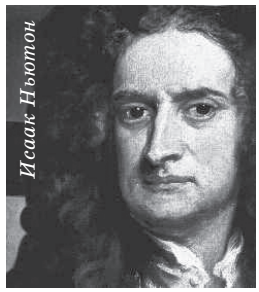
Учителю информатики: памятные даты января

1 января 1878 года на юге полуострова Ютландия родился Агнер Крарул Эрланг (1878–1929) — датский математик, пионер в изучении трафика в телекоммуникационных системах.

В 1909 году была опубликована первая работа Эрланга “Теория вероятностей в телефонии”, получившая потом признание во всем мире. Его формулы стали фундаментом при разработке теории телетрафика [1].



Именем этого исследователя названа единица интенсивности нагрузки в телекоммуникационных системах (*эрланг*).

4 января 1643 года (по григорианскому календарю) в Вулсторпе (Англия) родился Исаак Ньютон (1643–1727) — один из величайших ученых в истории человечества, заложивший основы современного естествознания.

По словам А.Эйнштейна [2], “Ньютон был первым, кто попытался сформулировать элементарные законы, определяющие ход широкого класса процессов в природе с высокой степенью полноты и точности” и “...оказал своими трудами глубокое и сильное влияние на все мировоззрение в целом”.

Его именем названа единица силы в Международной системе единиц (*ньютон*).

Памятники великим ученым. Собрание картин Либига. Памятник находится в Кембридже

Для человека, разработавшего научные принципы, которые остаются актуальными и в наши дни, Ньютон был человеком с неортодоксальными интересами: его чрезвычайно занимала алхимия — мистическое искусство превращения металлов в золото. Эти исследования отнимали немалую долю времени великого ученого, и он оставил после себя обширные записи о проводимых им алхимических опытах, хотя и не опубликовал ни одной работы на эту тему.

7 января 1954 года в нью-йоркском отделении фирмы IBM на электронной счетной машине IBM-701 была осуществлена первая реализация машинного перевода. Машина перевела с русского языка на английский 60 несложных предложений. Например: “Обработка повышает качество нефти” или “Международное понимание является важным фактором в решении политических вопросов”. Для этого был составлен словарь из 250 слов и создана специальная программа, использующая шесть правил перевода. Это событие вошло в историю информатики как “Джорджтаунский эксперимент”.

Продолжение на с. 2

СОДЕРЖАНИЕ НОМЕРА

№ 1 (506)

1–15 января 2006

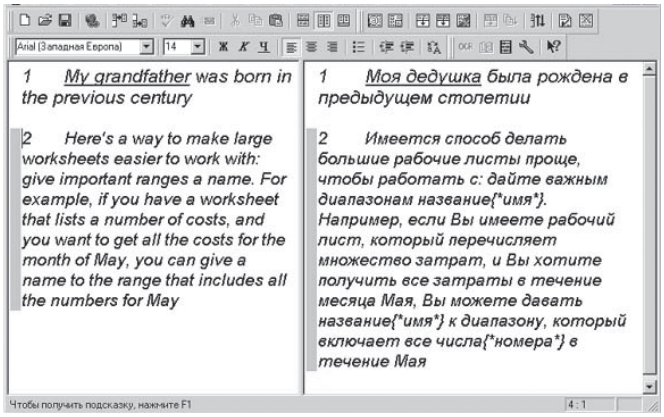
ИНФОРМАТИК

Методическая газета для учителей информатики

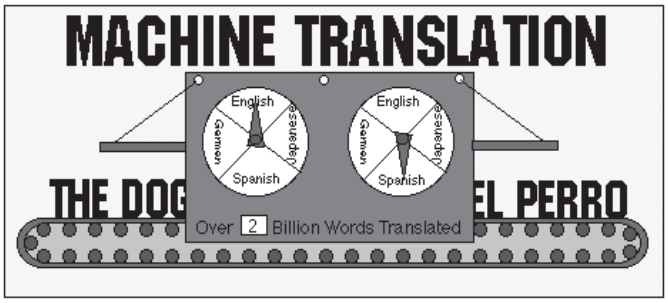
Учителю информатики: памятные даты января

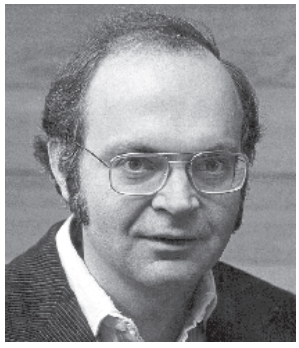
Продолжение. См. с. 1

В этой связи нельзя не вспомнить и Алексея Андреевича Ляпунова, который был одним из основателей кибернетики в нашей стране. В том же 1954 году он высказал мысль, что в будущем развитии машин перспективным является их использование как "подсобного средства мышления" [3], и это будет иметь большее значение, чем непосредственное вычисление. Среди прикладных применений ЭВМ Ляпунов выделял и проблему машинного перевода. "Перевод текстов с одних языков на другие имеет большое значение для развития культуры и является необходимым звеном в обмене культурными ценностями, созданными различными народами" [3]. А поскольку на перевод текстов затрачивается огромный труд, то А.А. Ляпунов считал назревшим "вопрос об автоматизации и механизации процесса перевода. В последнее время для механизации перевода открылись большие возможности, связанные с созданием автоматических электронных вычислительных машин" [3].



Примеры машинного перевода. Грамматические и прочие согласования не всегда соблюдаются, но смысл текста, как правило, понять можно





Дональд Эрвин Кнут

10 января 1938 года в Милуоки (штат Висконсин) родился Дональд Эрвин Кнут — автор знаменитого труда "The Art of Computer Programming" ("Искусство программирования").

"На мировом рынке компьютерной литературы существует ряд книг, предназначенных для обучения основным алгоритмам и используемых при программировании. Их довольно много, и они в значительной степени конкурируют между собой. Однако среди них есть особая книга. Это труд "*Искусство программирования*" Д.Э. Кнута, который стоит вне всякой конкуренции и входит в золотой фонд мировой литературы по информатике..."

