающее градиентную заливку, пока еще различается. Так оно будет записываться для браузеров на движке webkit (Chrome, Safari, Opera):

-webkit-**linear-gradient**(top, rgb(0,0,250) 0%, rgb(0,0,40) 100%).

Так — для Internet Explorer:

-ms-**linear-gradient**(top, rgb(0,0,250) 0%, rgb(0,0,40) 100%).

И так — для Mozilla Firefox:

-moz-**linear-gradient**(top, rgb(0,0,250) 0%, rgb(0,0,40) 100%).

Для экономии места в статье указан только -webkit-.

Итак, создадим простейшую иллюзию — шар, отбрасывающий тень. Для этого будет достаточно открыть HTML-редактор и добавить в код страницы теги `<div></div>`, после чего в раздел каскадных стилей вставить следующее:

```
div {height: 100px; width: 100px;
            border-radius: 50px;
  box-shadow: 5px 5px 10px rgba(60, 60,
                                60, 0.95);
 background: -webkit-linear-gradient(top,
            rgb(0,0,250) 0%,
            rgb(0,0,40) 100%);
}
```

Данное правило устанавливает круглую форму элемента, полупрозрачную тень и градиент, создающий иллюзию подсветки.

Если к указанным свойствам присовокупить простую анимацию (которая задается в CSS3 либо с помощью переходов (transition), либо с помощью ключевых кадров (keyframes), то можно достичь и более достоверной иллюзии (подробную инструкцию см. в [2]).

Как и в случае с SVG, можно создавать описания с помощью тегов, но можно и генерировать код HTML/CSS по вручную созданному изображению. Пример такого облачного редактора — Tridiv (*http://tridiv.com/app/*) на с. 20.

Для создания других эффектов широко используются облачные инструменты — онлайн-генераторы:

- *http://css3gen.com/box-shadow/* (генерация теней);
- *http://www.css3maker.com/css3*-animation.html# (генерация анимаций).

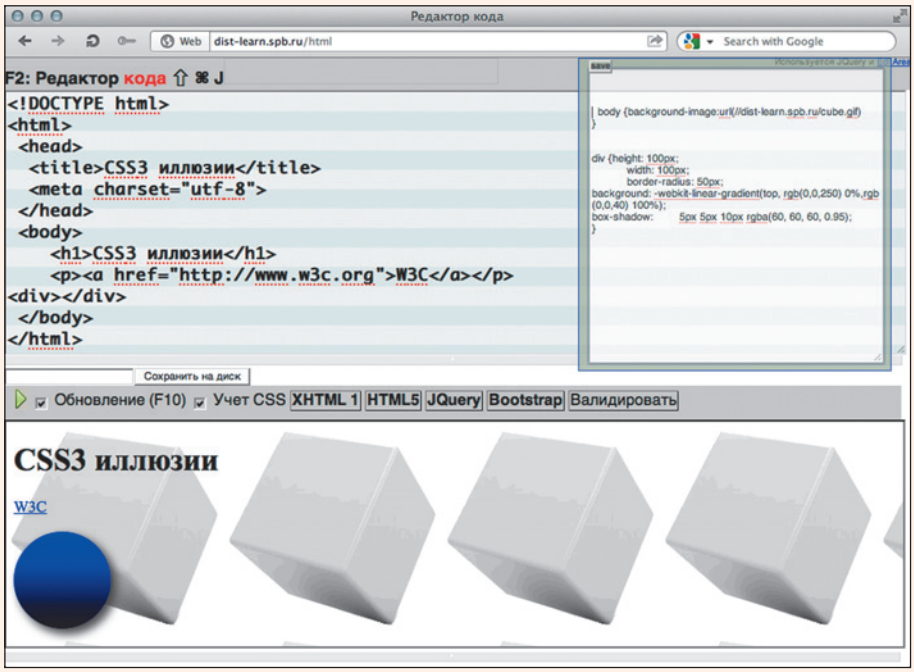
Список возможных анимаций (включая 3D-эффекты) находится по адресу *http://leaverou.github.io/animatable/*, а проверить и отладить временные параметры анимаций можно с помощью *http://www.the-art-of-web.com/css/timing-function/#*.

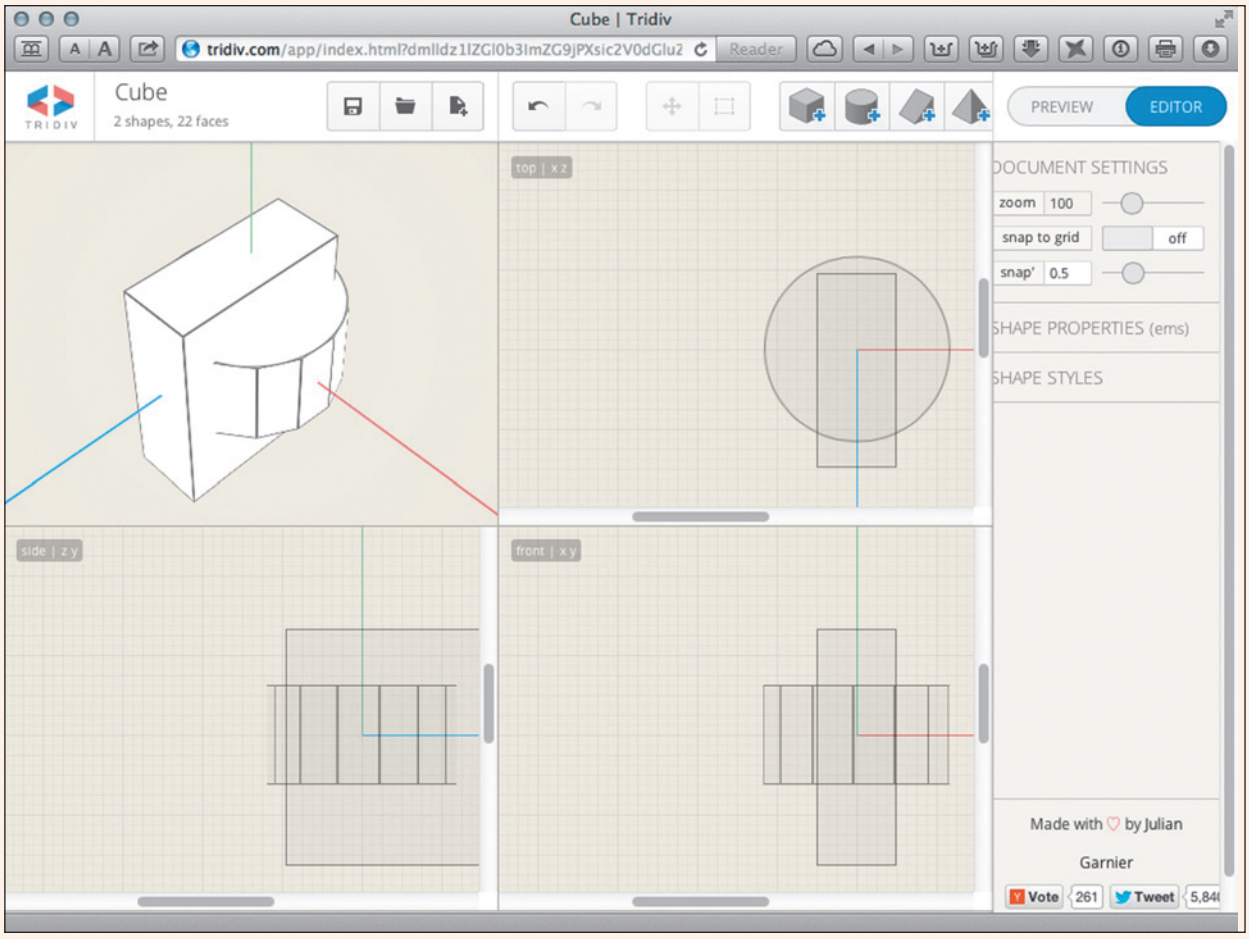
Использование динамической анимации, то есть управление свойствами каскадных стилей с помощью сценариев Javascript, позволяет достигать очень качественных эффектов, хотя фактически применяются три основных аффинных преобразования — -webkit-transform: translate, rotate и scale, то есть перенос, поворот и масштабирование, а также перспектива. Соответствующий пример (анимация трехмерных облаков) с подробной инструкцией можно найти в [3].

Таким образом, за счет использования каскадных стилей можно имитировать трехмерность, и это чисто визуальный способ, трюк. Однако уже в течение нескольких лет разрабатывается технология, позволяющая с помощью рассмотренного нами ранее тега `canvas` реализовать рендеринг настоящих трехмерных изображений.

WebGL

Можно указать два основных API, с помощью которых осуществляется разработка приложений, включающих 3D-графику. Это DirectX фирмы Microsoft и открытая кроссплатформенная библиотека OpenGL. Одна из идей реализации 3D-графики в вебе состоит в том, чтобы использовать низкоуровневые функции этих интерфейсов, вызывая их через браузерный Javascript-код. Упомянутый в начале этой статьи режим просмотра 3D-структуры страницы появился в Firefox 11 (март 2012) и был основан на применении WebGL — технологии, позволяющей через Javascript обращаться к функциям OpenGL. На момент написания данной статьи WebGL по умолчанию поддерживается в Chrome и Firefox, в Safari нужно





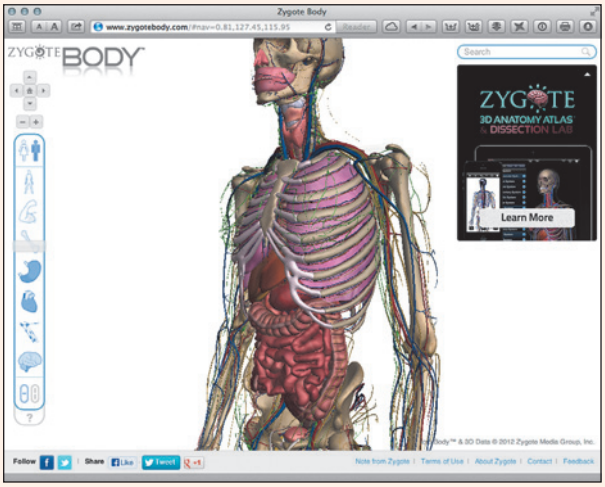
включить соответствующий пункт меню. Одним из самых ярких примеров применения WebGL является браузер тела (3D-атлас анатомии) Zygote-Body: *http://www.zygotebody.com*.

WebGL применяется как полномасштабная альтернатива Flash. Так, панорамы Google в соединении с анимацией превращаются с помощью WebGL в трехмерные анимированные экскурсии: *http://hyperlapse.tllabs.io/*.

WebGL полностью управляется с помощью событийного механизма Javascript, что позволяет реализовать интерактивность, вплоть до создания виртуальных миров, характерных для десктоп-игр: *http://alteredqualia.com/three/examples/webgl_cars.html*.

Необходимо отметить, что программирование отображения трехмерных объектов всегда крайне многословно, что связано с необходимостью покомандно реализовывать различные аспекты и этапы отображения. Это инициализация контекста, шейдеров и буферов, настройка матриц для отображения и преобразования и, собственно, вывод массивов точек в контекст. Для вывода даже самого примитивного трехмерного объекта, вроде куба или сферы, необходимо на чистом Javascript писать многие десятки строчек программного кода. Для учителя это неудачно, поскольку получается, что даже начальные примеры оказываются громоздкими и плохо мотивируют к дальнейшему изучению. Разумеется, изучение 3D-графики в школе, а тем более на веб-платформе — это удел проф-ориентированных учащихся (в рамках элективных курсов или кружков). Но данное обстоятельство не отменяет актуальности темы и ее несомненной перспективности. Количество же строк можно сократить, прибегнув к подключаемой библиотеке, где для решения многих базовых задач уже созданы функции, которые остается только вызывать с указанием тех или иных параметров.

Чтобы не утомлять читателя, мы покажем, как происходит подключение WebGL в чистом виде, а затем приведем пример с использованием сторонней библиотеки.



В нашей предыдущей статье мы использовали базовый код для обращения к Canvas, напомним его:

```
<!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>
    <canvas id='myCanvas' width='400'
     height='200'></canvas>
 </body>
</html>
```

Для работы с WebGL потребуется поменять совсем немного:

```
var ctx = canvas.getContext('experimental-
webgl');
```

После этого в переменной `ctx` будет находиться объект, свойствами и методами которого следует оперировать для осуществления всех намеченных шагов. Например, для очистки контекста вывода нужно выполнить:

```
ctx.clear(ctx.COLOR_BUFFER_BIT   |   ctx.
DEPTH_BUFFER_BIT)
```

В сценариях с WebGL используются такие команды, как bindBuffer, vertexAttribPointer, drawArrays, все весьма низкоуровневые (см. [4]), поэтому, для того чтобы сделать процесс создания объектов сколь-либо доступным, необходимо обратиться к более высокоуровневой библиотеке, например, Three JS. Библиотека в виде подключаемого Javascript-файла содержит все необходимое для оперирования 3D-объектами. Для того чтобы создать изображение куба, к примеру, будет достаточно следующего помещаемого в body кода:

```
    <script src="//www.html5canvastutori-
als.com/libraries/three.min.js"></script>
    <script>
       var renderer = new THREE.WebGLRen-
derer();
      renderer.setSize(window.innerWidth,
window.innerHeight);
      document.body.appendChild(renderer.
domElement);
        var camera = new THREE.Perspec-
tiveCamera(45, window.innerWidth / window.
innerHeight, 1, 1000);
      camera.position.z = 500;
      var scene = new THREE.Scene();
      var cube = new THREE.Mesh(new THREE.
CubeGeometry(200, 200, 200), new THREE.
MeshNormalMaterial());
      cube.overdraw = true;
      cube.rotation.y = 45;
      scene.add(cube);
      renderer.render(scene, camera);
    </script>
```

Мы видим здесь, что библиотека инициализирует `renderer` (это фактически `canvas`), после чего можно создать камеру, сцену, куб, добавить куб к сцене и осуществить рендеринг. Доступный для копирования код находится по адресу *http://pastebin.com/ZFcHq0sX*, а работающая веб-страница — по адресу *http://dist-learn.spb.ru/three.js.html*.

Читателю предлагается доработать данный пример следующим образом:

- создать кнопки “+” и “–” для изменения поворота (rotation) по всем трем осям;
- вызывать `renderer.render` всякий раз после изменения поворота;
- добавить возможность автоматического изменения угла (анимацию).

Тем самым будет реализовано простейшее управление ракурсом просмотра трехмерного объекта. Более подробные инструкции и примеры можно найти в [5].

Итак, еще раз отметим, что тема рендеринга трехмерных объектов, вообще говоря, требует соответствующей математической подготовки, а необходимость оперировать сценариями — подготовки в области Javascript. В то же время, для создания несложных сцен достаточно понимания общей схемы работы примеров и внесения фрагментарных изменений. Представляется актуальной разработка элективного курса для информационно-технологического профиля, посвященного разработке 3D-приложений на основе WebGL.

Литературные и интернет-источники

1. *Государев И.Б.* Теги и холст (SVG vs Canvas). Декларативная и динамическая веб-графика // “Информатика” № 7–8, 2013.

2. Простая анимация псевдотрехмерного объекта (шара) [Электронный ресурс] *URL: http://habrahabr.ru/post/168885/* (дата обращения: 01.10.2013).

3. CSS3 3D-преобразования [Электронный ресурс] *URL: http://habrahabr.ru/post/144585/* (дата обращения: 01.10.2013).

4. Создание сцены по шагам на чистом Javascript [Электронный ресурс] *URL: http://learningwebgl.com/blog/?p=28* (дата обращения: 01.10.2013).

5. Примеры работы с Three JS [Электронный ресурс] *URL: http://www.html5canvastutorials.com/three/html5-canvas-webgl-rotating-cube/* (дата обращения: 01.10.2013).

Индустрия: черепаха или киты

И.А. Сукин,
г. Переславль-Залесский

Введение

Сейчас мы уже привыкли к тому, что новые языки программирования появляются как грибы после дождя, а полет фантазии и конструкторской мысли их создателей ограничен только принципиальными законами классической физики и информатики. Широкое, повсеместное распространение как вычислительной техники, так и знаний о ней, о ее устройстве и принципах ее работы позволило почти каждому мечтателю воплотить в жизнь свои, порой не самые далекие от безумных, идеи. Однако так было не всегда, и еще четверть века назад проектирование языков программирования было делом избранных, им занимались крайне искушенные в предмете специалисты, продумывавшие каждый элемент, каждый оператор, каждое ключевое слово, каждую малейшую крупицу своего творения. В немалой степени такое различие между эпохами обусловлено тем, что в прошлом программирование служило в основном нуждам индустрии и корпорациям, теперь же к нему смогли сполна прикоснуться и простые смертные.

Разумеется, многим “простым смертным” рано или поздно становится интересно, как все начиналось, какие корни имеет их любимый язык программирования и в каком виде индустриальная разработка пребывает в наше время. Ответы на эти вопросы и просты, и сложны одновременно. Просты они по той причине, что индустрия — это огромная неповоротливая махина, которая чрезвычайно не любит меняться, а пытаться повернуть ее — все равно что в одиночку изменить курс океанского лайнера. Сложны же эти ответы потому, что, как и в любой громоздкой системе, в индустрии крайне высоки требования к способам описания ее элементов и внутренних связей, поэтому от проектировщиков требуются незаурядное мастерство и филигранная точность вкупе с очень четким взглядом в будущее.

Производственные языки программирования появились на самой заре эры прикладной информатики, и,

разумеется, в самом начале они были исключительно машинными. Громоздкие машины, занимавшие огромные залы вычислительных центров, понимали только совокупности электрических сигналов, управляемых топологией ключей и перемычек. Человеку, оператору, приходилось подстраиваться под привычки и желания компьютера. Шли годы, аппаратные компоненты машин дешевели, появлялись новые технологии, на кристаллах микросхем инженеры умудрялись размещать все большее и большее количество логических элементов, а человеческий труд тем временем оставался таким же дорогим. Все это привело к появлению языков программирования высокого уровня, которые значительно облегчили жизнь программистам и позволили им использовать свое время более эффективно. Примерно в это же время компьютеры вышли за границы исследовательских лабораторий и вычислительных центров и, если так можно сказать, "миниатюризировались" настолько, что стали доступны более широкой аудитории. Если до того момента позволить себе ЭВМ могли только узкоспециализированные или очень богатые корпорации, то после этой, не слишком громко афишируемой, революции возможность автоматизировать свою производственную деятельность получили многие.

В этой статье я рассмотрю основные знаковые языки программирования, оказавшие влияние на мышление, на индустрию, оценю то, как сама индустрия сказалась на их развитии. Немаловажно будет отметить, почему в свое время были приняты те или иные технические решения при реализации этих языков, какую жизнь имели результаты этих решений и каким оказался итог их внедрения для всего проекта.

Больше истории!

Вся наша история началась в не самое хорошее для всего человечества время — в эпоху Второй мировой войны, причем вовсе не на стороне тех, кого мы сейчас знаем как победителей. Между 1936-м и 1945 годами немецкий инженер Конрад Цузе разработал несколько вычислительных машин, предназначенных в первую очередь для нужд оборонного комплекса Германии. Возможно, к счастью для нас, ни одна из этих машин не осуществила свою задачу, однако Цузе придумал для них первый язык программирования высокого уровня и назвал его Plankalkul, "Программное исчисление", черпая свое вдохновение из работы Готтлоба Фреге по логике и основаниям математики "Begriffsschrift". Plankalkul, если можно сказать, значительно опередил свое время и поддерживал такие возможности, как использование подпрограмм (структурное программирование), использование оператора присваивания (вместо аналога машинного MOV), циклы, ветвления, операции над массивами и связными списками, что было недоступно в языках высокого уровня еще в течение значительного времени. К сожалению, Цузе опубликовал работу, посвященную своему языку, только в 1972 году, и особого идеологического влияния на другие проекты Plankalkul не имел. Тем не менее он остается в истории в качестве некоего памятника не только первому языку высокого уровня, но и первому языку, создававшемуся для серьезного промышленного применения.