Издатели русского перевода книги Дональда Кнута "*Искусство программирования*".

Профессор Кнут удостоен многочисленных премий и наград, среди которых [4]: премия Ассоциации по вычислительной технике (ACM Grace Murray Hopper Award), премия имени Тьюринга (ACM Alan M. Turing Award), медаль за научные заслуги (National Medal of Science), премия Института инженеров по электротехнике и радиоэлектронике (IEEE Computer Pioneer Award), премия Математической ассоциации США (MAA Lester R. Ford Award), награда за достижения в области передовых технологий (Kyoto Prize). Является почетным доктором многих университетов мира, в том числе Санкт-Петербургского университета. Дональд Кнут является также создателем всемирно известной "компьютерной типографии", которая состоит из систем TEX и METAFONT, предназначенной для издания математических книг.

15 января 1850 года в Москве родилась Софья Васильевна Ковалевская (1850–1891) — первая женщина-профессор, первая женщина — член-корреспондент Петербургской академии наук, член Московского математического общества.

Софья Васильевна Ковалевская относилась к числу талантливых ученых и интересных людей. Она была поэтом и публицистом, неплохо разбиралась в психологии, имела богатую фантазию и вообще была человеком многогранным. Друзьями Ковалевской являлись и ученые, и писатели, и политические деятели. Кроме научных работ, ей принадлежит ряд художественных произведений. Но в первую очередь эта женщина была математиком. Много труда стоило Ковалевской пробить себе дорогу, преодолеть общее предубеждение против женщины-исследователя. Но настолько ярок был ее талант, настолько велика тяга к науке, что величайшие ученые признавали ее своей достойной коллегой.

В 1874 году Ковалевская, которая к тому времени уже несколько лет являлась ученицей выдающегося немецкого математика Карла Вейерштрасса, закончила работу "К теории дифференциальных уравнений в частных производных" и вскоре представила ее в качестве докторской диссертации. Совет Гёттингенского университета присудил Ковалевской степень доктора философии по математике и магистра изящных искусств "с наивысшей похвалой" [5].

Конечно, Ковалевская занималась не только чисто научной деятельностью: она долгое время преподавала в университете, работала в редакции шведского математического журнала "Acta Mathe-

Окончание на с. 48

Как чиновники представляют себе Интернет в школах

Эти строки пишет школьный учитель, первые опыты которого в компьютерных сетях произошли довольно поздно — году в 1992—1993-м, причем почти сразу пришлось осваивать и локальные сети, и их администрирование, и работу в сетях глобальных — первые электронные адреса в нашей школе появились никак не позднее 1993 года (и активно используются по сию пору). Развивалось тогда все это в основном усилиями энтузиастов, никакой речи об "интернетизации" всех школ Москвы (тем более России) еще не только не велось, но, видимо, даже не планировалось. С другой стороны, всем сколько-то разбирающимся в предмете было очевидно, что для успешной работы образовательной сети необходима программа государственная, или по крайней мере многолетняя продуманная и финансируемая программа соответствующего охвата. Тем радостнее было видеть постепенное усиление внимания разных руководителей образования и руководителей государства к развитию сети, к работе в сети школ и школьников, к решению технических проблем. К сожалению, такая оптимистичная оценка продержалась не слишком долго...

В Москве чиновники постепенно стали закидывать школы разными длиннющими занудными анкетами с громкими названиями (от "Паспорта образовательного учреждения" до базы "Всеобуч", где про каждого ребенка спрашивалось до 30 полей разной анкетной информации[1]) вперемешку со странными анкетами с вопросами типа "А зачем школе Интернет? А как вы его собираетесь использовать?"; в последнее время стали спрашивать список электронных адресов и фамилии учителей (спасибо, что не учеников), которые используют Интернет в работе. Как ответить на этот вопрос — почти все сегодня это делают, а что учитель использует в работе непосредственно, а что опосредованно — кто сможет ответить? Например, одна моя коллега недавно на перемене уточняла координаты и часы работы музея на Бородинском поле, другой подбирал себе на кафедру принтер, а третий уточнял расписание поездов перед поездкой со школьниками в Ярославль. Формально можно сказать, что ни к урокам, ни к работе учителя все это не имеет никакого отношения.

В последние годы Информационно-аналитический центр Департамента образования г. Москвы претендует на все большую долю работы по поддержке "интернетизации" школ города. Я по долгу службы общался с техническими специалистами этого центра, когда-то давно — с его начальником. Люди вроде бы вполне разумные и квалифицированные. Тем приятнее было увидеть небольшую книгу В.М. Полонского "Образовательные ресурсы в сети Интернет", только что выпущенную тиражом 2000 экз. от имени этого центра[2]. Книга аккуратная, на весьма плотной бумаге (что странно для методического пособия. Зачем? Ведь книга для работы, а не для парада!). В руки взять приятно! Видимо, наконец сделали нормальный обзор накопленного в Рунете материала. Наработано-то немало за последние 7—10 лет.

Вот только открывать и читать я стал напрасно! Введение пестрит казенными и ничего не значащими "правильными" фразами, из которых нельзя понять, кому адресована эта книга — школе или вузу, директору (завучу), учителю, родителю или ученику (студенту). Видимо, автор сам того не понял. Дальше — больше. Следующий раздел называется "Понятийно-терминологический аппарат". Вот только научного коммунизма в его худшем виде нам и не хватало! И действительно, автор не объясняет новичкам, как по адресу в сети прикинуть, в какой это стране или городе; автор не сообщает, какой программой можно смотреть все это[3]; не найдем мы в этом аппарате и объяснения, что значат странные буквы http, https или ftp. Нет, "понятийный аппарат" включает двухстраничную дискуссию о том, чем портал лучше сайта, причем для надежности приведены два определения, из которых видно, что определение портала лучше и точнее, так как определение портала "дается в документе, утвержденном Коллегией Министерства образования России от 04.04.2002 (протокол № 7)".

Не скрою — в этот момент я уже предвидел, что мои чаяния про эту книгу не оправдываются. Я ожидал грамотной и аккуратной работы по созданию перечня разумных сайтов по поддержке преподавания математики, физики, русского языка, географии, иностранных языков. Нет, тут этого явно не будет! Ждать нечего. Проверю оглавление. К сожалению, я прав. В оглавлении из 64 страниц книги 23 страницы (с 9-й по 32-ю) посвящены "официальным, федеральным и региональным порталам и сайтам", после этого только ШЕСТЬ страниц "образовательных порталов и сайтов учреждений, межшкольных и методических объединений и отдельных лиц". Затем "Информационные центры, базы данных и библиотеки" (12 страниц) и, наконец, "сайты и порталы управления образованием" (9 страниц).

[1] К слову, интересно спросить, а почему эта база недоступна к использованию руководителями учебных округов, ведь по почте несколько раз приходили вопросы о том, учится ли в школе тот или иной ребенок?..

[2] *Полонский В.М.* Образовательные ресурсы в сети Интернет. / Методическое пособие. М., 2005, 64 с. Департамент образования города Москвы. Информационно-аналитический центр. Типография "Интел универсал". (Почему у книги отсутствует издательство, ISBN и прочие подобные атрибуты, не указано!)

[3] В соседней брошюре того же центра настолько превалируют платные продукты фирмы Microsoft, что возникает ощущение, будто крупнейшая корпорация просто спонсирует Центр Департамента образования г. Москвы. Многие мои ученики и коллеги давно используют браузеры Opera или Mozilla вместо громоздкого и неудобного Internet Explorera.

Последние пять страниц книжки заняты "web-адресами источников в области образования и педагогики". Итак, книга явно школьная, а не вузовская. Явно чиновничья, а не учительская. Осталось только отдать ее директору и завучу — пусть пользуются.

Привычка все проверять самому подвела меня в третий раз. Я открыл первый большой раздел. Он сверстан как таблица в три колонки. В первой — название ресурса (1—2 строки), в третьей — его интернет-адрес (столько же), во второй, в центральной, — странный текст о том, каков сайт. Только три примера.

• ***www.mon.gov.ru*** — "Данный сайт является федеральным органом исполнительной власти". Я-то по наивности думал, что у нас люди управляют государством, а оказывается — сайты! Дальше на полстраницы я читаю формальные обязанности министерства, знакомые по газетам и постановлениям (кое-где в случайных местах добавлено слово "сайт").

• Сайту Департамента по молодежной политике, воспитанию и социальной защите детей в том же ключе выделено три с половиной страницы. Текст чудовищен в своей неструктурированности и перегруженности странными терминами типа "образовательные учреждения VIII вида", 16 раз использует автор в этом тексте слово "образовательный" в разных падежах, по 6—8 раз слова "государственный" и "федеральный". Чуть-чуть нормального текста — в последних 15 строках.

• ***www.ntf.ru*** — сайт НФПК, одной из крупнейших сегодня гранторазпорядительных организаций, содержит, по мнению автора, не списки грантов, бланки анкет и отчетов по ним, не информацию о полученных и выполненных по грантам работ. А что? А вот, оказывается, что: "Материалы и ссылки на них, которые содействуют выводу российского образования на уровень современных требований". Я-то по старинке думал, что этому способствует в первую очередь работа учителей! Нет, оказывается, "материалы и ссылки на них" в НФПК.

Дальше интересно быть перестало. Книгу НИКОМУ ПОКАЗЫВАТЬ НЕЛЬЗЯ! Краткое изучение оставшегося показало многочисленные и странные повторения (типа наличия в одном и том же перечне под разными именами ***www.techno.edu.ru*** или ***www.students.ru*** и ***students.ru*** под одним названием — я не проверял, но убежден, что это один сайт на самом деле. Кстати, если книга для школ, то, может быть, его лучше было бы не упоминать ни разу, а не делать это дважды!).

Все же стало интересно — а что у нас с сайтами для учителя, для школьных предметников. Я же знаю, что есть сайты газет и журналов, издательств, олимпиад, университетов и хорошие методические коллекции по предметам. Нет, не нашел я в этой книге не только "Кванта" или "Первого сентября" (у обоих изданий есть хорошие сайты), но и сайтов издательств "Просвещение", "Дрофа", "Физматлит", на которые не раз доводилось заходить. Единственный сайт, непосредственно адресованный предметникам естественно-научного цикла (мне лично оно ближе),— "Естественно-научный образовательный портал" ***www.en.edu.ru***.

Я просмотрел только математику. Более случайного набора неотобранных, неструктурированных, путаных и неточных ссылок мне давно не приходилось видеть. Навигация очень специальна, дизайн отсутствует, наполнение чужое. Деления на школу и институт нет. Подписи к ссылкам, простите, идиотичны. Правда, тут мне как раз попались на глаза и "Квант", и "Первое сентября". Первый был упомянут дважды в одном из списков, в первом случае была неверна и ссылка, и описание, во втором — автором журнала объявлен его нынешний главный редактор Ю.А. Осипьян, а не тысячи лучших авторов популярных и научных статей по физике и математике, которые опубликовали свои статьи в "Кванте" с января 1970 года по нынешнее время (а не с 1975 года, как ошибочно указано в описании на сайте ***www.en.edu.ru***). Рядом в том же списке — ссылки на отдельные брошюры и статьи без указания их авторов (их найти можно только уже внутри описания ресурса). В общем, барахолка большая, а уровень систематизации — на уровне плохого пятиклассника, которого не учили ни дома, ни в школе.