Простейшим типом данных в Plankalkul был единичный бит, на основании которого строились более сложные типы, такие, как целые числа и числа с плавающей запятой, при этом вещественные числа представлялись в дополнительном коде и использовали "скрытый" бит мантиссы задолго до появления стандарта IEEE. В качестве составных типов данных в язык были включены массивы, списки и записи, которые в числе прочего могли быть рекурсивно вложенными. Plankalkul не содержал оператора `goto` (на радость Эдсгеру Дейкстре) и реализовывал вместо этого полноценное ветвление и циклы со счетчиками с возможностью выхода на произвольный уровень во вложенном цикле. Одним из действительно поражающих наше сегодняшнее воображение фактов является наличие в Plankalkul утверждений (assertions), представляющих собой математические выражения (равенства и неравенства), которые должны быть истинными в определенных точках программы. В других языках эта возможность была реализована только спустя 25 лет. Запись программ в Plankalkul выглядит несколько непривычно для человека, знакомого с современными популярными языками программирования, однако люди, имевшие дело со ставшими не так давно довольно известными системами автоматического доказательства теорем с зависимыми типами, найдут в ней что-то отдаленно привычное. В первой строке оператора указывается собственно операция, во второй строке — индексы элементов массивов, входящих в выражение в первой строке, а в третьей строке — типы этих самых элементов. Таким образом, запись становится двухмерной. В качестве примера рассмотрим программу, присваивающую значение выражения `A(3) + 2` переменной `A(6)`.

```
   | A + 1 => A
V  | 3       5
S  | 1.n     1.n
```

Здесь `1.n` обозначает целый тип данных, состоящий из *n* битов.

Fortran

Одним из самых больших достижений практической информатики XX века являлось, безусловно, создание в 1954 году языка программирования высокого уровня под названием FORTRAN (***FOR****mula* ***TRAN****slator* — преобразователь формул [в машин-

ный язык]). У истоков его создания стояла известная всем по финансированию и разработке многих других индустриальных проектов корпорация IBM, сконструировавшая в то время знаменитую ЭВМ “IBM 704” — один из первых компьютеров, имевших аппаратную поддержку вычислений с плавающей точкой и индексной адресации.

FORTRAN 0

Планы по разработке FORTRAN существовали, конечно, еще до появления IBM 704 весной 1954 года, однако Джон Бэкус и его рабочая группа опубликовали технический отчет, озаглавленный как “*The IBM Mathematical FORmula TRANslating System*: *FORTRAN*” только в ноябре. В этой работе содержалось описание того, что сейчас принято называть FORTRAN 0. Бэкус и его коллеги были крайне оптимистично настроены, они утверждали, что их система не только генерирует код, по эффективности сравнимый с кодом, написанным руками, по удобству и простоте программирования сопоставимый с интерпретируемыми макросистемами, что FORTRAN будет устранять ошибки программиста и содержать развитую систему отладки. В какой-то степени эти надежды были удовлетворены, в какой-то — стали “Святым Граалем” для всех последующих поколений разработчиков.

Важно понять, что FORTRAN появился и развивался в следующей обстановке:

- компьютеры были медленными и не слишком надежными;
- компьютеры применялись в основном для научных расчетов;
- не существовало никаких относительно эффективных способов программирования;
- время машины все еще было дороже, чем время программиста, поэтому требовался компилятор, генерирующий очень быстрый код.

В дальнейшем мы проследим историю развития FORTRAN до его современного вида.

FORTRAN I

FORTRAN 0 претерпел значительные изменения на протяжении всего периода своей жизни, с момента официального утверждения в январе 1955 года и до появления первого компилятора в апреле 1957 года. Язык, получившийся в результате этих метаморфоз, принято теперь называть FORTRAN I, и он описан в самом первом руководстве пользователя по FORTRAN (“Programmer’s Reference Manual”, IBM, 1956). FORTRAN I содержал средства форматированного ввода-вывода, поддерживал идентификаторы переменных длиной до шести символов (в FORTRAN 0 было более жесткое ограничение в два символа), подпрограммы (однако без возможности их раздельной компиляции), условный оператор `IF` и оператор организации цикла `DO`.

Условный оператор в FORTRAN I был некоторым “шагом назад”. В FORTRAN 0 выражение под `IF` было логическим, и в нем могли использоваться операции отношений, такие, как “>” или “<”. К сожалению, символы “>” и “<” отсутствовали в алфавите ЭВМ IBM 704, и перед разработчиками встала необходимость заменить логический оператор `IF` арифметическим, тем самым знаменитым “трехметочным `IF` из FORTRAN”. Этот оператор имеет следующий вид:

```
IF (арифметическое выражение) N1, N2, N3
```

— где с помощью `N1`, `N2` и `N3` обозначены *метки* (адреса для перехода) в программе. Если значение выражения было отрицательным, то осуществлялся переход на метку `N1`, если оно было равно нулю — на метку `N2`, если положительным — на `N3`. Такой трехметочный оператор `IF` фактически дожил в составе FORTRAN до наших дней, хотя и был объявлен устаревшим в стандарте Fortran 90.

Оператор организации цикла имел такой вид:

```
DO N1 переменная = нач_зн, кон_зн
```

— где метка `N1` — адрес последнего оператора в теле цикла. Как можно видеть, это обычный оператор организации цикла со счетчиком. Цикл с проверкой условия не был введен в состав FORTRAN I по причине неэффективности компиляции такого оператора в машинный код IBM 704. Интересна тесная связь между этими двумя проектами, однако сейчас уже никто не помнит, были ли синтаксис и семантика FORTRAN навязаны разработчиками IBM 704, или же создатели языка оказали влияние на инженеров “голубого гиганта”.

С самого начала в FORTRAN была известная многим “фиксированная форма записи программ”, связанная с необходимостью ввода программ с перфокарт (*рис.* 1 на с. 23). Длина строки была ограничена 80 символами. Если первый символ в строке был “C”, то строка воспринималась как комментарий. Во всех других строках:

- первые пять полей занимала опциональная метка (число);
- если в поле 6 находился непустой символ, текущая перфокарта считалась продолжением предыдущей;
- в полях 7–72 находился собственно оператор;
- поля 73–80 игнорировались, и обычно туда записывалась информация, позволяющая идентифицировать карту.

Также одной из знакомых многим программистам на FORTRAN особенностей было то, что FORTRAN I не имел способов указания типов переменных. В качестве определяющего элемента синтаксиса использовался так называемый “сигил” (*sigil*, здесь я передаю привет всем программистам на Perl), то есть первый символ в имени переменной. Переменные, начинавшиеся на `I`, `J`, `K`, `L`, `M`, `N`, считались целыми, а все остальные — вещественными. Такое решение обусловлено тем, что целые переменные, как правило, были индексами.

Несмотря на скептическое отношение к FORTRAN программистов старой школы, привыкших все реализовывать напрямую в машинных кодах, компилятор FORTRAN I и впрямь был одним из

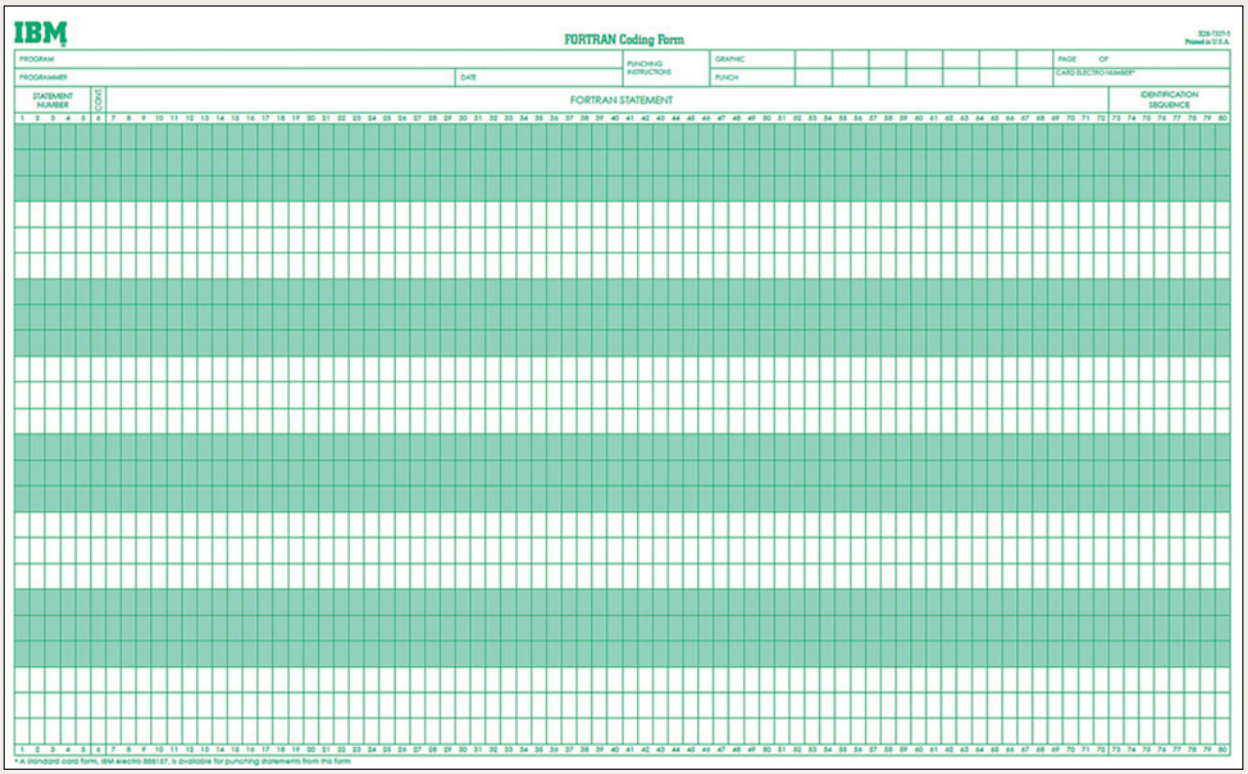


Рис. 1. Форма для записи FORTRAN-программ

самых серьезных достижений инженерной мысли, генерируя невероятно быстрый код. Уже в апреле 1958 года больше половины программ для IBM 704 были написаны на FORTRAN.

FORTRAN II

FORTRAN II представляет собой дальнейшее развитие FORTRAN I и появился весной 1958 года. В нем было исправлено множество ошибок предыдущей версии и добавлена возможность раздельной компиляции подпрограмм. Теперь при изменении только одной подпрограммы больше не требовалось перекомпилировать все остальные (это также позволило использовать в программах на FORTRAN подпрограммы, написанные на машинных языках). Проблема раздельной компиляции серьезно ограничивала размер программы — на FORTRAN I можно было написать не более 300–400 строк кода, после чего компиляция становилась слишком затратной. В качестве примера кода на FORTRAN II можно привести программу, рассчитывающую площадь треугольника по формуле Герона.

```
C ИСПОЛЬЗУЕМ СТАНДАРТНУЮ ФУНКЦИЮ SQRT
C ВВОД — СЧИТЫВАТЕЛЬ КАРТ 5, ЦЕЛЫЕ ЧИСЛА
C ВЫВОД — ПРИНТЕР КАРТ 6, ВЕЩЕСТВЕННЫЕ ЧИСЛА
      READ INPUT TAPE 5, 501, IA, IB, IC
  501 FORMAT (3I5)
C IA, IB И IC НЕ ДОЛЖНЫ БЫТЬ ОТРИЦАТЕЛЬНЫМИ
C КРОМЕ ТОГО, СУММА ДЛИН ДВУХ СТОРОН ТРЕУГОЛЬНИКА
C ДОЛЖНА БЫТЬ БОЛЬШЕ ДЛИНЫ ТРЕТЬЕЙ СТОРОНЫ
      IF (IA) 777, 777, 701
  701 IF (IB) 777, 777, 702
  702 IF (IC) 777, 777, 703
  703 IF (IA+IB-IC) 777,777,704
  704 IF (IA+IC-IB) 777,777,705
  705 IF (IB+IC-IA) 777,777,799
C ВЫХОД, ЕСЛИ ПРОЧИТАЛИ НЕВЕРНЫЕ ПАРАМЕТРЫ
  777 STOP 1
C ПОСЧИТАЕМ ПЛОЩАДЬ ПО ФОРМУЛЕ ГЕРОНА
  799 S = FLOATF (IA + IB + IC) / 2.0
      AREA = SQRT(S * (S — FLOATF(IA)) * (S — FLOATF(IB)) *
     +      (S — FLOATF(IC)))
```

```
      WRITE OUTPUT TAPE 6, 601, IA,
                           IB, IC, AREA
601 FORMAT (4H A= ,I5,5H  B= ,I5,5H
                C= ,I5,8H  AREA= ,F10.2,
     +          13H SQUARE UNITS)
      STOP
      END
```

FORTRAN IV, FORTRAN 66, FORTRAN 77

FORTRAN III, хоть и был разработан, никогда не имел значительного распространения. FORTRAN IV, появившийся в период с 1960-го по 1962 год, наоборот, стал одним из самых популярных вариантов FORTRAN и до сих пор считается его первым действительно успешным и стабильным стандартом. В нем впервые появились возможность объявления типов переменных, логический оператор IF и соответствующий ему логический тип данных, а также возможность передавать подпрограммы в качестве параметров других подпрограмм.

Другим знаковым событием в ранней истории FORTRAN было решение Американской ассоциации по стандартизации (теперь *ANSI*) в сотрудничестве и при спонсорстве Ассоциации производителей торгового оборудования (BEMA) о создании комитета, основной целью которого было написание “Американского стандарта” для языка FORTRAN. В 1966 году комитет завершил работу над стандартом, и новый язык, практически полностью унаследованный от FORTRAN IV, был назван FORTRAN 66.

После официальной стандартизации разработчики вводили свои расширения в язык. К 1969 году ситуация стала такой, что комитет ANSI принял решение о создании нового стандарта FORTRAN. Работа продолжалась долгих восемь лет и вылилась в FORTRAN 77, который до сих пор является стандартом de facto среди подавляющего большинства разработчиков на FORTRAN. Модификация привнесла следующие изменения:

- блочный оператор ветвления, в дополнение к меточному, операторы `ELSE` и `ELSE IF`;
- различные улучшения оператора цикла, такие, как введение параметров и отрицательный шаг;
- прямой файловый ввод-вывод;
- средства для работы с символьными строками;
- устойчивые локальные переменные;
- возможность неявного указания типов.

Стандарт FORTRAN 77 уверенно вывел FORTRAN на сцену научных вычислений в качестве фаворита и обеспечивает ему такую репутацию до наших времен.

Fortran 90

Fortran 90 был первым стандартом, кардинально поменявшим вид языка Fortran (как минимум его название стало содержать только одну заглавную букву), а его создание было обусловлено крайне архаичным видом программ на FORTRAN 77 по сравнению с программами на языках, появившихся к началу 1990-х годов. В 1992 году уже знакомый нам комитет ANSI опубликовал документ, описывающий наиболее существенные изменения Fortran за всю его историю. В числе этих изменений были:

- возможность писать программы в свободной форме, не привязываясь к номерам строковых полей, указывая идентификаторы и ключевые слова строчными буквами и т.п.;
- максимальная длина идентификатора была увеличена до 31 символа;
- рекурсивные подпрограммы, поддержка необязательных и именованных параметров;
- поддержка *срезов* массивов;
- поддержка программных модулей;
- перегрузка операторов;
- динамическое выделение памяти под массивы и изменение их размера;
- оператор `CASE`;
- поддержка множества встроенных функций для работы с массивами и матрицами.

Кроме того, в Fortran 90 появилась довольно важная концепция объявления некоторых свойств и элементов языка устаревшими и нерекомендуемыми к использованию. При этом устаревшие элементы должны быть удалены в следующем стандарте, а нерекомендуемые к использованию — через один стандарт.

Fortran 95, Fortran 2003, Fortran 2008 и Fortran 2015

Следующим после Fortran 90 стал стандарт Fortran 95, в котором добавились конструкции, облегчающие распараллеливание и векторизацию программ, такие, как конструкция FORALL. Основой стандарта стал *High Performance Fortran* — вариант языка для суперкомпьютеров. Fortran 95 является наиболее распространенным стандартом Fortran, поддерживаемым большинством современных компиляторов. Стандарт 2003 года внес в язык поддержку объектно ориентированного программирования, взаимодействия с программами на Си, честной IEEE арифметики.

Fortran 2008 опять расширил возможности языка в области параллельного программирования, включив расширение Coarray Fortan. Программы на этом языке выполняются так, будто бы они работают на множестве вычислителей асинхронно и параллельно. Каждый экземпляр программы при этом имеет свою копию данных, называемую *изображением*. В качестве примера можно рассмотреть следующую простую программу.

```
program Hello_World
  implicit none
  integer :: i  ! Локальная переменная
  character(len=20) :: name[*] ! скалярный сомассив, каждое изображение
  ! имеет собственную копию этой переменной
  ! Важно: "name" — это локальная переменная, а "name[<index>]" — переменная
  ! в определенном изображении.
  ! "name[this_image()]" — это то же самое, что и "name".
  ! Только для пользователя изображения №1.
  if (this_image() == 1) then
    write(*,'(a)',advance='no') 'Enter your name: '
    read(*,'(a)') name
    ! Отдадим полученную информацию остальным изображениям
    do i = 2, num_images()
      name[i] = name
    end do
  end if
  sync all ! Синхронизируем изображения
  ! Выведем приветствие от каждого изображения
  write(*,'(3a,i0)') 'Hello ',trim(name),' from image ', this_image()
end program Hello_world
```

Кроме этого, был введен оператор DO CONCURRENT, позволяющий явно указывать на возможность параллельного выполнения итераций цикла, если они не зависят друг от друга. Находящийся в настоящее время в разработке стандарт Fortran 2015 должен привнести дальнейшие улучшения в развитие параллелизма (со стороны Coarray Fortran) в Fortran и во взаимодействие с программами, написанными на других языках. Другим важным направлением развития языка является создание индустриального стандарта Fortran для работы в системах реального времени.

Что написано на Fortran?

Почему я посвятил языку Fortran такой большой объем материала? Потому что количество программного кода, написанного на нем, сложно переоценить. Он остается языком de facto для программирования суперкомпьютеров, астрономических и метеорологических расчетов, моделирования климата, компьютерной линейной алгебры, оптимизации, вычислительной химии, вычислительной механики жидкостей, экономики, физики и многих других областей. Большая часть бенчмарков (программ для определения максимальной производительности) для суперкомпьютеров до сих пор написана на Fortran. К языку до сих пор проявляют стабильно высокий интерес такие корпорации, как IBM, Intel, AMD, Fujitsu, и, пожалуй, сложно назвать язык, более популярный в индустриальной среде.

В чем причины такой популярности? Ответ на этот вопрос прост: Fortran предоставляет удивительно ясную и чистую с инженерной и научной точки зрения формальную систему для создания программ, не ухудшая при этом производительность откомпилированного кода, позволяет легко писать высокопараллельные приложения, имеет за своей спиной долгую историю, полную проб, ошибок, мягких и радикальных изменений. Можно быть уверенным, что этот язык еще долго будет жить и занимать свою нишу.

LISP

Следующей важной вехой в истории индустриального программирования стало появление в сугубо академической среде языка LISP. Его разработка была по большей части сподвигнута массовым ростом интереса к проблемам искусственного интеллекта, который привел сначала к созданию фактически машинного (однако для машины несколько более “высокого” уровня) языка IPL, а затем и LISP’а. Самым важным инструментом в этой области была обработка связных списков с элементами-символами (символьными строками).

Корпорация IBM проявляла значительный интерес к искусственному интеллекту, в первую очередь со стороны методов автоматического доказательства теорем, однако ее инженеры посчитали, что дешевле будет расширить уже имевшийся в их распоряжении FORTRAN возможностями по обработке списков, и реализовали язык FLPL (***FORTRAN List Processing Language***), который некоторое время использовался для доказательства простейших теорем планиметрии.

Примерно в это же время, летом 1958 года, в исследовательский отдел был принят Джон Маккарти из Массачусетсского технологического института. Он должен был заниматься разработкой формальных конструктивных требований для системы символьных вычислений. В качестве пробной проб-

лемной области Маккарти выбрал дифференциальное исчисление, и, как потом оказалось, не зря. В ходе изучения задач в этой области и способов их решения он сформулировал основные требования к языку реализации: поддержка рекурсии и логических условных выражений. Единственный язык высокого уровня в тот момент, FORTRAN I, не имел такой поддержки. Другой ряд требований относился к присутствию в языке связных списков, динамического выделения памяти для них и поддержки автоматического освобождения этой памяти. Излишние ручные операции по явному управлению памяти слишком отвлекали бы разработчиков от создания аккуратных алгоритмов искусственного интеллекта, таких, как алгоритм символьного дифференцирования. Ничего из этого в FLPL не было, поэтому Маккарти твердо решил создать принципиально новый язык. Однако первая версия LISP родилась уже не в IBM, а в MIT, в сотрудничестве с Марвином Минским.

Common Lisp

Очень долгое время LISP оставался академической игрушкой для сотрудников лаборатории искусственного интеллекта MIT. Он не имел единой формы, стандартного синтаксиса или семантики, включал в себя только два типа данных: символы (атомы) и списки символов и ограниченное число функций над значениями этих типов.

В начале восьмидесятых годов существовало уже несколько коммерческих серьезных реализаций LISP от разных корпораций, например, Zetalisp для LISP-машин от Symbolics, Interlisp. К 1984 году известный специалист в прикладной информатике Гай Стил решил наконец разработать единый стандарт на LISP. Его попытка вылилась в диалект Common Lisp, который был стандартизован ANSI.

В Common Lisp содержится невероятное количество мощных языковых средств для поддержки всех возможных парадигм и способов программирования:

- простые и составные типы данных;
- поддержка сложных синтаксических конструкций для составных данных;
- поддержка объектно ориентированного программирования;
- поддержка обобщенного (generic) программирования;
- поддержка сложной системы макросов для генерации кода “на лету”;
- поддержка системы рестартов для обеспечения качественной обработки ошибок;
- мощные средства отладки.

В качестве примера кода на Common Lisp приведу (без комментариев) отрывок из подсистемы машинной индукции системы автоматического доказательства теорем ACL2:

```
(defun unchangeables (formals args quick-block-info subset ans)
; We compute the set of all variable names occurring in args in
; unchanging measured formal positions.  We accumulate the answer onto
; ans.
  (cond ((null formals) ans)
        ((and (member-eq (car formals) subset)
              (eq (car quick-block-info) 'unchanging))
         (unchangeables (cdr formals) (cdr args) (cdr quick-block-info) subset
                        (all-vars1 (car args) ans)))
        (t
         (unchangeables (cdr formals) (cdr args) (cdr quick-block-info) subset
                        ans))))
(defun changeables (formals args quick-block-info subset ans)
; We compute the args in changing measured formal positions.  We
; accumulate the answer onto ans.
  (cond ((null formals) ans)
        ((and (member-eq (car formals) subset)
              (not (eq (car quick-block-info) 'unchanging)))
         (changeables (cdr formals) (cdr args) (cdr quick-block-info)
                      subset
                      (cons (car args) ans)))
        (t
         (changeables (cdr formals) (cdr args) (cdr quick-block-info)
                      subset
                      ans))))
```

В настоящее время Common Lisp имеет хоть и не такое широкое, но достаточно устоявшееся применение в системах, где требуется совмещение огромной гибкости с достаточной надежностью. В качестве хрестоматийного примера обычно приводится случай отладки “на живую” программы на LISP, ра-

ботавшей на космическом аппарате в ста миллионах километров от Земли. Ни в одном другом индустриальном языке подобное не было бы возможно, поэтому космическая отрасль очень любит LISP.

ALGOL

К концу пятидесятых годов двадцатого века в индустрии программирования насчитывалось уже как минимум три полноценных языка высокого уровня, самым перспективным из которых был FORTRAN, всецело принадлежавший американской корпорации IBM. Это вызвало серьезные споры среди европейцев, которым хотелось бы иметь независимый язык программирования общего назначения.

ALGOL 58

На встрече европейского Общества прикладной математики и механики GAMM с американской ACM в конце мая 1958 года в Цюрихе были приняты следующие положения, касающиеся разработки нового языка:

- синтаксис языка должен быть максимально близок к общепринятой математической нотации, а программы на нем должны читаться с минимальными пояснениями;
- язык должен использоваться для описания вычислительных процессов в публикациях;
- программы на этом языке должны транслироваться в машинные языки.

Если первая и третья цели были уже привычными и покоренными Фортраном, то вторая была чем-то совершенно новым в индустрии. На встрече в Цюрихе обсуждение нового языка вызвало крайне бурную дискуссию, результатом которой стал язык IAL (***I**nternational **A**lgorithmic **L**anguage*, Международный Алгоритмический Язык). В качестве варианта предлагалось также название ALGOL (***ALGO**rithmic **L**anguage*), однако оно было отклонено, поскольку не отражало международного духа языка. Несмотря на это, уже в течение года вариант ALGOL стал преобладающим и язык получил название ALGOL 58. ALGOL был одновременно и наследником, и "европейским ответом" FORTRAN.

Главными отличиями ALGOL от FORTRAN тех времен были формальная поддержка типизации переменных, наличие составных операторов, поддержка имен идентификаторов произвольной длины и массивов произвольного числа размерностей (при этом граница массива могла определяться программистом), поддержка вложенных операторов ветвления. ALGOL позаимствовал, с некоторым изменением, из Plankalkul синтаксис для оператора присваивания ":=".

Проект ALGOL 58 с большим энтузиазмом приняли американские программисты. В США ALGOL лег в основу трех важных языков: MAD из университета Мичигана, NELIAC для американской военно-морской электронной группы и JOVIAL для научных разработок ВВС США. Отношение IBM к новому языку было довольно сложным. Сначала в "голубом гиганте" предложили стандартизовать ALGOL 58 и реализовали его для своих компьютеров 700-й серии, однако уже в середине 1959 года было принято решение отказаться от ALGOL в пользу FORTRAN.

AlGOL 60

В 1959 году на Международной конференции по обработке информации Джон Бэкус представил доклад, в котором была представлена новая форма описания синтаксиса языков программирования, теперь известная как форма Бэкуса — Наура. Соавтор Бэкуса, Питер Наур, какое-то время размышлял над возможностью применить эту форму к разработке очередной версии языка программирования ALGOL и в январе 1960 года предложил свои изменения на собрании комитета по разработке этого языка. Среди нововведений языка, названного ALGOL 60, оказались следующие вещи:

- концепция локальных программных блоков;
- передача параметра в подпрограмму по значению и по имени;
- поддержка рекурсии;
- поддержка автоматических массивов, диапазон изменения индексов которых определяется во время выполнения значениями переменных;
- операторы форматированного ввода-вывода были отклонены как слишком зависящие от особенностей машины.

ALGOL 60 стал единственным официальным средством представления алгоритмов в научной литературе на следующие 20 лет, а каждый язык программирования, созданный после 1960 года, что-то позаимствовал у него. Это был исторический момент. Впервые язык программирования был создан международной инициативной группой, впервые язык имел формально описанный синтаксис и впервые язык программирования был машинно-независимым. Минусами ALGOL 60 являлись его невероятный объем, который не был реализован практически ни в одном компиляторе, сложность реализации элементов, зависящих от машины, и практическая невозможность конкуренции с FORTRAN. Рассмотрим такой пример программы на ALGOL 60:

```
comment Пример программы на ALGOL 60
Ввод:  целое число, меньшее ста, определяющее длину списка, элементы списка
Вывод: Количество элементов списка, больших, чем среднее арифметическое;
begin
  integer array intlist [1:99];
```

```
  integer listlen, counter, sum, average, result;
  sum    := 0;
  result := 0;
  readint(listlen)
  if (listlen > 0) ^ (listlen < 100) then
    begin
    comment Считывание элементов списка;
    for counter := 1 step 1 until listlen do
      begin
      readint(intlist[counter]);
      sum := sum + intlist[counter];
      end;
    comment вычисление среднего;
    average := sum / listlen;
    comment вычисление результата;
    for counter := 1 step 1 until listlen do
      if intlist[counter] > average
        then result := result + 1;
    comment вывод результатов;
    printstring("Число элементов, больших среднего значения:");
    printint(result);
    end
  else
    printstring("Ошибка, неправильная длина списка");
end
```

ALGOL 68 и ALGOL-W

Разработка ALGOL не закончилась в 1960 году. Через восемь лет ван Вийнгаарденом была представлена новая версия — ALGOL 68, основной упор в котором был сделан на ортогональность, то есть на тотальную унификацию всего и вся и предоставление программисту как можно более универсальных инструментов для выполнения задачи. ALGOL 68 предоставлял небольшой набор операторов и типов данных и практически неограниченные возможности их комбинирования таким образом, чтобы получались работающие программы. В силу большой сложности разработки соответствующего компилятора и отсутствия важных покровителей в лице крупных корпораций, ALGOL 68 не стал слишком распространенным языком.

В ALGOL 68 были включены такие возможности, как

- динамические массивы и срезы массивов;
- сложная система преобразования типов;
- поддержка определения собственных операторов;
- поддержка параллельного исполнения.

ALGOL-W был другим ответвлением от ALGOL 60, в этот раз в сторону простоты. Автором этого языка был Никлаус Вирт, создавший впоследствии на его основе Pascal.

В качестве примера кода на ALGOL 68 рассмотрим классический пример синхронизации потоков “производитель – потребитель” с использованием семафоров. Здесь производитель увеличивает переменную на единицу, а потребитель — уменьшает.

```
BEGIN
  INT n := 0, SEMA consume = LEVEL 0,
                   produce = LEVEL 1;
  PAR BEGIN
    DO DOWN produce;
      print (n +:= 1);
      UP consume
    OD,
    DO DOWN consume;
      print (n -:= 1);
      UP produce
    OD
  END
END
```

COBOL

Одним из самых необычных языков своего времени стал COBOL. Несмотря на свое широкое распространение, он очень слабо повлиял на последующие языки программирования. Его можно назвать самым фундаментальным языком разработки коммерческих приложений.

В 1959 году в сфере коммерции существовало лишь небольшое число макроязыков, как несколькими годами ранее, до Фортрана, в научной сфере. Прародителем COBOL был язык FLOW-MATIC, разработанный Грейс Хоппер из корпорации Remington-Rand UNIVAC. В 1953 году она высказала судьбоносное мнение о том, что “математические программы следует писать с использованием математической нотации, программы же для обработки

данных нужно писать на английском языке". В то время это звучало совершенно фантастически, но Хоппер сумела убедить руководство компании в справедливости своих слов.

В мае 1959 года, через год после собрания разработчиков ALGOL в Цюрихе, Министерство обороны США провело в Пентагоне встречу, посвященную разработке языка для финансовой и коммерческой отрасли. Формировавшийся язык было решено назвать CBL, а в качестве основных требований выдвигались приближенность синтаксиса к естественному языку, даже в ущерб скорости работы, и необходимость преодолевать ограничения ЭВМ тех времен. В языке должны были быть отдельные разделы для описания данных и описания подпрограмм, а понятия индексов массивов и математические выражения должны были быть переведены на английский.

Соответствующий язык был реализован в кратчайшие сроки и был в итоге назван COBOL 60. В нем впервые появились зачатки макросов, иерархических структур данных и человекочитаемых имен переменных. Описание данных вообще всегда было самой сильной стороной COBOL, числа описывались с точностью до числа разрядов, а положение десятичной точки задавалось программистом. Файловые типы данных описывались крайне подробно, для улучшенной генерации отчетов. Кроме всего прочего, COBOL был первым языком программирования, созданным при поддержке Министерства обороны США и получившим, тем самым, огромнейший финансовый и информационный толчок уже на самом старте.

Кроме самого первого варианта, для языка COBOL было выпущено еще несколько стандартов: COBOL 1968, COBOL 1974, COBOL 1985 и COBOL 2002. Стандарт 2002 года полностью избавился от наследия перфокарт — фиксированной формы записи программ, ввел инструменты объектно ориентированного программирования, средства взаимодействия с кодом на других языках, поддержку платформ .NET и Java, а также опциональную поддержку чисел с плавающей запятой (до этого в языке существовали только более важные в финансовой сфере числа с фиксированной запятой).

Огромнейшее количество программ на Коболе разработаны для правительства США, военных структур и коммерческих организаций, для операционных систем фирмы IBM, таких, как z/OS и z/VSE, UNIX, Windows. По самым скромным оценкам, в конце 1990-х годов на COBOL было написано более 80% всех бизнес-приложений, что составляло более двухсот миллиардов строк кода, при этом более пяти миллиардов новых строк появлялось ежегодно. В наше время популярность COBOL, в том числе и в бизнесе, незначительно падает, поскольку все активнее развивается среднее и малое предпринимательство, не являющееся основным заказчиком программного обеспечения, а также в силу действия некоторых старых фундаментальных ограничений языка. Сам COBOL в разное время был воспринимаем с разных сторон, например, Дейкстра считал, что обучение этому языку следует считать "преступлением".

В заключение раздела рассмотрим программу, выводящую "Hello world" посередине окна терминала:

```
  program-id. ghello.
data division.
working-storage section.
01      var     pic x(1).
01      lynz    pic 9(3).
01      colz    pic 9(3).
01      msg     pic x(15) value "Hello, world!".
procedure division.
        accept lynz from lines end-accept
        divide lynz by 2 giving lynz.
        accept colz from columns end-accept
        divide colz by 2 giving colz.
        subtract 7 from colz giving colz.
        display msg
                at line number lynz
                column number colz
        end-display
        accept var end-accept
        stop run.
```

PL/I

Безусловно, одним из самых старых "монстров" индустрии является знаменитый PL/I. В начале 60-х годов большинство программистов в науке использовали FORTRAN, а большинство бизнес-разработчиков — COBOL. Обе группы двигались навстречу друг другу по пути, который неизбежно вел к проблемам. Объем информации в науке рос экспоненциально и требовал сложных средств ввода-вывода, а экономи-

ческие приложения все сильнее и сильнее обрастали математикой. Уже знакомая нам корпорация IBM заинтересовалась ситуацией и представила разработку универсального компьютера — легендарного IBM System/360. При этом, разумеется, возникла и идея об универсальном языке программирования, призванном заменить одновременно FORTRAN, LISP, COBOL и языки ассемблера.

В 1963 году из сотрудников IBM и группы SHARE был собран комитет по разработке нового языка, названного изначально FORTRAN VI. Члены комитета были уверены, что на все про все им хватит три месяца, однако итоговый срок был увеличен до февраля 1964 года. Первая работоспособная версия компилятора языка, называвшегося уже NPL, была представлена в декабре 1964 года. В 1965 году, во избежание путаницы названий с Национальной физической лабораторией Великобритании, язык был уже окончательно переименован в PL/I.

PL/I содержал то, что считалось лучшими частями ALGOL 60 (рекурсия, блоки кода), FORTRAN IV (раздельная компиляция), COBOL 60 (развитые структуры данных, мощные средства ввода-вывода и генерации отчетов) и некоторые новые механизмы. Поскольку на PL/I возлагались надежды как на наиболее общий и универсальный язык, число его возможностей не поддается разумному перечислению. До сих пор считается, что никогда не существовало ни одного компилятора, в котором PL/I был бы реализован полностью или хотя бы почти полностью. В языке впервые были реализованы:

- параллельно выполняемые задачи (однако сама реализация была ужасной);
- мощнейшая система выявления и обработки исключений и ошибок;
- отключаемая рекурсия;
- указатели в качестве типа данных;
- обращения к подмассивам (например, к строке матрицы как к вектору).

Оглядываясь сейчас назад, сложно представить, как можно было создать такой громоздкий инструмент, как PL/I, чья структура была основана на допущении, что любая полезная структура или конструкция, которая только может быть реализована, должна быть включена в язык. Небезызвестный Эдсгер Дейкстра говорил, что PL/I достиг такого уровня, что его уже невозможно было охватить и контролировать с помощью интеллекта. Тем не менее этот язык был относительно успешен и в 70-х годах активно применялся как в коммерции, так и в науке и образовании. На нем была написана ОС Multics — предшественница UNIX.

В качестве примера кода на PL/I можно рассмотреть наивную процедуру умножения матриц:

```
/* C = A x B */
MMULT: procedure (a, b, c);
   declare (a, b, c)(*,*) float controlled;
   declare (i, j, m, n, p) fixed binary;
   if hbound(a,2) ^= hbound(b,1) then
      do;
         put skip list
            ('Нельзя перемножить такие матрицы');
         signal error;
      end;
   m = hbound(a, 1); p = hbound(b, 2);
   if allocation(c) > 0 then free c;
   allocate c(m,p);
   do  i = 1 to m;
      do j = 1 to p;
         c(i,j) = sum(a(i,*) * b(*,j) );
      end;
   end;
end MMULT;
```

APL

Существом совершенно иного рода является матричный язык APL. Впервые он был описан в книге Кеннета Айверсона “A Programming Language”, изданной в 1962 году. Кеннет находился в поиске лаконичной и однозначной математической нотации, пригодной для реализации на набиравших популярность вычислительных машинах.

APL содержит большое количество мощных операторов, работающих сразу над целыми массивами или их срезами. Операторы, как правило, обозначаются одним специальным символом, присутствующим на особой “клавиатуре APL” (см. *рис.* 2 на с. 31). Один оператор может выполнять, например, транспонирование матрицы. Считается, что большое количество крайне перегруженных семантикой операторов весьма отрицательно сказывается на читаемости программ, и APL пригоден только для быстрого прототипирования, не требующего последующего чтения и модификации кода. Апологеты языка категорически несогласны с таким утверждением и успешно применяют APL для реализации довольно сложных экономических, корпоративных и научных систем с помощью очень коротких участков кода.

APL не получил слишком широкого распространения, однако вокруг него сформировалось крепкое сообщество, включающее в себя программистов из самых крупных мировых банков, таких, как Goldman Sachs и Morgan Stanley. Язык используется в своей сфере до сих пор и не претерпел никаких значительных изменений с момента своего появления. На основе APL были также разработаны языки J и K,

Рис. 2. Клавиатура APL

рассчитанные в первую очередь на работу с ASCII-клавиатурой.

В качестве примера краткости и сложности можно привести программу, считающую все простые числа от 1 до *R*:

```
(~R∈R∘.×R)/R←1↓⍳R
```

Ada

Мы подошли к заключительной части статьи и рассмотрению самого фундаментального проекта по созданию языка программирования в истории. Конечно же это Ada, разрабатывавшаяся опять же для Министерства обороны США в уже сложившейся вычислительной среде, что сильно сказалось на форме языка.

К 1974 году больше половины компьютеров в парке вычислительной техники Министерства обороны приходилось на встроенные устройства, то есть ЭВМ для управления теми или иными технологическими объектами большой сложности. Сложность систем росла очень быстро, а вслед за ней росла и стоимость. Программисты Министерства писали на четырех сотнях различных языков, ни один из которых не был стандартизован. Это сильно затрудняло как саму разработку, так и повторное использование уже написанного кода. Такая ситуация привела к тому, что, независимо друг от друга, армия, флот и ВВС США предложили создать стандартный язык программирования высокого уровня для программирования встроенных систем.

В январе 1975 года директор Отдела исследований и проектирования Министерства обороны Малкольм Карри создал рабочую группу по созданию такого языка. В эту группу входили представители всех военных служб и служб по связи с другими странами. Уже в апреле того же года рабочая группа издала документ с требованиями к создаваемому языку, названный Strawman document (документ соломенного человека). Документ был разослан по войскам, федеральным агентствам и другим заинтересованным структурам. Уже в августе был издан Woodman document (документ деревянного человека), а в январе 1976 года — Tinman document (документ оловянного человека), в котором присутствовал уже полный список требований к языку и все его ожидаемые характеристики.

В апреле 1977 года было решено провести конкурс на разработку необходимого языка программирования. Условия конкурса были описаны в Ironman document (документ железного человека). К начальной фазе разработки были допущены четыре подрядчика: компании Softech, SRI International, Cii Honeywell/Bull и Intermetrics. Проекты всех четырех корпораций были основаны на языке Pascal.

В финал конкурса прошли двое: Intermetrics и Cii Honeywell/Bull, после чего был издан очередной документ — Steelman document (документ стального человека). Уже в мае 1979 года в качестве победителя был выбран язык, предложенный Cii Honeywell/Bull, единственного иностранного (французского) подрядчика из финалистов. Руководителем инициативной группы в этой корпорации был Жан Ишбиа. В то же время Джек Купер из командования материальным обеспечением ВМФ США предложил назвать язык Ada — в честь Августы Ады Байрон, первого признанного программиста в истории.

Сразу после этого началась финальная стадия проектирования, завершившаяся в феврале 1980 года выпуском Stoneman document (документа каменного человека), а в 1983 году язык Ada был официально стандартизован ANSI, после чего всякое его развитие прекратилось на продолжительный срок.

Ada 83

Оригинальный проект языка Ada содержал множество важных концептуальных возможностей. В него были включены пакеты из Modula-2, средства выделения в отдельные модули объектов, спецификаций типов и процедур, обеспечивающие должное сокрытие внутренних деталей. Разумеется, в Ada была сложная система обработки ошибок и исключительных ситуаций. Блоки кода могли быть обобщенными (generic, полиморфными для многих типов) и имели возможность выполняться параллельно. Грубо говоря, Ada стала тем, чем должен был быть PL/I.

Важно понять, что

- разработка языка велась в конкурентной среде с жестким отбором предложенных проектов;
- в Ada было включено огромное число возможностей из современных ей языков, причем включение это было грамотным, а не бездумным;
- разработка компилятора Ada была невероятно сложной задачей, первые серьезные проекты этого рода появились лишь спустя четыре года после финализации самого языка.

Все это способствовало созданию у Ada имиджа мощного, но слишком сложного языка.