Можно было бы продолжать. Можно было бы просмотреть другие ресурсы. (Наверняка среди них есть достойные. Упомянуты, например, известные и хорошие сайты ряда крупных и известных библиотек.) Только не хочется. Не верю я в грамотность и аккуратность этого отбора, в его заинтересованность, профессиональность и непредвзятость. Кстати, что там с повышением квалификации учителя? Ведь если неквалифицированно написана книга, то, может быть, можно поговорить с квалифицированными специалистами? Нет, не нашел я никаких курсов, а ведь в Москве они есть и сайты для того есть (МИОО, например, полностью публикует информацию о своих курсах — ***www.mioo.ru***). И уж чего проще для такой книги — взять 5—10 лидеров по посещаемости (ведь это можно и по статистике ИАЦ посмотреть, и на ***www.rambler.ru***), и написать по ним аннотированный комментарий. Я могу понять, почему в официальную книгу не попала библиотека Мошкова (хотя с минимальными замечаниями, по-моему, можно и нужно!), но совсем не могу понять, чем провинились многочисленные сайты по олимпиадам для школьников, например, ***olimpiada.ru*** или ***olympiads.ru***.

И последнее, судя по тиражу (напомню, две тысячи) книга предназначена для распространения по школам города. А вот это уже будет явная диверсия. Хорошие и грамотные учителя просто выкинут книгу через 5—7 минут. А вот посредственные могут начать думать, что Интернет им ровно для этих сайтов (извините, порталов) и только для них и дан. Что ничего полезного для школы в русском Интернете нету, что сеть можно использовать только для игры, а по делу — только так.

Я знаю, что Любовь Петровна Кезина или Юрий Михайлович Лужков много внимания уделяют развитию образовательного Интернета, но лично сами и не должны в деталях понимать работу сети, а вот от их советников хотелось бы видеть бóльшую ответственность при "методической поддержке лиц, работающих в сфере управления образованием, а также преподавателей, учащихся и студентов".

Виталий ДМИТРИЕВ

О разных экзаменах по информатике и о том, как с ними бороться

Л.С. Великович,
Москва

Школьная информатика, пожалуй, как ни один другой предмет "приближена" к реальной жизни. Этот очевидный фактор порождает различные последствия, самым распространенным из которых является обучение чисто пользовательским навыкам работы с компьютером (так называемый "кнопочно-мышечный" подход).

Мне уже не раз приходилось писать о, к сожалению, имеющей место устойчивой тенденции подмены научной составляющей предмета технологической. Нельзя ставить знак равенства между изучением информатики и изучением инструментария компьютерных технологий, поскольку второе является лишь малым подмножеством первого.

Мне могут возразить — это же и так очевидно... Согласен, но посмотрите на реалии, и вы поймете мою обеспокоенность. Выдающийся, на мой взгляд, немецкий ученый Манфред Брой писал: "Информатика — наука молодая, но, несмотря на это, она уже широко используется в различных приложениях. Это одновременно и удача, и беда для информатики. Удача потому, что иначе на информатику вряд ли могли бы оказывать плодотворное влияние импульсы из приложений. Беда потому, что слишком быстро распространяются незрелые концепции, от которых впоследствии бывает очень трудно избавиться.

Поэтому в информатике больше, чем в других областях науки, справедлив принцип постоянного критического переосмысления содержания и осознания ограниченности приобретенных знаний" (*Брой М.* Информатика / Диалог-МИФИ, 1996, с. 8). Наличие пользовательских навыков работы с самыми распространенными программными продуктами, такими, как текстовые редакторы, электронные таблицы, графические пакеты и т.п., безусловно, помимо всего прочего, является важным аспектом социальной адаптации учащихся к "взрослой" жизни, будь то учеба в вузе или работа; да и просто становится непременным атрибутом культуры современного человека.

Кстати, тут можно отметить еще одну интересную тенденцию, суть которой состоит в том, что чисто технологические, казалось бы, навыки переходят в область культурологическую. Так было со средствами связи, с телевидением и т.д. И, видимо, этот процесс носит вполне объективный характер.

Однако проблема представляется более широкой. Реальность предъявляет новые требования — происходит внедрение ЕГЭ, многие вузы выносят информатику на вступительные экзамены (только в Москве таких уже более 10, причем среди них такие солидные, как МЭСИ, ГУУ, МИИГАиК и др.).

Если внимательно посмотреть на содержание тестов ЕГЭ или вариантов вступительных экзаменов в вузы, то можно заметить, что "технологический" аспект там минимален, а во втором случае практически отсутствует, зато вопросы по элементам алгоритмического программирования, которые так активно вытесняются из школьной информатики, в билетах вступительных экзаменов присутствуют практически всегда.

В результате уровень знаний учащихся даже тех школ, где в изучении предмета нет полной "замкнутости" на технологический инструментарий, недостаточен для решения некоторых весьма простых задач. Таким образом, если "выход" школьной информатики еще как-то "стыкуется" с содержанием тестов ЕГЭ, то, если сопоставить его с "входом" вузовским, мы видим "нестыковку" весьма серьезную, как ни в одном другом предмете школьной программы.

Эту "нестыковку", как представляется, можно условно разделить на две части. К первой можно отнести вопросы, связанные с разделами "Системы счисления" и "Алгебра логики", ко второй — темы "Алгоритмы" и "Основы программирования". В данной статье остановлюсь на первой части, хотя есть и вполне конкретные соображения по второй.

По моему опыту, проблема может быть успешно решена путем использования тестов из ЕГЭ и задач из вариантов вступительных экзаменов по информатике в вузы непосредственно на занятиях, в том числе и на этапе контроля уровня знаний.

Приведу примеры двух контрольных заданий по системам счисления и по алгебре логики, которые используются мной на занятиях в 10-м классе. Первые пять заданий в каждом взяты из тестов ЕГЭ, вторые пять — из примерных вариантов вступительного экзамена по информатике в МЭСИ.