Ada 95, Ada 2005, Ada 2012

В 1988 году было решено, что Ada нуждается в переработке. Процесс изменения проводился в три стадии: определение требований к переработанному языку, разработка описания этого языка и физическая реализация языка.

В качестве четырех основных требований были выдвинуты следующие: возможность создания интерфейсов (в том числе графических) пользователя, поддержка объектно ориентированного программирования, увеличение гибкости библиотек, усовершенствование механизмов управления совместно используемыми данными. Итоговый язык получил имя Ada 95. Стандарт 2005 года внес дополнительные расширения в уже указанные области. В Ada 2012 впервые появились механизмы контрактного программирования, аспектно ориентированного программирования и строгая поддержка реального времени. На базе Ada (на базе ее подмножества) был создан язык с поддержкой формального доказательства правильности программ SPARK.

В настоящее время Ada является не слишком широко распространенным языком, но занимает свою специфичную нишу, нишу приложений реального времени для встраиваемых систем, и не собирается уступать ее новичкам. Тем не менее популярность языка медленно, но неуклонно падает, а сам он все больше и больше обрастает возможностями, как некогда PL/I. Сейчас трудно сделать точный прогноз, но скорее всего Ada все же останется на своем месте еще на некоторое время.

В качестве примера приведем уже знакомую нам программу перемножения матриц на Ada:

```
with Ada.Text_IO;                use Ada.Text_IO;
with Ada.Numerics.Real_Arrays;   use Ada.Numerics.Real_Arrays;
procedure Matrix_Product is
   procedure Put (X : Real_Matrix) is
      type Fixed is delta 0.01 range -100.0..100.0;
   begin
      for I in X'Range (1) loop
         for J in X'Range (2) loop
            Put (Fixed'Image (Fixed (X (I, J))));
         end loop;
         New_Line;
      end loop;
   end Put;
   A : constant Real_Matrix :=
         (  ( 1.0,  1.0,  1.0,   1.0),
            ( 2.0,  4.0,  8.0,  16.0),
            ( 3.0,  9.0, 27.0,  81.0),
            ( 4.0, 16.0, 64.0, 256.0)
         );
   B : constant Real_Matrix :=
         (  (  4.0,     -3.0,      4.0/3.0,  -1.0/4.0 ),
            (-13.0/3.0, 19.0/4.0, -7.0/3.0,  11.0/24.0),
            (  3.0/2.0, -2.0,      7.0/6.0,  -1.0/4.0 ),
            ( -1.0/6.0,  1.0/4.0, -1.0/6.0,   1.0/24.0)
         );
begin
   Put (A * B);
end Matrix_Product;
```

Итоги

Что же можно сказать в итоге о медленно плывущих китах индустриального программирования? Индустрия — область, которая не хочет и не может меняться слишком быстро. Все языки в ее рамках вынуждены существовать под строгим надзором и стараться не изменять давно сложившимся принципам. Вопреки заверениям популяризаторов новых инструментов, никакие корпорации не собираются отказываться от хорошо работающих старых. Заглядывая в будущее, можно сказать, что эта ситуация не поменяется до тех пор, пока не будет изобретена вычислительная машина, основанная на радикально новых принципах, например, квантовый компьютер. До тех пор “старики” и любители строгости программного кода могут спать спокойно.

Список ссылок

1. *http://software.intel.com/en-us/blogs/2013/08/08/doctor-fortran-goes-dutch-fortran-2015* — рассуждения о будущем стандарте Fortran 2015.
2. *http://gcc.gnu.org/fortran/* — самый развитый свободный компилятор Fortran.
3. *http://www.masswerk.at/algol60/report.htm* — документ об ALGOL 60.
4. *http://jmvdveer.home.xs4all.nl/algol.html* — современная реализация ALGOL 68.
5. *http://www.microfocus.com/solutions/cobol* — компания Micro Focus, основной разработчик COBOL в наши дни.
6. *http://www.dyalog.com* — реализация APL, стандарт дефакто в наши дни.
7. *https://sites.google.com/site/baavector/home* — сайт Журнала Британской ассоциации APL “Vector”.
8. *http://www.adaic.org* — сайт, посвященный языку Ada.
9. *http://www.ada-auth.org* — сайт, посвященный Ada 2012.

Педагогический университет
«Первое сентября»

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

ДИСТАНЦИОННЫЕ КУРСЫ ПОВЫШЕНИЯ КВАЛИФИКАЦИИ

(с учетом требований ФГОС)

С 1 ноября производится прием заявок на второй поток 2013/14 учебного года

образовательные программы:

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **108** УЧЕБНЫХ ЧАСОВ

  Стоимость – 2990 руб.

- НОРМАТИВНЫЙ СРОК ОСВОЕНИЯ – **72** УЧЕБНЫХ ЧАСА

  Стоимость – 2390 руб.

По окончании выдается удостоверение о повышении квалификации установленного образца

Перечень курсов и подробности – на сайте edu.1september.ru

Пожалуйста, обратите внимание:

заявки на обучение подаются только из Личного кабинета, который можно открыть на любом сайте портала www.1september.ru

Навигационная панель — “лента” для учебного проекта

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

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

При подготовке презентационных материалов, являющихся результатами выполнения коллективного проекта, к публикации (размещению на web-сайте, на компакт-диске или ином носителе), как правило, возникает проблема сборки этих материалов в единое целое и создания удобной для пользователя (который будет просматривать собранные материалы) интерактивной оболочки.

Удобнее всего создать такую оболочку с использованием web-технологий (с использованием языка разметки гипертекста HTML, а также, если необходимо, таблиц каскадной стилевой разметки (CSS) и Java-скриптов). При этом все презентационные материалы (презентации PowerPoint, обособленные htm-файлы и размещенные в отдельных папках web-сайты, видеофрагменты, аудиофрагменты, документы Word, таблицы Excel, графические и любые другие файлы, включая архивы и исполняемые программы) “подключаются” к этой оболочке, играющей роль “содержания”, посредством гиперссылок.

Такой комплект из интерактивной пользовательской оболочки и презентационных материалов можно в дальнейшем без каких бы то ни было переделок или доработок:

- разместить на web-сайте для онлайн-просмотра через сеть Интернет или на сервере школы для просмотра материалов по локальной сети;
- записать на компакт-диск (CD, DVD) в качестве раздаточного материала;
- распространять (в исходном виде или в виде архива) на любом носителе информации.

Можно разработать пользовательскую оболочку, которая сочетала бы в себе два на первый взгляд противоречащих друг другу качества:

1) возможность выбора интересующей презентации щелчком мыши на одной из представленных миниатюр (например, уменьшенных копий первого слайда каждой презентации);

2) при просмотре выбранной презентации — максимально возможный раз-

мер области просмотра, практически во все окно браузера.

Реализовать эти возможности можно, используя технологии динамического HTML (DHTML), основанные на изменении непосредственно в процессе работы пользователя с web-страницей (которую, по сути, и представляет собой наша пользовательская оболочка) параметров отображаемых на ней объектов при помощи скриптов (например, JavaScript). При этом создание такой оболочки тоже будет представлять собой учебный проект!

Разработаем элемент управления сайтом, которому дадим название “лента” (“ribbon”) и который должен работать следующим образом:

Эскизы “ленты” в раскрытом и свернутом состоянии

1) В обычном режиме, в том числе при просмотре выбранной презентации, “лента” скрыта; внизу окна браузера располагается только узкая полоска с кнопкой раскрытия “ленты”. Соответственно, просматриваемая презентация занимает почти весь экран.

2) При щелчке мышью на кнопке раскрытия “ленты” она появляется внизу окна браузера (соответственно, размеры области, занятой ранее просматриваемой презентацией, уменьшаются). Кнопка раскрытия “ленты” при этом отображается как “неактивная”, а еще одна кнопка, предназначенная для свертывания “ленты”, наоборот, становится “активной”.

3) “Лента” содержит несколько миниатюр, каждая из которых соответствует одной из презентаций, — столько, сколько умещается по ширине окна браузера. Слева и справа в “ленте” размещаются кнопки “стрелка влево” и “стрелка вправо”, щелчки на которых мышью позволяют прокручивать “ленту” в соответствующем направлении для просмотра всех содержащихся в ней миниатюр.

4) Отыскав в “ленте” требуемую миниатюру, щелчком мыши можно открыть ее на просмотр в расположенной над “лентой” области просмотра. При этом “лента” автоматически скрывается (от нее опять остается узкая полоска внизу окна браузера), а выбранная презентация отображается практически во все окно браузера.

5) Раскрытую “ленту” можно свернуть в любой момент, щелкнув мышью на соответствующей кнопке. При этом в области просмотра остается та же самая презентация, которая была открыта до этого.

1. Создайте в папке **Итог** вложенную папку **Презентации**. Создайте в ней вложенные папки, имена которых соответствуют фамилиям учащихся. В каждой папке разместите презентацию соответствующего учащегося. Кроме того, поместите в каждую папку миниатюрное изображение первого слайда этой презентации размерами 200 × 150 пикселей, сохраненное в файле формата JPEG с именем, совпадающим с фамилией учащегося (и с именем соответствующей папки). Для этого можно:

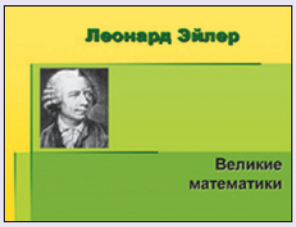
а) открыть первый слайд презентации в программе PowerPoint и сохранить его в файл **.jpg**, выбрав в меню пункт **Файл, Сохранить как**, а затем выбрав в окне сохранения файла в списке **Тип файла** пункт **Рисунок в формате JPEG (*.jpg)**;

б) начав показ презентации, при отображении первого слайда (если на нем использованы анимационные эффекты — при отображении всех его элементов) нажать клавишу <PrintScreen> и вставить помещенную в буфер копию экрана в любом графическом редакторе, а затем сохранить ее в графическом файле формата JPEG.

Примеры таких миниатюрных изображений:

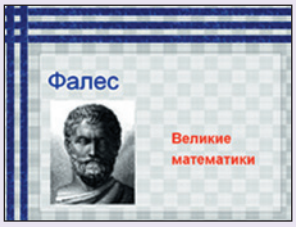
Блохина.jpg

Иванов.jpg



Разаренова.jpg

Шибилева.jpg



2. Создайте при помощи текстового редактора “Блокнот” в папке **Итог** файл **index.html** со следующим HTML-листингом. При этом в строках, выделенных подчеркиванием, введите список имеющихся в папке **Презентации** папок с именами-фамилиями учащихся и список имен файлов находящихся в них презентаций (ниже в листинге даны конкретные примеры).

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html" charset="windows-1251">
  <title> название проекта </title>
  <script language="JavaScript">
    <!--
    // при открытии страницы — открыть ее во весь экран
    self.moveTo(0,0)
    self.resizeTo(screen.availWidth,screen.availHeight)
      var h;       // исходная высота окна с презентацией
      var dh;      // изменение высоты окна с презентацией
      var ribbon_mini_h;  // высота ленты в свернутом состоянии
      var ribbon_maxi_h;  // высота ленты в раскрытом состоянии
      var lft;     // номер крайней миниатюры слева
    function inils() {  // начальные присваивания
      ribbon_mini_h=40;           // высота свернутой ленты
      ribbon_maxi_h=200;           // высота свернутой ленты
            // исходная высота окна браузера
      h=document.body.scrollHeight;
            // высота окна с презентацией при свернутой ленте (10 пикселей — запас)
      hslide_maxi=h-(ribbon_mini_h+10);
            // высота окна с презентацией при раскрытой ленте (10 пикселей — запас)
      hslide_mini=h-(ribbon_maxi_h+10);
      // массив имен миниатюр
      inames=new Array("Асланян", "Блохина", "Гордеенко", "Дубровин", "Иванов", "Кривошей", "Лапшин",
"Рагимов", "Разаренова", "Рыбалкин", "Силуянов", "Скаткова", "Слизкова", "Шибилева", "Ширяева") ;
      inom=inames.length;   // кол-во миниатюр
      // массив имен презентаций
        irefs=new Array("Асланян _ Алан Матисон Тьюринг.pps", "Блохина _ Альберт Эйнштейн.pps",
"Гордеенко _ Гаусс Карл Фридрих.pps", "Дубровин _ Ньютон Исаак.pps", "Иванов _ Леонард Эйлер.pps", "Кривошей
_ Михаил Васильевич Ломоносов.pps", "Лапшин _ Декарт Рене.pps", "Рагимов _ Архимед.pps", "Разаренова _ Фран-
суа Виет.pps", "Рыбалкин _ Пифагор Самосский.pps", "Силуянов _ Чебышёв Пафнутий Львович.pps", "Скаткова _
Клейн Феликс.pps", "Слизкова _ Софья Ковалевская.pps", "Шибилева _ Фалес.pps", "Ширяева _ Евклид.pps") ;
      lft=0;   // номер крайней миниатюры слева
               // (элементы массива нумеруются с нуля)
      ribbonGo();  // обновить содержимое ленты
      // сначала лента спрятана
      document.getElementById('main').height=hslide_maxi;
      document.all.ribbon.style.display="none";
    }
    function ribbonUp() {  // раскрыть ленту
      document.getElementById('main').height=hslide_mini;
      document.all.ribbon.style.display="";
      document.butdn.src="_imgs/button_dn.gif";
      document.butup.src="_imgs/button_up_no.gif";
    }
    function ribbonDn() {  // спрятать ленту
      document.getElementById('main').height=hslide_maxi;
      document.all.ribbon.style.display="none";
      document.butdn.src="_imgs/button_dn_no.gif";
      document.butup.src="_imgs/button_up.gif";
    }
    function ribbonRt() {  // прокрутить ленту вправо
      if (lft > 0) { lft=lft-1 }
      ribbonGo();  // обновить содержимое ленты
    }
    function ribbonLt() {  // прокрутить ленту влево
      if (lft < inom-4) { lft=lft+1 }
      ribbonGo();  // обновить содержимое ленты
    }
```

```
      function ribbonGo() {  // вывести ленту
        document.pic1.src="Презентации/"+inames[lft]+"/"+inames[lft]+".jpg";
        document.pic2.src="Презентации/"+inames[lft+1]+"/"+inames[lft+1]+".jpg";
        document.pic3.src="Презентации/"+inames[lft+2]+"/"+inames[lft+2]+".jpg";
        document.pic4.src="Презентации/"+inames[lft+3]+"/"+inames[lft+3]+".jpg";
        if (lft > 0) {
          document.butrt.src="_imgs/button_rt.gif"
        }
        else {
          document.butrt.src="_imgs/button_rt_no.gif"
        }
        if (lft < inom-4) {
          document.butlt.src="_imgs/button_lt.gif"
        }
        else {
          document.butlt.src="_imgs/button_lt_no.gif"
        }
      }
      function ribbonClick(i) {  // щелчок по миниатюре
       document.getElementById('main').src="Презентации/"+inames[lft+i]+"/"+irefs[lft+i];
        ribbonDn();  // скрыть ленту
      }
      //-->
    </script>
  </head>
  <body onLoad="inils();" bgcolor=#F5F5DC link=blue vlink=#000080 topmargin=0>
   <iframe src="start.htm" width="100%" height="550" align="center" id="main" frame-
border=0 scrolling=no></iframe>
    <center>
    <table width=100% border=0>  <!--  bgcolor=#f8f8b9 -->
      <tr>
        <td align=center><i>Презентации <b>" название проекта "</b></i></td>
        <td align=right width=50><img name="butdn" src="_imgs/button_dn_no.gif" bor-
der=0 alt="Свернуть ленту" onClick="ribbonDn();">   <img name="butup"
src="_imgs/button_up.gif" border=0 alt="Развернуть ленту" onClick="ribbonUp();"><td>
      </tr>
    </table>
    <table width=100% border=0 id="ribbon">
      <tr>
        <td width=50 align=left><img name="butlt" src="_imgs/button_lt.gif" border=0
alt="Прокрутить ленту влево" onClick="ribbonLt();"></td>
        <td width=50 align=center>
          <img name="pic1" border=1 width=200 height=150 alt="Посмотреть презентацию"
onClick="ribbonClick(0);">
        </td>
        <td width=50 align=center>
          <img name="pic2" border=1 width=200 height=150 alt="Посмотреть презентацию"
onClick="ribbonClick(1);">
        </td>
        <td width=50 align=center>
          <img name="pic3" border=1 width=200 height=150 alt="Посмотреть презентацию"
onClick="ribbonClick(2);">
        </td>
        <td width=50 align=center>
          <img name="pic4" border=1 width=200 height=150 alt="Посмотреть презентацию"
onClick="ribbonClick(3);">
        </td>
        <td width=50 align=right><img name="butrt" src="_imgs/button_rt.gif" border=0
                    alt="Прокрутить ленту вправо" onClick="ribbonRt();"></td>
      </tr>
```

```
    </table>
    </center>
</html>
```

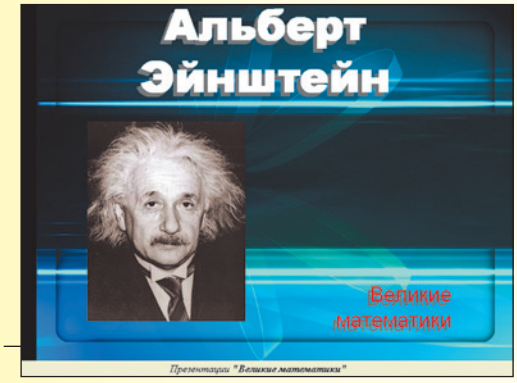
3. Создайте в папке **Итог** файл **start.htm**. Соответствующая web-страница будет загружаться в поле для просмотра презентаций при первом открытии пользовательской оболочки.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
    charset=windows-1251">
    <title> Название вашего проекта </title>
  </head>
  <body bgcolor= цвет фона >
    <center>
    <img src="_imgs\ картинка – "заставка" ">
    </center>
  </body>
</html>
```

4. Поместите в папку **Итог_imgs** графические файлы с изображениями кнопок раскрытия, свертывания и прокрутки “ленты” влево-вправо, “активные” и “неактивные” — например, такие:

	Свертывание “ленты”	Раскрытие “ленты”	Прокрутка влево	Прокрутка вправо
“Активные”				
	button_dn.gif	**button_up.gif**	**button_lt.gif**	**button_rt.gif**
“Неактивные”				
	button_dn_no.gif	**button_up_no.gif**	**button_lt_no.gif**	**button_rt_no.gif**

5. Откройте созданный файл **index.htm** двойным щелчком мыши и оцените достигнутый результат. Проверьте работу всех “механизмов” созданной “ленты”: ее раскрытие и свертывание с помощью соответствующих кнопок; прокрутку миниатюр презентаций влево и вправо (в том числе “до упора”, чтобы убедиться, что кнопки прокручивания при этом переходят в “неактивный” режим); выбор и просмотр презентаций. (При появлении окна запроса о том, что нужно сделать с pps-файлом, щелкайте в нем мышью на кнопке **Открыть**. В начале работы с оболочкой не забудьте “разблокировать” активное содержимое в окне браузера.)

Со свернутой “лентой”	*С развернутой “лентой”*
	

6. Под руководством учителя запишите подготовленное содержимое папки **Итог** на диск CD-R. Вставив его в накопитель CD-ROM, проверьте срабатывание автозапуска, работу интерактивной оболочки и презентаций. (Вместо записи на диск CD-R можно создать файл образа CD и воспользоваться программой-эмулятором CD/DVD-накопителя.)

Литература

Богомолова О.Б., Усенков Д.Ю. Искусство презентации: практикум. М.: БИНОМ. Лаборатория знаний, 2010.

Общероссийский проект

Школа цифрового века

Интернет-обеспечение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

Отметьте журналы, которые вы хотели бы получать

- ☐ «Английский язык»
- ☐ «Библиотека в школе»
- ☐ «Биология»
- ☐ «География»
- ☐ «Дошкольное образование»
- ☐ «Здоровье детей»
- ☐ «Информатика»
- ☐ «Искусство»
- ☐ «История»
- ☐ «Классное руководство и воспитание школьников»
- ☐ «Литература»
- ☐ «Математика»
- ☐ «Начальная школа»
- ☐ «Немецкий язык»
- ☐ «Основы безопасности жизнедеятельности»
- ☐ «Русский язык»
- ☐ «Спорт в школе»
- ☐ «Технология»
- ☐ «Управление школой»
- ☐ «Физика»
- ☐ «Французский язык»
- ☐ «Химия»
- ☐ «Школьный психолог»
- ☐ «Школа для родителей»

Все отмеченные журналы придут в ваш Личный кабинет в следующем месяце

Вступайте в «Школу цифрового века»!