Задание по теме "Системы счисления"

1. Считая, что каждый символ кодируется одним байтом, оцените информационный объем следующего предложения из пушкинского четверостишия:

Певец-Давид был ростом мал, Но повалил же Голиафа!

1) 400 бит;
2) 50 бит;

3) 400 байт;
4) 5 байт.

2. Сколько существует различных последовательностей из символов "плюс" и "минус" длиной ровно в пять символов?
1) 64;
2) 50;
3) 32;
4) 20.

3. Обычный дорожный светофор без дополнительных секций подает шесть видов сигналов (непрерывные красный, желтый и зеленый, мигающие желтый и зеленый, красный и желтый одновременно). Электронное устройство управления светофором последовательно воспроизводит записанные сигналы. Подряд записано 100 сигналов светофора. В байтах данный информационный объем составляет
1) 37;
2) 38;
3) 50;
4) 100.

4. Как представлено число 83_{10} в двоичной системе счисления?
1) 1001011_2;
2) 1100101_2;
3) 1010011_2;
4) 101001_2.

5. Вычислите сумму двоичных чисел x и y, если $x = 1010101_2$, $y = 1010011_2$.
1) 10100010_2;
2) 10101000_2;
3) 10100100_2;
4) 10111000_2.

6. Десятичное число 10,2 перевели в восьмеричную систему счисления. Определить 1998-ю цифру после запятой.

7. В учебном центре имеются персональные компьютеры двух типов, всего 214 компьютеров, из них 120 первого типа и 44 второго. В какой системе счисления записаны числа?

8. Найти сумму всех целых 16-ричных чисел в диапазоне от A до AA, включая границы. Ответ записать в восьмеричной системе счисления.

9. Перевести число 2A,B из шестнадцатеричной в четверичную систему счисления.

10. Перевести число 123,7030125 из десятичной в восьмеричную систему счисления.

Задание по алгебре логики

1. Для какого числа X истинно высказывание
$$X > 1 \wedge ((X < 5) \rightarrow (X < 3))?$$
1) 1;
2) 2;
3) 3;
4) 4.

2. Укажите, какое логическое выражение равносильно выражению $\neg(\neg A \wedge B)$.
1) $A \vee \neg B$;
2) $\neg A \vee B$;
3) $B \wedge \neg A$;
4) $A \wedge \neg B$.

3. Символом F обозначено одно из указанных ниже логических выражений от трех аргументов: X, Y, Z.
Дан фрагмент таблицы истинности выражения F:

X	Y	Z	F
0	0	0	1
0	0	1	0
0	1	0	0

Чему равно F?
1) $X \wedge Y \wedge Z$;
2) $\neg X \vee \neg Y \vee Z$;
3) $X \wedge Y \wedge \neg Z$;
4) $\neg X \wedge \neg Y \wedge \neg Z$.

4. Для 5 букв латинского алфавита заданы их двоичные коды (для некоторых букв — из двух бит, для некоторых — из трех). Эти коды представлены в таблице:

A	*B*	*C*	*D*	*E*
000	01	100	10	011

Определить, какой набор букв закодирован двоичной строкой 0110100011000.
1) *EBCEA*;
2) *BDDEA*;
3) *BDCEA*;
4) *EBAEA*.

5. Для составления цепочек используются бусины, помеченные буквами: *A*, *B*, *C*, *D*, *E*. На первом месте в цепочке стоит одна из бусин *A*, *C*, *E*. На втором — любая гласная, если первая буква согласная, и любая согласная, если первая гласная. На третьем месте — одна из бусин *C*, *D*, *E*, не стоящая в цепочке на первом месте. Какая из перечисленных цепочек создана по этому правилу?
1) *CBE*;
2) *ADD*;
3) *ECE*;
4) *EAD*.

6. Собираясь в поход, ребята высказали следующие суждения:

а) Если будет холодно или пойдет дождь, поход не состоится;

б) Если будет холодно, то пойдет дождь;

в) Если не будет холодно и поход состоится, то дождя не будет.

Учитель сказал, что эти рассуждения можно свести к двум простейшим высказываниям. Сформулируйте их, решив задачу с помощью логических операций.

7. Упростить логическую функцию:

$$(A + C \to B)(A \to C)(\overline{B} \to \overline{AC})$$

$$(\overline{A} \to (C \to B))(B \to \overline{C} \to \overline{A})$$

8. Составить таблицу истинности для логической функции. Определить ее тип.

$$((A \leftrightarrow \overline{BC}) \to \overline{C}) \to (\overline{A} + B \leftrightarrow C)$$

9. Является ли тождественно-истинной данная формула:

$$(A \to C) \to (B \to C) \to ((B + A) \to C)?$$

10. Упростить логическую функцию

$$(A \to (B + C)) \leftrightarrow ((A \to B) + \overline{(A + C)}).$$

Упрощенный вид должен содержать не более двух логических операций.

Внимательный читатель заметит, что в примерах алгебры логики используются различные обозначения одних и тех же логических операций. Это сделано сознательно, поскольку в различных источниках приняты различные обозначения. В приведенных заданиях первые пять (ЕГЭ) значительно легче последних (МЭСИ) и, как правило, не вызывают трудностей у учащихся. Это тоже дает свой эффект. Учащиеся учатся ранжировать задания по степени сложности и распределять свое время соответствующим образом. Следует отметить, что на выполнение каждого такого задания я отвожу два урока и применяются они в 10-м классе школы физико-математического профиля, что, естественно, не означает невозможность использования такого подхода в иных школах. Все дело в "дозировке". Убежден, что решение вариантов, предлагаемых на ЕГЭ (примеры 1—5), должно быть посильно для любого школьника, изучающего предмет. Что же касается задач второй группы (6—10), то при наличии хотя бы двух часов в неделю информатики задача представляется решаемой. Естественно, что для этого потребуется определенное переосмысление содержательного аспекта. Хочу отметить, что, как показывает опыт, "жесткая" привязка к какому-либо учебнику информатики не обеспечит достижение данной цели, тем более что из 13 рассмотренных учебников информатики ни один не прошел экспертизу РАН (*http://echo.msk.ru/news/272796.html*).