Прием заявок на сайте

digital.1september.ru

Логика, творчество, интеллект

Об опыте проведения конкурса по математике и информатике

Предмет математики настолько серьезен, что полезно не упускать случая, делать его немного занимательным.

Блез Паскаль

Ел.А. Мирончик,
Ек.А. Мирончик,
Т.Н. Дутко,
г. Новокузнецк

В статье представлен сценарий проведения конкурса по математике и информатике среди учащихся 5–6-х классов. Городской конкурс, по итогам проведения которого была написана данная статья, проводился на базе одного образовательного учреждения и был рассчитан на десять команд. Вместе с тем материалы конкурса носят достаточно универсальный характер и могут быть использованы при проведении различных внеклассных мероприятий.

Для воспитания успешной личности необходимо развивать в ней такие качества, как логика, творчество и интеллект. Гармоничное сочетание трех этих слов может послужить залогом успеха в жизни, а пока они стали ключом к победе в конкурсе. На празднике математики и информатики мы сами постарались пользоваться ими, а также дали возможность проявить их и старшеклассникам, и конечно же командам-участницам. Именно поэтому мероприятие получило название “Логика. Творчество. Интеллект”.

Конкурс проводился в рамках традиционной в лицее декады математики и информатики. На этапе подготовки и во время проведения большая роль отводилась старшеклассникам. Была сделана попытка свести роль взрослых во время проведения конкурса к минимуму. Старшеклассники разрабатывали программы для некоторых этапов, работали над дизайном грамот и других печатных материалов, выступали в роли судей, тьюторов команд и фотографов. Работа нашлась всем желающим, даже тем, у кого математика не самый любимый предмет. Одной из воспитательных целей мероприятия

было развитие коммуникативных качеств через объединение детей разного возраста в одну команду.

К участию в конкурсе приглашались команды школ и лицеев города, состоящие из четырех человек, по два пятиклассника и два шестиклассника. Конкурс был разбит на несколько этапов, размещенных по разным кабинетам школы. Для организации перемещений гостей по незнакомому зданию к каждой команде был прикреплен взрослый ученик — тьютор. Но, кроме организационной функции, тьюторы играли роль помощников, иногда дающих наводящие вопросы и настраивающих на позитивное восприятие возможных неудач.

По плану проведения мероприятия сначала все участники собрались в актовом зале для приветствия, знакомства с тьютором и проведения первого испытания. Затем команды двигались по индивидуальным маршрутным листам. После прохождения всех этапов команды возвращались в зал для проведения заключительного испытания и подведения итогов. В то время как участники соревнований решали первое задание, учителя совершили экскурсию по всем этапам и собрались для обсуждения заданий и критериев оценивания в актовый зал.

В таблице приведена схема маршрутов в расчете на 10 команд. Маршруты составлены так, чтобы на каждом этапе, за исключением первого и последнего, одновременно находилось две команды, которые шли параллельно в ходе всего мероприятия (им соответствовал маршрут с одним номером). На каждое испытание отводилось по 10 минут, затраченное время не учитывалось при оценивании. В задачи тьюторов входил контроль времени, проведенного командой на этапе, чтобы избежать столпотворения.

В таблице приведена схема передвижения команд по этапам.

Далее опишем специфику, правила и методику оценивания каждого этапа. Позиционируя конкурс как межпредметный, будем описывать знания из математики и информатики, необходимые для выполнения задания.

Этап: Домашнее задание

Во время подготовки к конкурсу участникам необходимо было составить ребус, зашифровав название своей команды. Заметим, что слово должно иметь отношение к математике или информатике. Кроме этого, при выполнении домашнего задания участники использовали навыки работы в текстовом редакторе (работа с рисунками, фигурный текст), некоторым также помогли навыки создания рисунков в графическом редакторе или навыки поиска в сети Интернет.

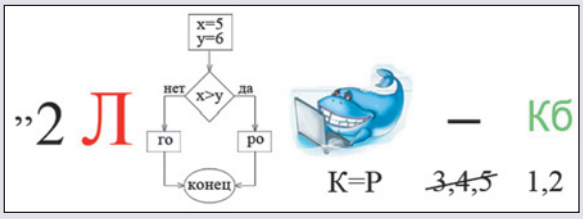
На первом этапе конкурса участникам было предложено в течение 10 минут разгадать названия всех команд-соперниц. Правильно разгаданный ребус оценивался в один балл.

Предлагаем вам некоторые ребусы. В процессе подготовки организаторы не вносили изменений в предоставленные ребусы.

Пятнашки

	Маршрут 1	Маршрут 2	Маршрут 3	Маршрут 4	Маршрут 5	Учителя, сопровождающие команды
1	Приветствие					
2	Домашнее задание					Экскурсия по этапам
3	Сломанный калькулятор	Пазлы	Робот	Встреча с мудрецом	В гостях у хозяина пещеры	Презентация
4	Робот	В гостях у хозяина пещеры	Пазлы	Сломанный калькулятор	Встреча с мудрецом	
5	В гостях у хозяина пещеры	Встреча с мудрецом	Сломанный калькулятор	Робот	Пазлы	
6	Встреча с мудрецом	Робот	В гостях у хозяина пещеры	Пазлы	Сломанный калькулятор	
7	Пазлы	Сломанный калькулятор	Встреча с мудрецом	В гостях у хозяина пещеры	Робот	
8	Математика вокруг нас					Анкетирование
9	Награждение					

Монитор



Алгоритмики

Квадрат

При подведении итогов ни одна команда не получила максимального количества баллов. Виной этому, на наш взгляд, слишком сложные ребусы, составленные командами.

Этап: Сломанный калькулятор

На этом этапе участникам предлагалось решить примеры, используя приложение “Калькулятор”. Сложность заключалась в том, что в калькуляторе были “сломаны” некоторые кнопки. Для реализации этого этапа учащийся 10-го класса Ткачев Александр в среде Delphi разработал приложение “BrokenCalc”, которое состоит из модуля ученика и модуля учителя. Модуль учителя позволяет отключать любые кнопки. Модуль ученика работает практически как стандартное приложение. Настройки активности кнопок модуль ученика берет из заданной учителем конфигурации. Кроме этого, калькулятор подсчитывает количество нажатых участником кнопок, что позволяет объективно и оперативно оценивать результат.

На конкурсе были отключены кнопки с цифрами 3 и 5 (см. *рис.* 1).

Таким образом, целью участников было не только верно решить примеры, но и сделать это при минимальном количестве нажатых кнопок.

Для этого этапа каждая команда была разделена на пары, состоящие из одного пятиклассника и одного шестиклассника. В зачет команды входил результат лучшей пары в каждом примере. Затем был составлен рейтинг команд, и, исходя из него, команды получили от 0 до 10 баллов.

Рис. 1. Модуль ученика

Представляем примеры, предложенные участникам.

1. $3 \cdot 1248 =$
2. $1612 \cdot 5 =$
3. $1812 \cdot 10 - 225 =$
4. $\frac{2}{5} \cdot 988 =$
5. $(638 + 1549) : 30 =$
6. $326 \cdot \frac{3}{5} + 988 =$
7. $433 \cdot 162 - 1945 =$

Рассмотрим возможные действия ученика при решении первого примера.

$$3 \cdot 1248 =$$

Вычисление этого примера можно осуществить решением примера 1248 + 1248 + 1248, в этом случае количество нажатых кнопок равно 14. Этот же результат можно достигнуть вычислением (1 + 2) · 1248, при этом результат нажатий равен 8. Так как данный калькулятор работает как знакомый детям “Обычный” калькулятор, а не “Инженерный”, то у него так же нет приоритета операций, а значит, нет необходимости расстановки скобок.

На этом этапе командам были необходимы навыки счета и элементарных арифметических преобразований. Кроме этого, участники продемонстрировали возможность работать в незнакомом приложении.

По проведенному анкетированию учителей можно сказать, что этот этап, на их взгляд, мог вызвать у участников наибольшие затруднения, тем не менее большинство команд справилось с поставленной задачей, однако не все выбрали оптимальное решение. По отзывам учеников из команды нашего лицея, этап был признан сложным, но самым интересным.

Разработанную программу можно взять на сайте *www.liceym111.ru*. Она может быть полезной не только в рамках конкурса, но и на уроках математики в среднем звене.

Этап: Робот

Оборудованием для выполнения этапа были программируемый робот LEGO MindStorm NXT 2.0 и рулетка. Участникам приходилось корректировать характеристики движения робота, чтобы научить его проезжать определенное расстояние, необходимое для доставки груза на модель строительной площадки. Чтобы получить максимальную оценку, детям необходимо было догадаться, как с помощью рулетки правильно выполнить измерение расстояния и вычислить верное значение переменной, изменяемой непосредственно на пульте управления роботом.

Модель робота была разработана учениками 9-го класса Удниковым Денисом и Тереховым Никитой. Конфигурация приведена на рисунке.

Капралов Степан, учащийся 11-го класса, разработал в среде BricxCommanderCenter программу, осуществляющую захват груза, движение по прямой с введенной характеристикой, влияющей на пройденное расстояние, сброс груза и движение обратно.

Некоторые участники сразу брались за рулетку, измеряя все, что требуется, и вычисляя правильный ответ. Но большинство, покрутив ее в руках, старались просто угадать ответ случайным подбором чисел.

За выполненное задание с проведением верных измерительных и вычислительных операций команда получала 10 баллов. Если подход к решению выбран верно, но допущены ошибки в измерениях — 8 баллов. За выполнение задания методом подбора верного значения — 4 балла. Если результат не был достигнут — 0 баллов.

При выполнении этого задания участники демонстрировали умение давать команды исполнителю в незнакомом формате, опознать, составить и решить задачу на движение, а также применить измерительные и вычислительные навыки.

Получилось так, что команды получили за этот этап либо 10, либо 4 балла. Все участники достигли необходимого результата, но разными способами: либо составлением задачи с последующим вычислением, либо подбором искомого коэффициента. Так как робот для многих детей был совершенно новым и необычным, было предусмотрено дополнительное время, которого в результате оказалось достаточно для подбора параметра. С другой стороны, этот этап ярко продемонстрировал проблему школьного образования — неумение применить имеющиеся знания на практике.

Этап: В гостях у хозяина пещеры

По замыслу организаторов это самый сложный этап, в то же время и самый театрализованный. Местом проведения был школьный музей природы, дорога к которому идет по длинному темному коридору. Главным действующим лицом был хозяин пещеры, роль которого исполнял ученик 10-го класса Фаустов Федор. По дороге к пещере вооруженный фонариком проводник Боков Никита рассказывал про голодного хозяина и неумелых слуг. Поэтому первым заданием было помочь слугам накормить хозяина пещеры, пожарив ему гренки.

Для этого участникам предлагалось решить классическую задачу на смекалку. Чтобы пожарить гренку с одной стороны, требуется одна минута. Необходимо пожарить три гренки как можно быстрее. Каждую гренку необходимо прожарить с двух сторон. На сковороде помещается ровно две гренки. Максимальное количество баллов (четыре балла) участники получали за результат в три минуты. Приготовление гренок за четыре минуты оценивалось в два балла.

Для этого конкурса был приготовлен реквизит: одна сковорода и три гренки на каждую команду. Слуги осуществляли отсчет времени.

На этом испытания не заканчивались. Сытый хозяин оказался рассеянным и согласен был выпустить гостей, но потерял свой ключ. Поэтому вторым заданием было решение логической задачи. Участникам были предложены конверты, на которых были написаны высказывания, по которым необходимо было определить, где лежит ключ. Сложность состояла в том, что только на одном из трех конвертов было истинное высказывание. Приведем пример одного из вариантов. Другие варианты этого задания можно посмотреть в статье Е.А. Мирончик в журнале “Информатика в школе” № 5 (69), 2011, с. 45.

Надпись на голубом конверте	Надпись на розовом конверте	Надпись на желтом конверте
Ключ в розовом конверте	Ключ в голубом или желтом конверте	Ключ в голубом или розовом конверте

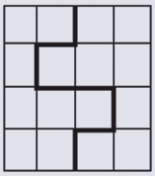
Кроме ответа, в каком конверте находится ключ, требовалось указать конверт с истинным высказыванием. Если команда правильно отвечала на оба вопроса, она получала 6 баллов, на один из вопросов — 3 балла, иначе 0 баллов. Данный этап, с точки зрения организаторов, самый сложный во всем конкурсе, так как детям необходимо было решать задачу на смекалку и серьезную логическую задачу. Максимальное количество баллов за весь этап также равно 10. Малое количество баллов команды получили потому, что нет навыков решения логических задач.

Этап: Встреча с мудрецом

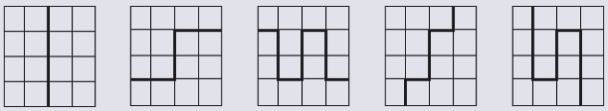
На этом этапе участники подвергались двум испытаниям на пространственное воображение, которые предлагал мудрец.

В первой части этапа участникам было предложено из магнитного набора, которым оборудованы кабинеты математики, собрать указанное количество треугольников. Детали для сборки к каждому заданию лежали в отдельной секретной коробке.

Вторым заданием на этом этапе участникам предлагалось разрезать квадрат размером 4 × 4 клетки на две равные части, причем резать разрешается строго по клеткам. Детям был предложен один вариант решения этой задачи, изображенный на рисунке. Необходимо было найти еще пять способов. Оборудование для этапа — картонные заготовки квадрата и ножницы.



Варианты решения задачи приведены ниже. За каждый вариант команда получала по одному баллу.



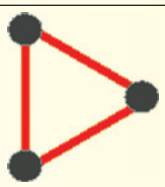
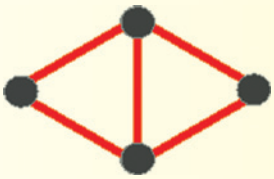
Итого максимальный результат на этом этапе (10 баллов) получала та команда, которая успешно справилась со всеми секретными коробками и нашла пять способов разрезания квадрата. В целом все команды хорошо справились с заданием и получили высокий балл.

Этап: Пазлы

На этом этапе участникам были предоставлены карточки с цифрами, скобками и знаками (их количество приведено ниже), из которых необходимо было получить верное числовое равенство, включив в него наибольшее количество карточек.

По одной карточке	+ − : · = 6 7 8 9 0
По две карточки	1 2 3 4 5 ( )

Самым простым решением было составить пример из пяти карточек. Например, 1 + 3 = 4. Рассуждая далее, участник мог заменить 3 на частное 9 : 3, поменяв равенство на 1 + 9 : 3 = 4. При этом количество карточек увеличивалось на 2, затем, анало-

Содержимое секретной коробки	Задание	Ответ	Комментарий
Три палочки, три шарика	Собрать один треугольник		Данное задание рассчитано на ознакомление участника с механизмом сбора конструктора. За верное решение команда получала 1 балл
Пять палочек, четыре шарика	Собрать два треугольника		Второе задание также оценивалось в 1 балл
Шесть палочек, четыре шарика	Собрать четыре треугольника		Третье задание было самым сложным, так как требовало выхода из плоскости в пространство, поэтому было оценено в 3 балла

гично заменив 4 на разность 12 – 8. Таким образом, равенство будет выглядеть следующим образом: 1 + 9 : 3 = 12 – 8. А количество используемых карточек станет равным 10, что уже заметно больше изначальных 5. Кроме этих действий, можно было добавить скобки там, где они, по сути, не нужны, но и не меняют результат. Данный этап выявлял вычислительные навыки участников и умения совершать арифметические преобразования.

Неверно составленное равенство оценивалось в 0 баллов. За верное равенство из пяти карточек команда получала 1 балл. За каждую карточку, добавленную к пяти, результат команды был увеличен на 1 балл. После прохождения конкурса судьи на данном этапе составляли рейтинг; исходя из места в нем, был выставлен результат по десятибалльной шкале.

Обращаем ваше внимание, что при проведении этого этапа среди карточек не должно быть более одной карточки с цифрой 0. Это позволит участникам составить равенство из всего набора, умножив правую часть на 0 и приравняв ему же.

Этап: Математика вокруг нас

По мнению участников конкурса, этот этап оказался для них самым сложным, потому что необходимо было увидеть мир глазами математика. В привычных, обыденных вещах для большинства детей оказалось проблематично разглядеть математические термины. В течение десяти минут было предложено найти и показать максимальное количество математических терминов на фотографиях, сделанных учениками лицея и организаторами в рамках подготовки к конкурсу. Тьюторы подсчитывали общее количество терминов, названных и верно показанных участниками. Но сложным этот этап оказался и для судейства, так как среди терминов, названных участниками, были незнакомые для тьюторов. На часах шестиклассник увидел квантовый модуль.

Приведем некоторые варианты фотографий, а также примеры терминов, которые могли “найти” участники. Результаты также были оценены по десятибалльной шкале.

По отзывам участников, это был интересный этап еще потому, что на фотографиях были здания и обстановка (школа, двор, парк), которые дети видят каждый день. Поэтому, подготавливая фотографии к данному этапу, рекомендуем сфотографировать узнаваемые детьми места.

Итоги конкурса

В I Открытом городском конкурсе “Логика. Творчество. Интеллект” приняло участие 10 команд из школ и лицеев города. По итогам конкурса было выявлено три команды-победительницы, которые получили грамоту за победу. Остальные команды получили грамоты за участие. Дизайн наградных материалов был разработан ученицей 10-го класса Ильиной Екатериной. Учитывая психологические особенности школьников среднего звена, конкурс выявлял победителя, не выявляя проигравших. Это стало возможным благодаря составленным заданиям и работе тьюторов, которые были ориентированы на помощь своей команде наводящими вопросами и эмоциональной поддержкой. Таким образом, не было команд, получивших все самые низкие оценки. Все дети ушли с конкурса довольные своим выступлением, сказав, что конкурс был интересным и увлекательным. Дети получили удовольствие от участия в нем. Также очень полезной оказалась подготовка конкурса со старшими школьниками. Они нашли практическое применение своим умениям в области программирования, техники, дизайна и организационным навыкам.

Треугольник, Циркуль, Транспортир, Прямоугольник, Равенство

Диаметр, Число, Цифра, Угол

Прямая, Дуга, Угол, Параллельные

Шестиугольник, Ломаная, Многоугольник

ШКОЛА ПРОГРАММИРОВАНИЯ

Когда человек хочет передвинуть гору,
он начинает с того,
что убирает маленькие камни.

Об использовании процедур и функций

Д.М. Златопольский,
Москва

Как, очевидно, известно читателям, в программах используются процедуры и функции. Что же это такое? Об обоих[1] этих понятиях можно сказать, что это — фрагмент программы, в котором решается какая-то часть всей задачи. Этот фрагмент оформляется особым образом и снабжается именем. Возникает вопрос: “Зачем нужно какую-то часть программы оформлять отдельно?”. Чтобы ответить на него, рассмотрим несколько примеров.

В программах часто встречаются повторяющиеся или похожие фрагменты. Например, для того чтобы вывести на экран “заставку” с изображением, представленным на *рис.* 1,

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        ЭТУ ПРОГРАММУ РАЗРАБОТАЛ ПЕТЯ ХАКЕРОВ
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

Рис. 1

— в программе на школьном алгоритмическом языке (система КуМир) должно быть записано:

```
нц для i от 1 до 40
  вывод "*"
кц
вывод нс
нц для i от 1 до 5
  вывод " "
кц
вывод "ЭТУ ПРОГРАММУ РАЗРАБОТАЛ
       ПЕТЯ ХАКЕРОВ", нс
нц для i от 1 до 40
  вывод "*"
кц
```

Видно, что в программе есть два абсолютно одинаковых фрагмента, относящихся к выводу линий из звездочек. Можно эти фрагменты оформить как процедуру.

На школьном алгоритмическом языке общий вид процедуры следующий[2]:

```
алг <имя_процедуры>
нач          <описания_переменных_величин_
используемых_в_процедуре>
  <тело_процедуры>
кон
```

— где <описания_переменных_величин_используемых_в_процедуре> — перечень используемых в процедуре величин с указанием их типа;

— <тело_процедуры> — операторы, реализующие решаемую с помощью процедуры задачу.

В нашей задаче процедура, с помощью которой можно получить линию из 40 звездочек, имеет вид:

```
алг Линия
нач цел i
  нц для i от 1 до 40
    вывод "*"
  кц
кон
```

Созданную процедуру можно использовать в основной части программы. Вызов процедуры на выполнение осуществляется отдельным оператором, в котором указывается имя процедуры:

```
Линия
вывод нс
нц для i от 1 до 5
  вывод " "
кц
вывод "ЭТУ ПРОГРАММУ РАЗРАБОТАЛ
       МИТЯ ХАКЕРОВ", нс
Линия
...
```

Нетрудно убедиться, что в данном случае применение процедуры уменьшает размер программы (это преимущество проявилось бы в еще большей степени, если бы в задаче пришлось рисовать линии три и более раз).