Конечно же возможности школьной информатики как средства социальной адаптации учащихся далеко не исчерпываются приведенным, и я могу рассказать еще многое об этом. В частности, о наработанных подходах по изучению алгоритмов и программирования. Это большая, непростая, но в то же время очень актуальная, можно сказать, "животрепещущая" тема, и если моим уважаемым коллегам этот разговор покажется интересным, его можно продолжить.

В СЛЕДУЮЩЕМ НОМЕРЕ БУДУТ ПОДВЕДЕНЫ ИТОГИ ВТОРОГО ТУРА ("СОБЛЮДАЕМ ТЕХНИКУ БЕЗОПАСНОСТИ") КОНКУРСА "КАК ЭТО ДЕЛАЮ Я"

Для тех, кто не являлся нашим подписчиком в прошлом году, поясним. В 2005/2006 учебном году газета "Информатика" проводит методический конкурс для учителей информатики "Как это делаю я", **основная цель** которого — обмен личным профессиональным опытом решения повседневных методических задач. **Ключевая особенность** нашего конкурса заключается в том, что и задания, и форма предоставления конкурсных материалов фиксированы, что позволяет с достаточно высокой степенью объективности сравнить материалы участников. Кроме того, необходимо отметить, что соревновательная составляющая нашего конкурса, конечно, является вторичной. Мы уверены, что всем нам будет просто интересно и полезно обменяться опытом решения типичных проблем, которые возникают перед каждым из нас. Присланные материалы размещаются в разделе конкурса на сайте "Информатики" *http://inf.1september.ru*. ***Участвовать можно в любом количестве туров: каждый выбирает наиболее близкие и интересные для себя темы.***

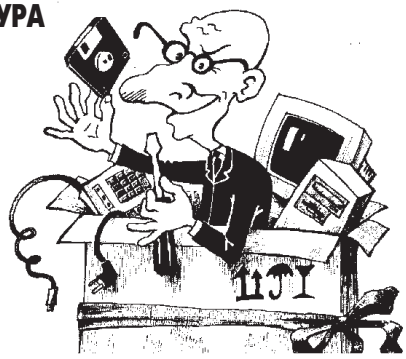


Иллюстрация Д.А. Бурусова из работы К.В. Чернобабовой

Тема четвертого тура (работы принимаются до 31.01.2006) — "Домашние задания". В отличие от большинства школьных предметов, для которых к каждому учебнику имеются различные сборники разноуровневых домашних заданий, мы таким сервисом не избалованы. В типичной ситуации каждый учитель имеет собственную копилку домашних работ. Поделитесь с коллегами собственным опытом организации системы домашних заданий!

VBA в приложении к Excel, Word и Power Point

О.А. ЖИТКОВА, Т.И. ПАНФИЛОВА,
Москва

В № 1—10/2006 вниманию читателей будет предложен цикл статей (разработок занятий), посвященных использованию VBA (Visual Basic for Application) пакета Microsoft Office. Все материалы прошли многолетнюю апробацию в НОУ "Ломоносовская школа" и могут быть использованы как на уроках, так и во внеурочной деятельности — на факультативах, кружках, при реализации учебных проектов.

Занятие № 1. Основные понятия языка VBA Excel

На занятии рассматриваются следующие вопросы:
- общее понятие о языке VBA;
- объекты VBA;
- свойства, методы и события;
- элементы языка VBA;
- структура редактора VBA.

VBA относится к языкам объектно-ориентированного программирования, поэтому знакомство с ним естественно начать с понятия объекта.

Объекты

Объект — основной элемент VBA Excel. В VBA *объектами* являются рабочая книга, рабочий лист и его составляющие.

Примеры объектов:
- `Sheet` — лист Excel;
- `Cell` — ячейка;
- `Range` — диапазон ячеек;
- `Application` — приложение;
- `UserForm` — пользовательская форма.

Доступ к объекту возможен через его методы и свойства.

Методы

Над объектами можно совершать различные действия. Действия, совершаемые над объектами, называются методами. Например, ячейку можно очистить (`Clear`), приложение закрыть (`Quit`), пользовательскую форму показать (`Show`) или скрыть (`Hide`).

Название метода отделяется от названия объекта точкой: ***объект.метод***.

Примеры использования методов:

`Range("B2:E2").Select` — выбрать диапазон ячеек B2:E2;
`Range("C1:C5").Clear` — очистить диапазон ячеек C1:C5;
`UserForm2.Hide` — скрыть форму № 2;
`UserForm5.Show` — показать форму № 5;
`Application.Quit` — выйти из приложения.

Свойства

Свойствами описываются характеристики объектов. Например, размер и цвет шрифта, положение формы на экране или состояние объекта (доступность, видимость). Чтобы изменить характеристику объекта, надо просто изменить значение свойства, т.е. присвоить ему определенные значения.

Синтаксис установки значения свойства:

```
Объект.Свойство = ЗначениеСвойства,
```

— где `Объект` обозначает имя объекта,

Свойство — имя свойства, которому присваивается значение.
Имя объекта отделяется от имени свойства точкой.
Примеры свойств:
Range("D1").Value = 2005 — поместить в ячейку D1 значение 2005.
Range("C1:C10").Text = "Информатика" — поместить в диапазон ячеек C1:C10 текст Информатика.
Range("B2").Font.Size = 14 — в ячейке B2 установить размер шрифта 14.

Методы и свойства некоторых объектов VBA

Объект: Application (*приложение*).
Метод: Quit (*закрыть*).
Свойство: Caption (*имя главного окна*).
Примеры. Application.Quit — закрыть приложение; Application.Caption = "Протокол" — установить в качестве заголовка окна приложения "Протокол".