[1] В языке программирования Си оба эти понятия называются общим термином “функция”.

[2] С правилами оформления процедур в языке программирования, который вы изучаете, ознакомьтесь самостоятельно.

Еще один пример — пусть в значительной степени условный, но достаточно показательный. Если необходимо на экране нарисовать из линий изображение слова “МУ”, то соответствующая программа для исполнителя Чертежник оформляется так:

```
алг МУ
нач
  …
  сместиться на вектор(0, 100)
  сместиться на вектор(50, –50)
  сместиться на вектор(50, 50)
  сместиться на вектор(0, –100)
  поднять перо
  сместиться на вектор(10, 0)
  опустить перо
  сместиться на вектор(50, 0)
  сместиться на вектор(0, 100)
  …
кон
```

Легко ли в такой программе найти ошибочный оператор, из-за которого, например, буква “У” изображена неправильно? Где нужно добавить операторы, чтобы получить на экране вместо “МУ” слово “МЯУ”?

Если же рисование букв “М” и “У” оформить в виде отдельных процедур, то основная часть программы примет вид:

```
  алг МУ
нач
  сместиться в точку(50, 100)
  М
  сместиться в точку(160, 100)
  У
кон
```

Конечно, при этом размер всей программы увеличится, но зато проявятся другие преимущества — программа станет более понятной, в ней легко найти нужное место, в том числе и связанное с ошибками. Если же нужно изобразить на экране слово “МУМУ”, то тогда применение процедур, кроме того, приведет и к уменьшению размера программы.

Возможность использования процедур и функций позволяет применять современные методы проектирования программ. Дело в том, что при программировании достаточно сложной задачи решить ее “одним махом”, т.е. написать последовательно всю программу от начала до конца, обычно невозможно. Процесс разработки программ — это творческий процесс, проходящий в несколько этапов. Вначале стараются разработать наиболее общую схему алгоритма, не останавливаясь на технических деталях его реализации. В результате такой алгоритм представляется в виде последовательности относительно крупных блоков, реализующих более или менее самостоятельные смысловые части алгоритма, в которых решаются какие-то частные задачи. Эти блоки, в свою очередь, могут разбиваться на меньшие подблоки и т.д. Процесс последовательного структурирования программы продолжается до тех пор, пока реализуемые блоками алгоритмы не станут простыми и легкопрограммируемыми.

Рассказав о преимуществах применения процедур, пойдем дальше.

Возможно также использование так называемых “процедур с параметром” — процедур, результат выполнения которых зависит от некоторых величин — их параметров. Пусть, например, в программе требуется рисовать линии из звездочек разной “длины”:

```
…
нц для i от 1 до 40
  вывод "*"
кц
…
нц для i от 1 до 20
  вывод "*"
кц
…
нц для i от 1 до 30
  вывод "*"
кц
```

Можно, конечно, для такой задачи создать три отдельные процедуры, решающие эти задачи. Но выиграем ли мы при этом в размере программы? Нет, конечно. Желательно разработать процедуру, которая “умеет” рисовать линии из звездочек любой длины. Это и будет упомянутая чуть выше процедура с параметром. Общий вид таких процедур:

```
алг <имя_процедуры> (арг <список_
     формальных_параметров>)
нач <описания_переменных_величин_
     используемых_в_процедуре>
  <тело_процедуры>
кон
```

— где <список_формальных_параметров> — перечень величин, от которых зависит результат выполнения процедуры (с указанием их типа); эти величины называют формальными параметрами процедуры. Формальные параметры используются в теле процедуры.

В нашем случае процедура, с помощью которой можно получить линию “любой” длины, имеет вид:

```
алг Линия(арг цел k)
нач цел i
  нц для i от 1 до k
    вывод "*"
  кц
кон
```

Результат ее выполнения зависит от величины *k*, которая и является единственным параметром процедуры.

Вызов такой процедуры осуществляется так:

<имя_процедуры> (<список_фактических_параметров>)

— где <список_фактических_параметров> — перечень конкретных значений параметров процедуры для решения конкретной задачи. Так, в нашем случае, для того чтобы использовать процедуру для рисования линии из 30 звездочек, вызов процедуры должен быть оформлен следующим образом:

```
Линия(30)
```

В качестве значений фактических параметров можно указывать[3]:

1) константы (заданные известные значения) — именно так было сделано в приведенном примере;

[3] Приведенный перечень будет далее уточнен.

2) имена величин, например, `Линия(длина1)`; как видно из примера, имена фактических и формальных параметров не обязательно должны совпадать;

3) выражения: `Линия(длина1 + 10)`.

В каждом из этих трех случаев тип фактического параметра должен совпадать с типом формального параметра, указанным в заголовке процедуры.

Ясно, что в процедуре могут быть использованы два и более формальных параметра. В таких случаях при оформлении списка фактических параметров следует учитывать следующие требования:

1) количество фактических параметров должно быть таким же, как и в списке формальных параметров в описании процедуры;

2) тип каждого фактического параметра должен совпадать с типом соответствующего формального параметра;

3) соответствующие фактические и формальные параметры должны совпадать по смыслу.

Если первые два требования ясны и понятны, то третье требует комментариев. Пусть, например, подготовлена процедура, изображающая на экране прямоугольник:

```
алг Линия(арг цел a, b)
нач
  …
кон
```

Как с ее помощью нарисовать прямоугольник шириной 20 и высотой 100? Какой из вариантов оформления вызова процедуры является правильным: `Линия(20, 100)` или `Линия(100, 20)`? Ответ зависит от того, какому размеру соответствует формальный параметр *a*, а какому — *b*. Если *a* — это высота прямоугольника, то правильный вариант — `Линия(100, 20)`, в противном случае первой должна быть указана ширина изображаемого прямоугольника, а второй — высота: `Линия(20, 100)`.

Рассмотрим далее очень показательный пример. Пусть в программе необходимо часто обменивать значениями две переменные величины. Для решения такой задачи составим[4] процедуру `Обмен`:

```
алг Обмен(арг цел a, b)
нач цел c
  c := a
  a := b
  b := c
кон
```

Используем эту процедуру для обмена значениями двух величин — *x* и *y*:

```
  алг Основная_программа
нач цел x, y
  |Исходные значения
  x := 10
  y := 100
  |Вызываем процедуру Обмен
  Обмен(x, y)
  |Выводим на экран новые значения
  вывод нс, "x=", x, " y=", y
кон
```

[4] В языке программирования Бейсик для обмена значениями двух величин имеется стандартная процедура SWAP.

Выполнив программу, обнаружим, что значения переменных не изменились! Почему — ведь вызывалась и работала процедура `Обмен`, в которой происходили изменения значений параметров? А все дело в том, что в школьном алгоритмическом языке, в языке Паскаль и ряде других возможны два вида формальных параметров:

1) формальные параметры-значения;

2) формальные параметры-переменные.

Особенностью первых является то, что все действия над ними внутри процедуры никак не отражаются на соответствующих фактических параметрах, т.е. какими фактические параметры были до вызова процедуры, такими они и останутся после завершения ее работы. Именно поэтому значения фактических параметров *x* и *y* не изменились — они были описаны как параметры-значения (со служебным словом **арг**). Чтобы в результате выполнения процедуры значения фактических параметров в основной части программы изменились согласно проведенным в процедуре действиям, формальные параметры должны быть описаны как параметры-переменные. В школьном алгоритмическом языке это делается путем сочетания служебных слов **арг** и **рез** в заголовке процедуры:

```
  алг Обмен(арг рез цел a, b)
…
```

в языке Паскаль — указанием в заголовке процедуры служебного слова **Var**:

```
Procedure Swap(Var a, b: integer);
…
```

Конечно, в описании процедуры можно сочетать и оба вида формальных параметров.

Внимание! При использовании формальных параметров-переменных в списке фактических параметров им может соответствовать только имя величины (константы и выражения в этом случае записывать нельзя).

Наглядной иллюстрацией механизма передачи значений в процедуру с формальными параметрами-переменными и из нее является следующий пример, в котором используется созданная ранее процедура `Обмен`:

```
алг Обмен_местами_двух_элементов_массива
нач цел i, цел таб m[1:10]
  нц для i от 1 до 10
    m[i] := rnd(100)
    вывод m[i], " "
  кц
  Обмен(m[1], m[10])
  нц для i от 1 до n
    вывод m[i], " "
  кц
кон
```

— т.е. в процедуре `Обмен` происходит обмен значениями двух ее параметров, которые являются значениями элементов массива, а в результате изменится и сам массив.

Теперь об особенностях использования процедур в языке программирования Бейсик. В нем все формальные параметры процедур, как говорится по умолчанию, считаются параметрами-переменны-

ми. При этом использование в качестве фактических параметров констант и выражений ошибкой не считается (результат выполнения процедур при таком оформлении их вызова определите самостоятельно).

Еще один пример, наглядно демонстрирующий особенности применения процедур с параметрами-переменными, приведен во второй статье рубрики "Школа программирования".

Прежде чем идти дальше, заметим, что переменные величины, описанные в процедуре, называют "локальными", а описанные в основной части программы — "глобальными". Локальные переменные "живут" только во время выполнения процедуры и поэтому недоступны в основной части программы и в других процедурах (когда вызывавшаяся процедура уже завершена). Например, при выполнении следующих четырех программ появится сообщение об ошибке, связанное с этим обстоятельством:

1

```
алг Основная_программа[5]
нач
  вывод a
кон
алг Процедура1
нач цел a
  a := 100
кон
```

2

```
алг Основная_программа
нач
  Процедура1
  вывод a
кон
алг Процедура1
нач цел a
  a := 100
кон
```

3

```
алг Основная_программа
нач
  Процедура1
  Процедура2
кон
алг Процедура1
нач цел a
  вывод a
кон
алг Процедура2
нач цел a
  a := 100
кон
```

4

```
алг Основная_программа
нач
  Процедура2
  Процедура1
кон
алг Процедура2
нач цел a
  a := 100
кон
алг Процедура1
нач цел a
  вывод a
кон
```

В то же время глобальные переменные доступны ("видны") в процедурах:

```
алг Основная_программа
нач цел a
  a := 100
  Процедура1
кон
алг Процедура1
нач
  вывод a
кон
```

Еще один случай, который следует рассмотреть, — совпадение имен локальной и глобальной переменных. Как, по-вашему, что будет выведено на экран в результате выполнения следующей программы:

```
алг Основная_программа
нач цел a
  a := 100
  Процедура1
кон
алг Процедура1
нач цел a
  a := 1
  вывод a
кон
```

Правильный ответ — 1 (говорят, что в этом случае локальная переменная "закрывает" собой глобальную).

Следует также отметить интересный (и полезный!) факт. В программе на языке Паскаль при отсутствии в последней процедуре оператора `a := 1` результат выполнения программы будет непредсказуемым — на экран может быть выведено любое число. Дело в том, что начальное присваивание *локальным переменным* процедур и функций нулевого значения не производится!

Теперь о том, что такое функция. Вы, конечно, знаете о так называемых "стандартных" (имеющихся в языке программирования) функциях (*sin*, *sqr* и др.). Что они делают при их использовании в программах? — Возвращают какое-то одиночное значение (результат расчетов или т.п.). Так вот, с такой же целью можно также создавать и использовать собственные функции. Пусть, например, в программе надо рассчитать значения x:

$$x = \frac{2+\sqrt{2}}{5+\sqrt{5}} + \frac{5+\sqrt{5}}{13+\sqrt{13}} + \frac{11+\sqrt{11}}{8+\sqrt{8}}$$

Видно, что в выражении имеются похожие фрагменты. Желательно для расчета дробей оформить функцию, которая вычисляла бы значения отдельных дробей вида $\frac{a+\sqrt{a}}{b+\sqrt{b}}$ при любых значениях a и b, а затем использовать ее для расчетов.

[5] В школьном алгоритмическом языке основная часть программы размещается раньше всех вспомогательных процедур.

Правила оформления функций в разных языках программирования различаются. В школьном алгоритмическом языке общий вид функций такой:

```
алг <тип_результата> <имя_функции> (арг <список_формальных_параметров>)
нач          <описания_переменных_величин_используемых_в_функции>
  <тело_функции>
кон
```

— причем последним оператором тела функции должен быть оператор присваивания, в левой части которого должно быть указано служебное слово знач. Значение в правой части оператора и будет значением функции.

Нетрудно увидеть, что в этом языке отличие функции от процедуры заключается в наличии в заголовке функции типа результата, который она возвращает. В других языках программирования, как правило, в описании функции указывается служебное слово "Function".

В нашем случае функция для расчета значений дробей указанного чуть выше вида оформляется так:

```
алг вещ Дробь(арг цел a, b)
нач цел числитель, знаменатель
  числитель := a + sqrt(a)
  знаменатель := b + sqrt(b)
  |Значение функции:
  знач := числитель/знаменатель
кон
```

Как и стандартные, функция, созданная программистом, используется в правой части оператора присваивания. Для этого надо указать ее имя, а в скобках — значения фактических параметров:

```
x := Дробь(2, 5) + Дробь(5, 13) +
+ Дробь(13, 8)
```

Требования к оформлению списка фактических параметров те же, что и для процедур с формальными параметрами-значениями.

Результат, возвращаемый функцией, может быть любого типа, в том числе логического[6].

Например, функция, определяющая, является ли натуральное число *a* делителем натурального числа *b*, имеет вид:

```
алг лог Делитель(арг цел a, b)
нач
  |Значение функции:
  знач := mod(b, a) = 0
кон
```

Созданная функция может быть использована в программе решения задачи: "Дано натуральное число *a*. Подсчитать количество делителей этого числа".

```
  алг Количество_делителей
нач цел a, возможный_делитель, количество
  вывод нс, "Задайте число a"
  ввод a
  количество := 0
  нц для возможный_делитель от 1 до div(a, 2)
    если Делитель(возможный_делитель, a)
      то
        количество := количество + 1
    все
  кц
  вывод нс, "Количество делителей числа a "
  вывод "равно ", количество
кон
```

Возникают вопросы: "Какая существует связь между процедурами и функциями? Можно ли вместо одной из них использовать другую?". Чтобы ответить на эти вопросы, надо вспомнить процедуры с формальными параметрами-переменными. Они, как и функции, возвращают в вызывающую их часть программы какие-то значения. Функция же всегда возвращает единственное значение. Это означает, что вместо функции можно использовать соответствующим образом оформленную процедуру. О том, целесообразно ли это делать, можно судить, например, по следующей процедуре, созданной для расчета значения дроби в рассмотренной выше задаче расчета значения *x*:

```
алг Дробь(арг цел a, b, арг рез вещ частное)
нач цел числитель, знаменатель
  числитель := a + sqrt(a)
  знаменатель := b + sqrt(b)
  частное := числитель/знаменатель
кон
```

Использование процедуры Дробь:

```
алг Основная_программа
нач вещ слагаемое1, слагаемое2,
        слагаемое3, x
  слагаемое1 := 0
  |Вызываем процедуру Дробь
  |для расчета первого слагаемого
  Дробь(2, 5, слагаемое1)
  слагаемое2 := 0
  |Вызываем процедуру Дробь
  |для расчета второго слагаемого
  Дробь(5, 13, слагаемое2)
  слагаемое3 := 0
  |Вызываем процедуру Дробь
  |для расчета третьего слагаемого
  Дробь(11, 8, слагаемое3)
  |Рассчитываем значение x
  x := слагаемое1 + слагаемое2 + слагаемое3
  …
кон
```

Вместо процедуры, возвращающей один результат, также можно (и даже, как следует из приведенного примера, как правило, целесообразнее) оформить функцию. Ясно, что когда число возвращаемых процедурой значений больше одного, применить функцию нельзя.

В заключение заметим, что процедура или функция может в процессе своей работы использовать любую другую процедуру (функцию) или саму себя. В последнем случае процедура (функция) называется *рекурсивной*. Рекурсивным процедурам и функциям будет посвящена статья в одном из следующих выпусков.

Задания для самостоятельной работы

1. Составьте процедуру, выводящую по периметру экрана прямоугольник из "звездочек" (*).

2. Подготовьте процедуру, выводящую на экран в строке любое количество некоторого символа.

[6] Если, конечно, такой тип предусмотрен в языке программирования.

3. Составьте процедуру, осуществляющую обмен значениями трех переменных величин a, b, c по следующей схеме: b присвоить значение a; c присвоить значение b; a присвоить значение c.

4. Даны стороны двух треугольников. Разработайте программу для нахождения суммы их периметров и сумму их площадей. (Определите процедуру для расчета периметра и площади треугольника по его сторонам.)

5. Разработайте программу для расчета значения x, определив и использовав функцию:

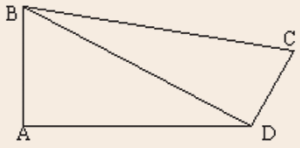
$$x = \frac{\sqrt{6}+6}{2} + \frac{\sqrt{13}+13}{2} + \frac{\sqrt{21}+21}{2}$$

6. Разработайте программу для определения значения $z = \operatorname{sign} x + \operatorname{sign} y$, где

$$\operatorname{sign} a = \begin{cases} -1 & \text{при } a < 0, \\ 0 & \text{при } a = 0, \\ 1 & \text{при } a > 0. \end{cases}$$

Значения x и y вводятся с клавиатуры. При решении задачи определите и используйте функцию *sign*.

7. Разработайте программу для нахождения периметра фигуры ABCD по заданным сторонам AB, AD и CD. (Определите функцию для расчета гипотенузы прямоугольного треугольника по его катетам.)



$\angle BAD = 90°$, $\angle BDC = 90°$

Рис. 2

8. Разработайте программу для расчета периметра треугольника, заданного координатами своих вершин. (Определите функцию для расчета длины отрезка по координатам его вершин.)

9. Получите с помощью программы все шестизначные счастливые номера. Счастливым называют такое шестизначное число, что сумма его первых трех цифр равна сумме его последних трех цифр. (Определите функцию для расчета суммы цифр трехзначного числа.)

10. Даны два предложения. В каком из них доля (в %) буквы “б” больше? (Определите функцию для расчета доли некоторой буквы в предложении.)

Программы (можно не все) присылайте в редакцию.

Дележ яблок

Обсудим программу решения такой задачи [1]: “Аня нарвала яблок и поровну раздала своим сестрам — Оле, Маше и Тане, а то, что осталось, съела. Оля свои яблоки поделила между тремя сестрами, а что осталось, съела. То же самое сделали Маша и Таня. Сколько яблок оказалось у каждой сестры, если число яблок, которое нарвала Аня, известно?”.

Можно, конечно, в программе непосредственно описать действия каждой из сестер:

```
алг Дележ
нач цел У_Ани, У_Оли, У_Маши, У_Тани
  |Ввод исходного числа яблок
  вывод нс, "Сколько яблок нарвала Аня? "
  ввод У_Ани
  У_Оли := 0 |Было у Оли
  |Ей Аня отдала треть
  У_Оли := div(У_Ани, 3)
  У_Маши := 0 |Было у Маши
  |Ей Аня отдала треть
  У_Маши := div(У_Ани, 3)
  У_Тани := 0 |Было у Тани
  |Ей Аня отдала треть
  У_Тани := div(У_Ани, 3)
  У_Ани := 0 |То, что осталось, Аня съела
  |После этого стала делить Оля
  У_Ани := div(У_Оли, 3)
  У_Маши := У_Маши + div(У_Оли, 3)
  У_Тани := У_Тани + div(У_Оли, 3)
  У_Оли := 0
  |Затем стали делить Маша
  У_Ани := У_Ани + div(У_Маши, 3)
  У_Оли := У_Оли + div(У_Маши, 3)
  У_Тани := У_Тани + div(У_Маши, 3)
  У_Маши := 0
  |и Таня
  У_Ани := У_Ани + div(У_Тани, 3)
  У_Оли := У_Оли + div(У_Тани, 3)
  У_Маши := У_Маши + div(У_Тани, 3)
  У_Тани := 0
  |Вывод ответа
  вывод нс, "После дележа у сестер
            будет яблок "
  вывод нс, "У Ани: ", У_Ани
  вывод нс, "У Оли: ", У_Оли
  вывод нс, "У Тани: ", У_Тани
  вывод нс, "У Маши: ", У_Маши
кон
```

— где *div* — функция, возвращающая целую часть частного от деления своего первого аргумента на второй (в других языках программирования для этого используется не функция, а специальная операция).