Объект: Sheet (*лист*), ActiveSheet (*активный/выбранный лист*).
Семейство: Sheets (*листы*).
Методы: Select (*выбрать*); ShowDataForm (*показать встроенную форму*).
Примеры: Sheets("Меню").Select — выбрать лист "Меню"; ActiveSheet.ShowDataForm — на активном в настоящий момент листе показать встроенную форму.

Объект: Range (*диапазон*).
Методы: Select (*выделить*); Clear (*очистить*).
Свойство: Name (*имя*).
Примеры. Sheets("Протокол").Range("B4:B10").Name = "Класс" — диапазону B4:B10, расположенному на листе "Протокол", присвоить имя "Класс"; Sheets("Протокол").Range("B4:B10").Select — выделить диапазон B4:B10 на листе "Протокол".

Объект, Семейство: UserForm (*пользовательская форма*).
Методы: Show (*показать*); Hide (*скрыть*).
Свойство: Caption (*текст, отображаемый в строке заголовка*).
Примеры. UserForm1.Show — показать пользовательскую форму номер один;
UserForm1.Hide — скрыть пользовательскую форму номер один;
UserForm1.Caption – "Информатика"— вывести заданный в кавычках текст в строке заголовка.

Объект, Семейство: TextBox (*Поле ввода*).
Свойство: Text (*содержимое*).
Примеры. UserForm1.TextBox1.Text = Date — в поле ввода номер один в пользовательской форме номер один записать текущую дату; UserForm1.TextBox2.Text = " " — очистить поле ввода номер два в пользовательской форме номер один.

Объект, Семейство: ComboBox (*поле со списком*).
Метод: AddItem (*добавить элемент в список*).
Свойства: Text (*содержимое*); Rowsource (*источник строк для списка*).
Примеры. UserForm1.ComboBox2.Text = " " — очистить значение поля для поля ввода со списком номер два в пользовательской форме номер один; UserForm2.ComboBox1.Rowsource="B2:B10" — источником строк для поля один со списком в пользовательской форме два установить данные из диапазона B2:B10;
UserForm1.ComboBox1.AddItem ("Факс") — добавить к списку элемент, заключенный в кавычки.

Объект, Семейство: OptionButton.
Свойства: Value (*значение*); Caption (*надпись*).
Примеры. UserForm3.OptionButton1.Value = True — выбрать переключатель номер один в пользовательской форме номер три; UserForm3.OptionButton1.Capture = "Успеваемость" — установить надпись "Успеваемость" рядом с переключателем в пользовательской форме номер три.
Объект, Семейство: CheckBox.
Свойства: Value (*значение*); Caption (*надпись*).
Примеры. UserForm2.CheckBox1.Value = True — установить флажок номер один в пользовательской форме номер два; UserForm3.CheckBox1.Value = False — сбросить флажок номер один в пользователь-

ской форме номер три; `UserForm4.CheckBox1.Capture="Класс"` — установить надпись "Класс" рядом с флажком в пользовательской форме номер четыре.

События

Событие представляет собой действие, распознаваемое объектом (например, щелчок мышью или нажатие клавиши, перемещение мыши или выход из программы), для которого можно запрограммировать отклик, т.е. реакцию объекта на произошедшее событие.

В языке VBA для каждого объекта определен набор стандартных событий.

Стандартное событие для объекта "кнопка" (`CommandButton`) — щелчок мышью (**Click**).

Если пользователь нажимает на кнопку, то это событие. На это событие должен быть отклик, т.е. выполнение какой-либо программы. Такая программа называется процедурой обработки событий и имеет стандартное имя. Если такой отклик не создан (не написана соответствующая программа), то система никак не будет реагировать на это событие.

Элементы языка VBA

Объекты — основные элементы языка VBA, но не единственные. К другим элементам относятся: константы, переменные, массивы, выражения, встроенные функции, встроенные диалоговые окна, операторы.

Константы — данные, не изменяющиеся в процессе решения задачи. Константы бывают двух видов: числовые и символьные.

Числовые константы — это целые либо вещественные числа.

Символьные константы — текст, заключенный в кавычки. Пример числовой константы — 5,8 (использование запятой или точки зависит от настроек операционной системы). Пример символьной константы — ООО "Темп".

Переменные — данные, значения которых меняются в ходе выполнения программы. Для переменной задается имя и тип.

Имя переменной должно начинаться с буквы и может содержать любую комбинацию букв, цифр и символов за исключением точек, пробелов и следующих символов: "!", "%", "&", "$", "#", "@". Длина имени не должна превышать 255 символов. Не следует использовать имена, совпадающие с ключевыми словами VBA и именами встроенных функций и процедур.

Основные типы переменных, их размеры и диапазоны принимаемых значений приведены в табл. 1 на с. 11.

Переменные типа `Variant` могут хранить все, что в них поместят. Их тип изменяется в зависимости от последнего присвоения. В программах переменные описываются с помощью специального оператора `Dim`.

Массив — упорядоченная совокупность однотипных переменных. Массивы имеют имя и размерность. Имя массива подбирается с учетом тех же правил, что и имена переменных. **Размерность** — это количество элементов (переменных), составляющих массив.

Из констант, переменных и встроенных функций (они рассмотрены далее) с помощью скобок и знаков арифметических операций ("+", "—", "*", "/", "^") можно составлять **выражения**. Частным случаем выражения может быть просто одиночный элемент, т.е. константа, переменная или обращение к встроенной функции.

Примеры выражений:

- `Z`
- `45`
- `(a+b)^2`
- `sin(y)`

В VBA имеется большой набор **встроенных функций**, которые разделяют на категории. Примеры категорий:

- математические функции;
- логические функции;
- функции проверки типов;
- функции преобразования форматов;
- функции времени и даты.