Программа получилась достаточно объемная и трудночитаемая (несмотря на наличие комментариев). Попробуем сократить ее. Видно, что в программе можно выделить четыре фрагмента, аналогичных по существу выполняемых действий, связанных с дележом яблок каждой из девочек. Если количество яблок “у отдающей их” обозначить *У_отдающей*, а у любых трех “берущих” — *У_берущей*1, *У_берущей*2 и *У_берущей*3, то изменения значений этих четырех величин можно оформить в виде отдельной процедуры:

```
алг Дележ(арг рез цел У_Отдающей,
          У_Берущей1, У_Берущей2,
          У_Берущей3)
нач
  У_Берущей1 := У_Берущей1 +
                div(У_Отдающей, 3)
  У_Берущей2 := У_Берущей2 +
                div(У_Отдающей, 3)
  У_Берущей3 := У_Берущей3 +
                div(У_Отдающей, 3)
  У_Отдающей := 0
кон
```

Обратим внимание на то, что все параметры процедуры Дележ описаны как параметры-пере-

менные (см. статью “Об использовании процедур и функций” в этом выпуске). Причина этого понятна — значения параметров в ходе выполнения процедуры меняются.

С применением созданной процедуры основная часть программы будет такой:

```
алг Дележ_2
нач цел У_Ани, У_Оли, У_Маши, У_Тани
  вывод нс, "Сколько яблок нарвала Аня? "
  ввод У_Ани
  |Начальное число яблок
  |у других девочек
  Оля := 0
  Маша := 0
  Таня := 0
  |4 этапа дележа:
  Дележ(У_Ани, У_Оли, У_Маши, У_Тани)
  Дележ(У_Оли, У_Маши, У_Тани, У_Ани)
  Дележ(У_Маши, У_Тани, У_Ани, У_Оли)
  Дележ(У_Тани, У_Ани, У_Оли, У_Маши)
  |Вывод ответа
  …
кон
```

Задания для самостоятельной работы

1. Ответьте, пожалуйста, на вопрос: “Какие преимущества, по вашему мнению, дает использование процедуры Дележ?”.

2. Разработав программу (на языке программирования, который вы изучаете), определите искомые значения при числе яблок, которые нарвала Аня, равном:

1) 45;

2) 125 (допустим, что это возможно).

3. Определите, при каком минимальном исходном количестве яблок:

1) задача имеет решение;

2) после всех дележей оставшееся число яблок будет равно исходному (если после дележей девочки не будут иметь возможности есть оставшиеся яблоки ☺);

3) все четыре девочки в ходе дележей будут съедать яблоки.

Литература

1. *Дагене В.А., Григас Г.К., Аугутис К.Ф.* 100 задач по программированию. М.: Просвещение, 1993.

“ЛОМАЕМ” ГОЛОВУ

Крест-накрест

Переставив буквы в строках приведенного ниже квадрата, получите осмысленные слова, при этом в диагоналях квадрата соберутся еще два слова, связанные с информатикой и математикой. Найдите все слова и дайте комментарии к ним.

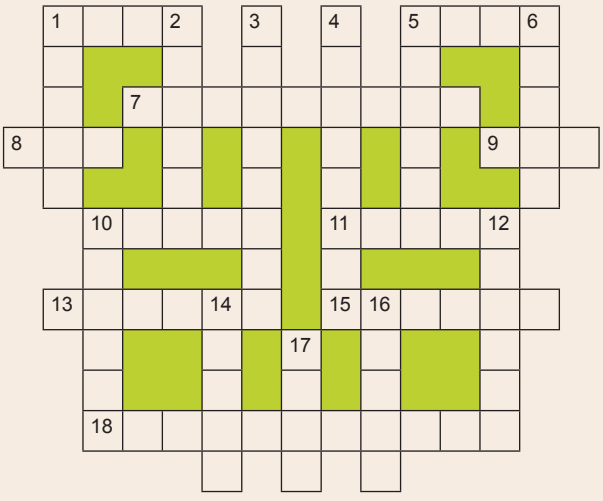
Р	Е	П	Е	С	О	Н
П	У	З	А	Р	А	К
Л	Е	Н	Е	Д	И	Е
Т	Р	И	О	М	О	Н
Б	А	С	М	А	Ш	Т
Р	О	Т	М	И	Р	А
К	А	Ш	А	В	И	Л

Цветные шарики

Имеются красный, синий, зеленый и черный шарики, среди которых могут быть волшебные. Специальное устройство — детектор позволяет определить, сколько из помещенных в него шариков волшебных. Как узнать, какие шарики волшебные, а какие — нет, всего за три измерения? Естественно, что должны рассматриваться “худшие” варианты, а не варианты типа “а вдруг повезет, и мы сразу все узнаем” ☺.

Кроссворд

Решите, пожалуйста, кроссворд.



По горизонтали

1. Структура данных, в которых применен принцип “последним пришел — первым вышел”.

5. Знак арифметической операции.

7. Алгоритм, записанный на языке, “понятном” компьютеру.

8. Единица измерения количества информации.

9. Язык программирования, названный в честь первой женщины — программиста.

10. Логическая копия данных, имеющихся в другом месте или в другом представлении.

11. Районный центр Смоленской области, с которым связано название наступательной операции Красной Армии во время Великой Отечественной войны.

13. Итальянский физик и физиолог, в честь которого названа единица измерения напряжения электрического тока.

15. Системное устройство в компьютере, а также бытовой прибор, в заданный момент времени выдающий определенный сигнал.

18. В программировании — процесс пошагового выполнения программы, в ходе которого программист видит последовательность выполнения команд и значения переменных на данном шаге выполнения программы, что позволяет легче обнаруживать ошибки.

По вертикали

1. В программировании — одновременное перемещение битовых значений вправо или влево.

2. Указатель места на экране компьютера.

3. В многозадачных системах — перемещение задачи из оперативной памяти на диск для освобождения места для работы задач с более высоким приоритетом, а также освобождение вагона от груза.

4. Часть текста.

5. Место хранения (постоянного или временного) информации в компьютере.

6. Инструментальная программная оболочка, в которой происходит работа пользователя, а также день недели.

10. Поворот на 360°.

12. Элемент электронной таблицы.

14. Последовательность символов, предназначенная для чтения человеком.

16. Один из первых языков программирования высокого уровня.

17. Часть света, в которой изготавливаются компьютеры “желтой” сборки.

Симметричная дата

Дата 21.02.2012 читается одинаково слева направо и справа налево. А будут ли после нее еще такие даты в нашем столетии?

Числовой ребус “ИВА и ТЕТИВА”

Решите, пожалуйста, числовой ребус:

ТЕТИВА = ИВА²

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

Мюнхгаузен-рыбак

Известный “правдец” (антоним от слова *лжец* ☺) барон Мюнхгаузен каждый день ходил на рыбалку, а возвратившись, говорил: “Сегодня я поймал больше рыб, чем позавчера, но меньше, чем неделю назад”.

1. Могли его высказывания быть истинными семь дней подряд?

2. Какое наибольшее число дней подряд эти высказывания могли быть истинными?

ЗАДАЧНИК

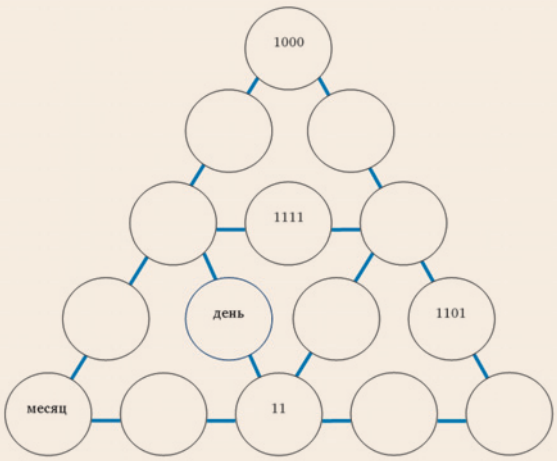
Кто этот человек?

Н.Е. Кордина,
учитель информатики школы № 1,
г. Демидов, Смоленская обл.

Решив четыре приведенные ниже задачи, по полученным результатам определите (используя Интернет или другие источники) фамилию и имя человека, о котором идет речь в заданиях. В ответе приведите также полученную при выполнении частных заданий информацию и ответьте на дополнительный вопрос: “В чем суть научной проблемы, решенной этим человеком?”.

1. Дата рождения

В кружочки (см. рисунок справа) были вписаны числа от 1 до 15 так, чтобы по периметру каждого из четырех треугольников сумма была одинакова. Затем числа перевели в двоичную систему счисления и часть чисел удалили. Восстановите расположение чисел, и вы узнаете день и месяц рождения “искомого” человека.



2. Место рождения

Наш герой родился в селе, название которого состоит из семи букв, номера которых в русском алфавите отвечают следующим условиям:

1) номер последней буквы равен разности 111_2 и 101_2;

2) номер первой буквы — это наименьшее общее кратное номера последней буквы и 111_2;

3) номер второй буквы равен корню x уравнения

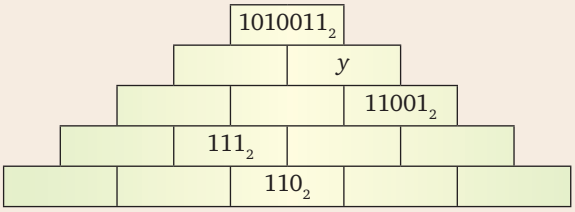
$$\frac{x}{10_8} = 1{,}01_2;$$

4) третья буква совпадает с пятой, и ее номер в $110{,}1_2$ раз больше номера последней;

5) номера четвертой и шестой букв — последовательные степени номера последней буквы.

3. Работа в институте

Человек, которого вы ищете, возглавлял в качестве директора Математический институт им. В.А. Стеклова более y лет. Неизвестное число вы найдете, восстановив пирамиду из чисел:



Каждое число в ней — это сумма чисел в двух клетках под ними. На первом "этаже" пирамиды, кроме числа 110_2, находятся также 111_2, 1010_2, 101_2.

4. Проблема

Его основные работы относятся к разделу математики — теории чисел. Нашему герою удалось решить проблему, получившую имя одного из математиков. Фамилия этого математика состоит из восьми букв. Номера этих букв в русском алфавите отвечают следующим условиям:

1) номер первой буквы — в системе счисления с этим основанием число 17 записывается в виде 101;

2) номер второй буквы — в системе счисления с этим основанием запись числа 483_{10} является трехзначной, начинается на 1 и заканчивается на 3;

3) номер третьей буквы равен наименьшему двузначному основанию системы счисления, запись которого в двоичной системе счисления оканчивается на 101;

4) номер четвертой буквы — это наименьшее основание системы счисления, кратное 3, в которой запись числа 27 оканчивается на 3, увеличенное в 101_2 раз;

5) номер пятой буквы равен количеству единиц в двоичной записи десятичного числа 173;

6) номер шестой буквы — основание системы счисления, в которой представлена информация в компьютерах;

7) номер седьмой буквы — ни простое, ни составное число;

8) номер восьмой буквы — это двузначное простое число; его запись в десятичной системе счисления отличается от записи в восьмеричной системе на 4, при этом первые цифры совпадают.

От редакции. Ответы, пожалуйста, присылайте в редакцию. Можно выполнять не все задания.

Четыре блюда

На завтрак приготовили блины со сгущенкой, пироги с капустой, оладьи со сметаной, пироги с вареньем. Лена, Аня, Таня и Света выбрали разные блюда. Определите, какое блюдо выбрала каждая из девочек, если известно, что Лена и Аня сладкоежки, Таня и Аня больше всего любят пироги.

Задача предназначена для учащихся 1–7-х классов.

Дележ яблок в среде Microsoft Excel ☺

В рубрике "Школа программирования" описана методика решения задачи о дележе яблок между четырьмя сестрами. Предлагаем читателям решить задачу средствами программы Microsoft Excel или подобной.

Вид листа показан на рисунке:

	A	B
1	**Дележ яблок**	
2	Было у Ани	
3	После дележа яблок Ани у девочек стало яблок:	
4	– у Оли	
5	– у Маши	
6	– у Тани	
7	– у Ани	
8	После дележа яблок Оли у девочек стало яблок:	
9	– у Маши	
10	– у Тани	
11	– у Ани	
12	– у Оли	
13	После дележа яблок Маши у девочек стало яблок:	
14	– у Оли	
15	– у Тани	
16	– у Ани	
17	– у Маши	
18	После дележа яблок Тани у девочек стало яблок:	
19	– у Оли	
20	– у Маши	
21	– у Ани	
22	– у Тани	

Исходное число яблок у Ани вводится в ячейку B2, искомые значения выводятся в ячейках B19:B22. Необходимые формулы введите самостоятельно.

Ответы на задание 2 для самостоятельной работы, предложенные в статье "Дележ яблок", пожалуйста, присылайте в редакцию.

Задачи на системы счисления

Е.А. Мирончик,
учитель информатики МБНОУ "Лицей № 111",
г. Новокузнецк Кемеровской обл.

1. Запишите числа 234_7, 432_7, 432_{10}, 234_5 в порядке возрастания. Ответ обоснуйте.

2. В электронных таблицах столбцы нумеруются последовательными буквами латинского алфавита: A, B, C, …, Z. После столбца Z идут столбцы AA, AB, AC, …, ZZ, AAA, AAB, AAB, … .

а) Выпишите номера столбцов ZZZ, AXZ, ZAX, AZX, AAA, XXX в порядке их следования;

б) между какими двумя столбцами находится столбец KYZ?

Почему данные задачи представлены как задачи на системы счисления?

3. Выпишите числа $2^n - 2$, 2^n, $2^{n+1} + 1$, где $n > 10$, в порядке возрастания количества единиц в их двоичной записи.

4. Каждую строку черно-белого рисунка размером $n \times n$ пикселей закодировали следующим образом: пиксель черного цвета обозначили единицей, белого — нулем и соответствующее число представили в восьмеричной системе счисления. Весь рисунок при этом кодируется последовательностью восьмеричных чисел: 46, 151, 251, 51, 51, 51, 51, 46. Что изображено на рисунке?

Аналогично можно этот рисунок закодировать в шестнадцатеричной системе счисления. Перекодируйте рисунок соответствующим образом.

5. Два некоторых десятичных числа *X* и *Y* перевели в системы счисления с основаниями 16 и 8. Часть символов при записи была утеряна. Два из четырех полученных чисел имеют вид (позиции утерянных символов обозначены символом “*”):

$$A*_{16} \text{ и } 1*3_8.$$

Можно ли сделать вывод о том, какое из чисел *X* и *Y* больше, или о равенстве этих чисел?

6. Некоторое десятичное число *X* перевели в системы счисления с основаниями 16, 8, 4, 2. Часть символов при записи была утеряна. Позиции утерянных символов обозначены символом “*”:

$$X = E*_{16} = *5*_8 = ***1_4 = *****1**_2.$$

Восстановите все числа и определите число *X*.

7. Некоторое десятичное число *X* перевели в системы счисления с основаниями 16 и 8. Часть символов при записи утеряна (позиции утерянных символов обозначены символом “*”):

$$**A_{16} \text{ и } 4*2_8.$$

Можно ли однозначно определить значение числа *X*? Если нет, то укажите все возможные значения числа *X*.

На олимпиаде по информатике

Об учениках, занявших первые пять мест на олимпиаде по информатике, имеется пять высказываний:

1) первое место занял Вася, а Юра — второе;

2) Саша занял второе место, а Вася — пятое;

3) второе место занял Иван, а Гриша оказался четвертым;

4) на первом месте был Гриша, а Юра — четвертым;

5) Юра был четвертым, а Иван — вторым.

Известно, что в каждом высказывании одно утверждение верное, а второе — нет. Кто какое занял место?

Продажа орехов

Торговец принес на рынок мешок орехов. Все орехи были одинаковые. Первый покупатель купил один орех, второй — два ореха, третий — четыре, и так далее: каждый покупатель покупал вдвое больше орехов, чем предыдущий. Орехи, купленные последним покупателем, весили 50 кг, после чего у торговца остался один орех. Можно ли определить:

1) сколько покупателей купили орехи;

2) какое число орехов было у торговца вначале;

3) сколько килограммов орехов было у торговца вначале?

Если какое-то значение (или все) определить нельзя, то укажите, что надо еще знать, чтобы найти его (их).

Ответы к заданию “Четыре вопроса” (рубрика “Поиск информации”)

1. В Чрезвычайной комиссии по расследованию преступлений царского режима заседал поэт Александр Блок.

2. Растение, которое идет на строение домов, мостов, мебели, зонтиков и вееров, — бамбук.

3. Военный плащ древних римлян назывался *сагум*.

4. Морскую принцессу из диснеевского мультфильма “Русалочка” зовут Ариэль.

Ответы прислали:

— Ахматгалиева Диана, Белая Алена, Гусева Маргарита, Димакова Арина, Нестерова Анастасия и Салимов Владислав, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Балашова Ирина и Скареднева Анастасия, г. Рязань, школа № 44, учитель **Марцинкевич Е.Е.**;

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

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

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

— Иванов Николай, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

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

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

— Яковлев Степан, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Решение японских головоломок “судоку”, опубликованных в майском выпуске, прислали:

— Ахматгалиева Диана, Белая Алена, Гусева Маргарита, Димакова Арина, Нестерова Анастасия и Салимов Владислав, г. Челябинск, школа № 124, учитель **Юртаева Г.Ю.**;

— Барановская Татьяна и Жукова Ирина, средняя школа села Горелово Тамбовской обл., учитель **Шитова Л.А.**;

— Лазаренко Нина, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Скареднева Анастасия, г. Рязань, школа № 44, учитель **Марцинкевич Е.Е.**;

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

Правильные ответы на задания, опубликованные в апрельском выпуске “В мир информатики”, прислали также ученики средней школы поселка Ерофей Павлович, Амурская обл., Сковородинский р-н, учитель **Краснёнкова Л.А.**:

— Гажалова Татьяна (задача “Четыре девочки”);

— Дьякова Екатерина и Козлова Юлия (кроссворд);

— Степанов Никита и Шалагинов Николай (девять ребусов).

Решение задачи “Восемь семейных пар” прислали:

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

— Васина Светлана и Хомутова Евгения, средняя школа деревни Муравьево, Вологодская обл., учитель **Муравьева О.В.**;

— Яковлев Степан, Чувашская Республика, г. Канаш, Канашский педагогический колледж, преподаватель **Воеводина Р.В.**

Мы не приводим решение из-за его значительного объема. А вот Арину, Диану, Евгению, Светлану и Степана, не побоявшихся взяться за решение этой непростой задачи, редакция решила наградить дипломами. Молодцы!

MICROSOFT EXCEL УГЛУБЛЕННО

Арифметика рекордов

А.И. Азевич,
Москва

Многоборье — венец легкой атлетики, которую часто называют “королевой спорта”. Наблюдая за выступлениями спортсменов, мы восторгаемся их выносливостью, силой и красотой. Метры, секунды и баллы, заработанные многоборцами в упорной борьбе, скрупулезно подсчитывает компьютер. Но “кухня цифр” скрыта от зрителей. Они видят лишь спортивное мастерство, волю к победе и уже готовые результаты. Попробуем, используя программу Microsoft Excel, приоткрыть скрытую от болельщиков тайну.

В многоборье соревнуются и женщины, и мужчины. Женщины участвуют в семи видах программы (беге 100 м с барьерами, прыжках в высоту, толкании ядра, беге на 200 м, прыжках в длину, метании копья и беге на 800 м), мужчины — в десяти. Соревнования проводятся в течение двух дней. Результат каждого участника определяется комбинацией его показателей в отдельных видах. Но как секунды складываются с метрами? В многоборье применяется особая система начисления баллов. Спортсмен получает их за каждый вид программы. Затем все очки суммируются. Компьютер выдает окончательный результат, который мы и видим на экране или на табло стадиона. В основу подсчетов положена формула $y = a|x - b|^c$, где $|x - b|$ — абсолютная величина разности $x - b$, причем (!) используется целая часть значения y. Поясним, что означают входящие в нее параметры:

— x — это результат, показанный спортсменом в той или иной спортивной дисциплине и выраженный в соответствующих единицах измерений;

	Дисциплина	Единица измерения результата	*a*	*b*	*c*
1	Бег 100 м с барьерами	секунды	9,23075	26,7	1,835
2	Прыжки в высоту	метры	916,325	0,75	1,348
3	Толкание ядра	метры	56,021	1,5	1,05
4	Бег на 200 м	секунды	4,99087	42,5	1,81
5	Прыжки в длину	метры	124,7435	2,1	1,41
6	Метание копья	метры	15,9803	3,8	1,04
7	Бег на 800 м	секунды	0,11193	254	1,88

Рис. 1

№ п.п.	Фамилия, имя	Страна	Бег 100 м с барьерами	Прыжки в высоту	Толкание ядра	Бег 200 м	Прыжки в длину	Метание копья	Бег 800 м
1	Икауниеце Лаура	Латвия	13,71	1,83	12,67	24,16	6,13	51,27	2:12,13
2	Савицкая Кристина	Россия	13,37	1,83	14,77	24,47	6,21	43,7	2:12,19
3	Зелинка Джессика	Канада	12,65	1,68	14,81	23,32	5,91	45,75	2:9,15
4	Скуйте Аустра	Литва	14,00	1,92	17,31	25,43	6,25	51,13	2:20,59
5	Нана Джиму Антуанетт	Франция	12,96	1,80	14,26	24,72	6,13	55,87	2:15,94
6	Йосипенко Людмила	Украина	13,25	1,83	13,90	23,68	6,31	49,63	2:13,28
7	Шварцкопф Лилли	Германия	13,26	1,83	14,77	24,77	6,30	51,73	2:10,50
8	Чернова Татьяна	Россия	13,48	1,80	14,17	23,67	6,54	46,29	2:9,56
9	Мельниченко Анна	Украина	13,32	1,80	12,96	24,09	6,40	43,86	2:12,90
10	Эннис Джессика	Великобритания	12,54	1,86	14,28	22,83	6,48	47,49	2:8,65

Рис. 2

	A	B	C	D	E	F	G	…	P	Q	R	S
1	Таблица результатов в семиборье (женщины, Лондон-2012)											
2	№ п. п.	Фамилия, имя	Страна	Бег 100 м с барьерами	Очки	Прыжки в высоту	Очки		Бег 800 м	Очки	Общая сумма	Место
3	1	Икауниеце Лаура	Латвия									
4	2	Савицкая Кристина	Россия									
…												

Рис. 3

— *a*, *b*, *c* — коэффициенты, относящиеся к виду соревнований;

— *y* — баллы, соответствующие показанному результату.

Приведем таблицу коэффициентов *a*, *b*, *c* женского многоборья [1].

Найдем результаты спортсменок в каждой дисциплине, которые были показаны на Олимпийских играх 2012 года в Лондоне (например, их можно взять с сайта [2]). По ним подготовим список спортсменок, занявших первые десять мест, и их результаты для каждого из семи видов программы, расположив их в произвольном порядке (не учитывая занятые ими места) — см. *рис.* 2.

Используя показатели первых десяти участниц программы, подготовим электронную таблицу, состоящую из нескольких листов:

— лист 1 — с результатами спортсменок; этот лист назовем Семиборье[7] — см. *рис.* 3;

— лист 2 — с таблицей коэффициентов *a*, *b*, *c* женского многоборья; имя листа — Коэффициенты_Женщины.

На первом листе в столбцы D, F, H, J, L, N и P внесем показатели по видам программы из таблицы на *рис.* 1. В столбце R получим итоговый результат каждой спортсменки по совокупности семи дисциплин. Для этого формулу в ячейке R3: =СУММ(E3;G3;I3;K3;M3;O3;Q3) распространим (скопируем) на остальные ячейки.

И, наконец, замыкает таблицу столбец, в котором будет показано место, занятое на Олимпиаде каждой спортсменкой (см. *рис.* 3).

Внимание! В процессе заполнения результатов спортсменок в беге на 800 м возникает одна проблема. В таблице на *рис.* 2 значение в этом виде указано в формате вида ?:??,??, где первый слева символ “?” соответствует количеству минут, второй и третий — количеству целых секунд, два крайних справа — количеству сотых долей секунды. А в таблице на *рис.* 1 в качестве результата, используемого для расчетов, указано время в секундах (возможно, нецелое значение). Это означает, что все значения из последнего столбца таблицы на *рис.* 1 должны быть пересчитаны в количество секунд. Это несложно сделать “в уме”, прибавив к 120 (число секунд в двух минутах) секун-

[7] Можно, конечно, оставить имя листа, имеющееся, как говорится, “по умолчанию” (Лист1). Однако содержательные имена листов помогают понять, какая информация представлена на том или ином листе. Чтобы переименовать лист, следует щелкнуть на ярлычке этого листа правой кнопкой мыши и в появившемся контекстном меню выбрать пункт Переименовать. — **Прим. ред.**

ды (дробное число). Например, для спортсменки из Латвии значение 2:12,13 соответствует 132,13 сек.

Теперь об оформлении листа Коэффициенты_Женщины. На первый взгляд его можно оформить аналогично таблице на *рис.* 1:

	A	B	C	D
1		*a*	*b*	*c*
2	Бег 100 м с барьерами	9,23075	26,7	1,835
3	Прыжки в высоту	916,325	0,75	1,348
4	Толкание ядра	56,021	1,5	1,05
5	Бег на 200 м	4,99087	42,5	1,81
6	Прыжки в длину	124,7435	2,1	1,41
7	Метание копья	15,9803	3,8	1,04
8	Бег на 800 м	0,11193	254	1,88

Рис. 4

Однако в этом случае мы столкнемся с очень серьезной проблемой, о которой подробно расскажем ниже.

Возвратимся к первому листу и обсудим формулу для расчета баллов первой спортсменки за бег на 100 м с барьерами (в ячейке E3). Она имеет вид:

=ОТБР(Коэффициенты_Женщины!B2*ABS(D3-Коэффициенты_Женщины!C2)^Коэффициенты_Женщины!D2)

Примечание. При ссылке в формуле на ячейку из другого листа перед адресом ячейки следует указать имя соответствующего листа с символом "!".

Такая формула основана на формуле для расчета *y*, приведенной в начале статьи (функция ABS возвращает абсолютную величину ее аргумента, функция ОТБР — целую часть, символ "^" — знак операции возведения в степень). Однако приведенная формула не может быть скопирована ни на другой вид спорта, ни на другую строку (результат другой спортсменки). И никакое использование абсолютных и смешанных адресов (с двумя или одним символом "$") не может решить эту проблему.

Целесообразно лист Коэффициенты_Женщины оформить так, как показано на *рис.* 5.

Обратите внимание: числа в столбцах вносятся через один (B, D и т.д.). Это сделано для того, чтобы в дальнейшем без труда копировать формулы на первом листе.

В этом случае на этом листе формула в ячейке E3 примет вид:

=ОТБР(Коэффициенты_Женщины!B$3*ABS(D3-Коэффициенты_Женщины!B$4)^Коэффициенты_Женщины!B$5)

Такую формулу можно распространить (скопировать) на другие строки столбца E. Благодаря использованию так называемых "смешанных ссылок" (B$3, B$4 и B$5) при копировании в новых формулах будут использованы адреса тех же ячеек, а вместо результата Лауры Икауниеце (из ячейки D3) используются результаты той или иной участницы.

Приведенную формулу можно также скопировать (но не распространить!) и в столбцы с другими видами спорта. Например, при копировании ее в диапазон ячеек G3:G12 имя столбца B в ссылках на лист Коэффициенты_Женщины изменится на D (то есть будут использованы коэффициенты *a*, *b*, *c* для прыжков в высоту).

В результате всех действий в столбцах E, G, I, K, M, O и Q появятся баллы, заработанные спортсменками во всех видах программы, а в столбце R — суммарный результат.

Итак, данные спортсменок внесены. Формулы работают. Общие суммы баллов подсчитаны. Осталось определить места, которые заняли девушки. Это позволит сделать функция РАНГ. Она возвращает ранг некоторого числа в списке чисел. Ранг числа — это его место относительно других значений в списке при сортировке всех чисел в порядке возрастания или убывания.

Общий вид функции:

РАНГ(число;ссылка;порядок)

где число — число, для которого определяется ранг;

— ссылка — диапазон ячеек с числами (нечисловые значения игнорируются);

— порядок — число, определяющее способ сортировки; в нашем случае он должен быть равен нулю, чтобы определялось место числа при сортировке списка чисел в порядке убывания.

Например, для числа 12 ранг в списке значений 7, 15, 12, 11, 9, 10 будет равен 2[8].

Итак, в ячейку S3 внесем формулу =РАНГ(R3;$R$3:$R$12;0) и распространим (скопируем)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	**Таблица коэффициентов**													
2		Бег на 100 м/б		Прыжки в/в		Толкание ядра		Бег на 200 м		Прыжки в длину		Метание копья		Бег на 800 м
3	*a*	9,23075		916,021		56,021		4,99087		124,7435		15,9803		0,11193
4	*b*	26,7		0,75		1,5		42,5		2,1		3,8		254
5	*c*	1,835		1,348		1,05		1,81		1,41		1,04		1,88

Рис. 5

[8] Интересно, что функция РАНГ присваивает повторяющимся числам одинаковый ранг. При этом наличие повторяющихся чисел влияет на ранг последующих чисел. Например, если в списке целых чисел дважды встречается число 10, имеющее ранг 5, число 9 будет иметь ранг 7. Это соответствует правилу присвоения мест в ряде видов спорта, когда команды или спортсмены, показавшие одинаковый результат, занимают одинаковое место.

	A	B	C	D	E	F	G	…	P	Q	R	S
1	**Таблица результатов в семиборье (женщины, Лондон-2012)**											
2	№ п. п.	Фамилия, имя	Страна	Бег 100 м с барьерами	Очки	Прыжки в высоту	Очки		Бег 800 м	Очки	Общая сумма	Место
3	1	Икауниеце Лаура	Латвия	13,71	1020	1,83	1016		132,13	934	6416	9
4	2	Савицкая Кристина	Россия	13,37	1069	1,83	1016		132,19	933	6452	8
…												

Рис. 6

ее на ячейки S4:S12. После этого в списке напротив каждой спортсменки появится занятое ею место — см. *рис.* 6.

Несмотря на то что мы все сделали аккуратно, итоговый результат смотрится не так, как хотелось бы. А именно — желательно список спортсменок разместить в соответствии с занятыми ими местами (во многих видах спорта во время прямых трансляций зрителям показывают такой список спортсменов, который в режиме текущего времени меняется). Список согласно занимаемых (занятых) мест получим на третьем листе, которому дадим имя Места.

Заполним на нем столбец A номерами мест:

	A	B	C
1	1		
2	2		
3	3		
…			
10	10		

Рис. 7

В столбце B получим фамилии и имена спортсменок, занявших соответствующее место. Это можно сделать с помощью функции ИНДЕКС. Во всех тонкостях работы этой функции здесь мы разбираться не будем, а скажем, что в нашем случае в ячейке B1 следует указать спортсменку из диапазона B3:B12 на листе Семиборье, номер места которой в диапазоне S3:S12 равен 1 (значению в ячейке A1 на листе Места). Соответствующее оформление функции:

```
=ИНДЕКС(Семиборье!B$3:B$12;ПОИСКПОЗ(СТРОКА();Семиборье!S$3:S$12;))
```

— где функции ПОИСКПОЗ и СТРОКА — стандартные функции Excel (их особенности также описывать не будем).

Так как в функции использованы смешанные ссылки (адреса), то ее можно скопировать и на другие ячейки столбца B.

Укажем также в отсортированном списке спортсменок страны, которые они представляют (в столбце C). В ячейку C1 введем формулу, во многом аналогичную приведенной чуть выше:

```
=ИНДЕКС(Семиборье!C$3:C$12;ПОИСКПОЗ(СТРОКА();Семиборье!S$3:S$12;))
```

Ее также можно скопировать. Результат показан на *рис.* 8.

	A	B	C
1	1	Эннис Джессика	Великобритания
2	2	Шварцкопф Лилли	Германия
3	3	Чернова Татьяна	Россия
…			

Рис. 8

Примечание. Если на листе Семиборье еще не будет данных по первому виду спорта, то у всех спортсменок будет установлен ранг, равный 1 (все заняли первое место ☺) — см. сноску 8 на с. 60. Кроме того, на листе Места в строках 2–10 выведены сообщения об ошибке типа #Н/Д. По мере заполнения листа данными эти значения, конечно, будут меняться.

Для полноты картины на листе Места не хватает заключительного аккорда. Вставим на лист (**Вставка | Рисунок | Из файла**) фотографию[9] российской спортсменки Татьяны Черновой, заслуженно получившей в этом неимоверно трудном виде спорта бронзовую медаль — см. *рис.* 9 на с. 62.

Все спортсменки выступили. Они изо всех сил, с невероятным напряжением и страстью старались показать все, на что способны. Результаты посчитаны. Итоги подведены. Кто-то из болельщиков радовался успехам своей страны, кто-то огорчался, надеясь в будущем на более высокие места.

Мы же вспомнили волнительные минуты прошедшей Олимпиады. Это был незабываемый праздник красоты, молодости, силы и мастерства. Наблюдая телевизионные трансляции из Лондона, мы с нетерпением ждали результатов спортсменов. А неутомимые компьютеры, без которых сегодня не может обойтись ни одно соревнование, считали секунды, метры, минуты. И конечно же баллы! Как это было, известно лишь специалистам в области обработки спортивной

[9] Фотография, как и другие вспомогательные материалы к статье, представлена на диске к данному номеру журнала.

	A	B	C	D	E	F	G
1	1	Эннис Джессика	Великобритания				
2	2	Шварцкопф Лили	Германия				
3	3	Чернова Татьяна	Россия				
4							
5							
6							
7							
8							
9							
10							

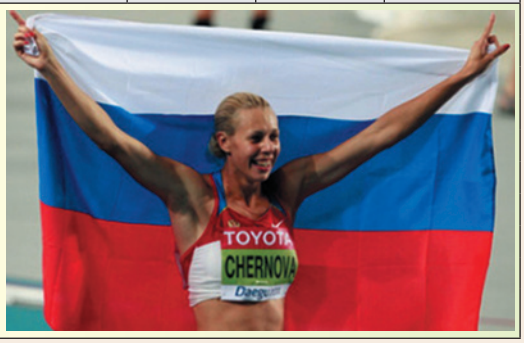


Рис. 9

информации. Теперь уже, правда, не только им, но и нам!

Задания для самостоятельной работы

1. Оформив листы согласно описанной в статье методике, определите места, занятые каждой из десяти спортсменок.

2. Подготовьте аналогичные листы для подведения итогов соревнований мужчин в десятиборье (для спортсменов, занявших первые десять мест). Результаты также можно найти по адресу [3]. А значения коэффициентов *a*, *b*, *c* для "мужских" видов спорта следующие — см. таблицу справа.

	Дисциплина	Единицы измерения результата	a	b	c
1	Бег на 100 м	секунды	25,4348	18	1,81
2	Прыжки в длину	метры	90,5674	2,2	1,4
3	Толкание ядра	метры	51,39	1,5	1,05
4	Прыжки в высоту	метры	585,64	0,75	1,42
5	Бег на 400 м	секунды	1,53775	82	1,81
6	Бег на 110 м с барьерами	секунды	5,74354	28,5	1,92
7	Метание диска	метры	12,91	4	1,1
8	Прыжки с шестом	метры	140,182	1	1,35
9	Метание копья	метры	10,14	7	1,08
10	Бег на 1500 м	секунды	0,03768	480	1,85

3. Подготовьте аналогичные листы для подведения итогов соревнований мужчин на чемпионате мира по современному пятиборью 2013 года (для спортсменов, занявших первые десять мест). Всю необходимую информацию найдите самостоятельно.

Источники сети Интернет

1. *ru.wikipedia.org / wiki / %D1%E5%EC%E8%E1%EE%F0%FC%E5*

2. *news.sportbox.ru / Vidy_sporta / Events / London_2012 / athletics / stats*

3. *news.sportbox.ru / Vidy_sporta / Events / London_2012 / athletics / stats / turnir_993*

ПОИСК ИНФОРМАЦИИ

Четыре вопроса

Ответы на приведенные ниже вопросы найдите в Интернете или в других источниках информации.

1. Чем занимался Томас Мор, прежде чем стать канцлером казначейства?

2. Бальтазар Грасиан полагал, что в 20 лет царит чувство, в 30 — талант. А что в сорок?

3. Кого муж царицы Нефертити провозгласил "единым богом"?

4. Кому из античных историков принадлежит фраза "У кого нет врагов, того губят друзья"?

ДЛЯ ЭРУДИТОВ

Викторина

Вы, наверное, слышали выражение "газетная утка"? Так называют лживое известие, напечатанное в газете. Определите, какая из четырех новостей является газетной уткой, а какая — нет. Но не наугад, а используя Интернет или другие источники информации.

1. Американская компания Solar Roadway строит дороги нового поколения, которые помогают сберечь городской бюджет в зимнее время — снег тает благодаря встроенным в дорожное полотно нагревателям, подпитываемым энергией солнца.

2. Российскими стоматологами для изготовления зубных протезов, коронок, мостов и вкладок используется около 13 тонн платины в год.

3. При использовании хлопковых подгузников вероятность возникновения сыпи в пять раз больше, чем при использовании одноразовых памперсов.

4. В Японии никто никогда не поднимет кошелек (разве для того, чтобы отнести его в полицейский участок). Там считается, что за такой неожиданный подарок судьба вскоре спросит очень строго, отняв у тебя что-то более ценное.

ЯПОНСКИЙ УГОЛОК

Необычные судоку

Предлагаем читателям решить две японские головоломки “судоку”, но не обычные, а с особенностями:

1)

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

2)

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

В первом случае в клеточки нужно вписать цифры от 1 до 9 так, чтобы во всех строках, во всех столбцах и во всех выделенных квадратиках 3 × 3, а также на главных диагоналях цифры не повторялись. Во втором — цифры не должны повторяться во всех строках, во всех столбцах и всех выделенных фигурных областях.

Ответы, пожалуйста, присылайте в редакцию (можно решать отдельные судоку).

КРЕПКИЙ ОРЕШЕК

Как обычно, в этой рубрике разбираются задачи, решение которых вызвало трудности.

Задача “Кучки монет”

Напомним условие: “Имеются 10 кучек монет, по 10 монет в каждой. Одна из кучек целиком состоит из фальшивых монет, но какая именно — неизвестно. Все настоящие монеты одинаковые, весом 10 г, все фальшивые тоже одинаковые, весом 11 г. Есть также пружинные весы, показывающие вес с точностью до 1 г. За какое минимальное количество взвешиваний можно определить кучку, целиком состоящую из фальшивых монет?

А если кучек 11 (в 11-й кучке тоже 10 монет)?

Решите также вариант задачи, в котором всего шесть кучек, и в каких-то двух кучках все монеты фальшивые”.

Благодарим Максима Бобкова и Дениса Воскресенского, Владимирская обл., г. Струнино, школа № 11 (учитель **Волков Ю.П.**), приславших правильное решение всех трех задач, они будут награждены дипломами, и приведем решение первой задачи.

Выяснить требуемое можно, пользуясь такой системой указаний:

1. Пронумеровать мешки числами от 1 до 10.
2. Из каждого мешка извлечь столько монет, каков его номер.
3. Определить на весах суммарную массу M всех извлеченных монет.
4. Проверить условие $M = 550$. Если да, то перейти к указанию 7, иначе — к следующему указанию.
5. Определить разность R, равную $M - 550$.
6. Объявить, что в мешке с номером R монеты фальшивые. Перейти к указанию 8.
7. Объявить, что мешка с фальшивыми монетами нет.
8. Конец.

С учетом приведенного решения предлагаем читателям решить две другие задачи, приведенные в конце условия, и прислать ответы в редакцию.

ВНИМАНИЕ! КОНКУРС

Конкурс № 105

В качестве задания этого конкурса предлагаем решить задачи на системы счисления, представленные в рубрике “Задачник” в этом выпуске.

Ответы отправьте в редакцию до 1 декабря 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. Можно решать не все задачи.

журнал

Информатика – Первое сентября

ПОДПИСКА В ПОЧТОВЫХ ОТДЕЛЕНИЯХ РФ

1-е полугодие 2014 года

Индекс	Название издания	Периодичн. в полугодие	1 месяц		6 месяцев	
			Каталожная цена (руб.)	Подписная цена (руб.)	Каталожная цена (руб.)	Подписная цена (руб.)
Название блока в разделе «Журналы»	**ПЕРВОЕ СЕНТЯБРЯ. ЖУРНАЛЫ ИЗДАТЕЛЬСКОГО ДОМА** (499)249-31-38					
79066	**Информатика – Первое сентября. Бумажная версия** С электронными приложениями и презентациями. *В июне не выходит.* *Подписка на июнь не принимается* *(-) 160 г 64 стр.*	5	308.00		1540.00	
12684	**Информатика – Первое сентября. Электронная версия на CD (полная копия бумажной версии)** *В июне не выходит.* *Подписка на июнь не принимается* *(-) 75 г*	5	118.80		594.00	

Подписку на журналы ИД «Первое сентября» можно оформить также на сайте www.1september.ru
При оформлении подписки на сайте оплата производится
по квитанции в отделении банка или электронными платежами on-line

VISA MasterCard QIWI WebMoney