Некоторые из функций приведены в табл. 2:

Таблица 1

Тип	Размер (байт)	Диапазон значений
Byte (байт)	1	От 0 до 255
Boolean (логический)	2	True или False
Integer (целое число)	2	От −32 768 до 32 767
Long (длинное целое число)	4	От −2 147 483 648 до 2 147 483 647
Single (число с плавающей запятой обычной точности)	4	Для отрицательных значений от −3,4E38 до −1,4E−45. Для положительных — от 1,4E−45 до 3,4E38
Double (число с плавающей запятой двойной точности)	8	Для отрицательных от −1,79E308 до −4,94E−342. Для положительных — от 4,94E−324 до 1,79E308
Currency (денежный)	8	От 922 337 203 685 477,5808 до 922 337 203 685 477,5807
Date (дата и время)	8	От 1 января 100 г. до 31 декабря 9999 г.
String (строка)	10+длина строки	От 0 до 2×10^9
Variant (вариант)		Зависит от содержимого переменной

Таблица 2

Таблица "Функции VBA"

Функция	Выражение
	Математические функции
Abs(x)	Модуль (абсолютная величина числа)
Cos(x)	Косинус
Exp(x)	Экспонента, т.е. результат возведения основания натурального логарифма в указанную степень
Log(x)	Натуральный логарифм
Sin(x)	Синус
Sqr(x)	Квадратный корень
	Логические функции
x1 AND x2	True, если *x*1 и *x*2 равны True; False, если либо *x*1, либо *x*2 равно False
x1 OR x2	True, если хотя бы один из *x*1, *x*2 равен True; False, если и *x*1, и *x*2 равны False
	Функции проверки типов
IsNumeric(*переменная*)	True, если переменная — число False, если переменная — не число
IsNull(*переменная*)	True, если переменная является пустым значением (Null) False, если переменная — не пустое значение
	Функции преобразования форматов
CInt(x)	*x* преобразуется в целое
CSng(x)	*x* преобразуется в число с плавающей точкой обычной точности
CStr(x)	*x* преобразуется в строку
	Функции времени и даты
Date	Текущая дата
Now	Текущие дата и время

Кроме перечисленных функций, объект `Applications` позволяет вызвать более 400 встроенных функций рабочего листа при помощи конструкции вида:

`Application.Функция Рабочего Листа(Аргументы).`

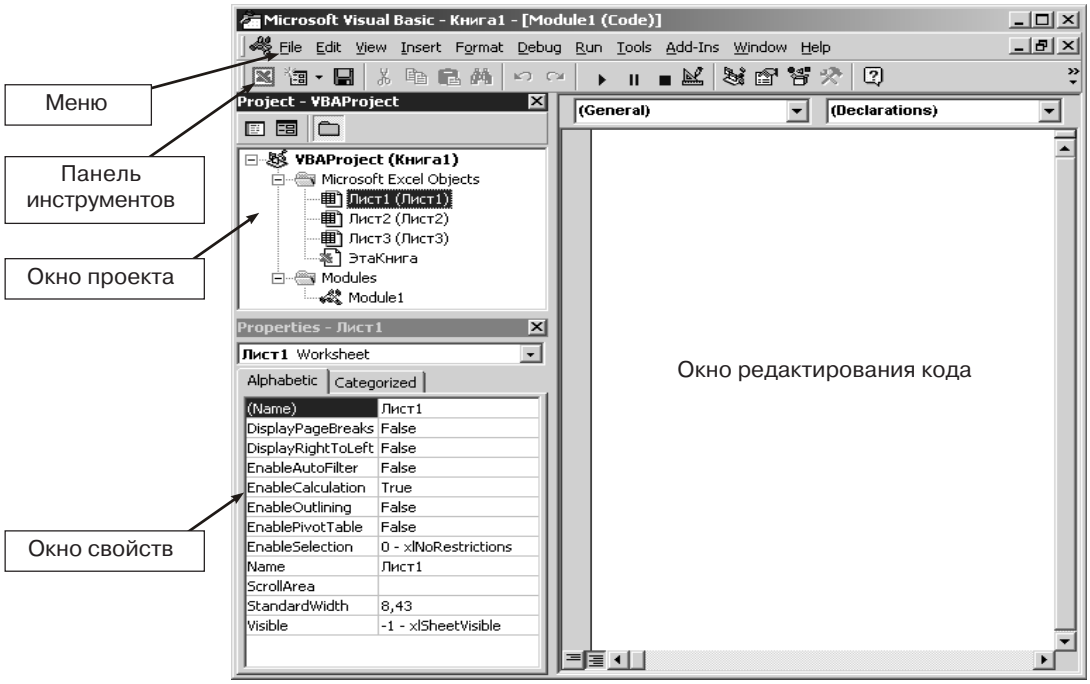
Примеры:

- `Application.Sum(Sheets("Проверка").Range("A1:B20"))` — суммируются значения из ячеек диапазона A1:B20, расположенного на листе "Проверка";
- `Application.CountA(Sheets("Ученики").Range("A:A"))` — подсчитывается количество непустых ячеек в столбце А на листе "Ученики".

Структура редактора VBA

Редактор VBA активизируется командой **Сервис → Макрос → Редактор Visual Basic**. После выполнения команд мы попадаем в редактор VBA. Возвратиться из редактора VBA в рабочую книгу можно нажатием кнопки **Вид(View) → Microsoft Excel**  . Рассмотрим основные компоненты окна редактора. Внешний вид окна редактора представлен на рисунке. Окно редактора состоит из следующих компонентов:

- панели инструментов;
- окно проекта;
- окно свойств;
- окно редактирования кода;
- меню.



Панели инструментов

Стандартная панель инструментов редактора Visual Basic содержит кнопки, предназначенные для выполнения наиболее часто используемых команд.

Панель разбита на отдельные сегменты по типу выполняемых команд.

Кнопки первого сегмента:

Кнопки для возврата в Excel, вставки элементов (модулей, процедур, экранных форм) в проект и сохранение рабочей книги.

Кнопки второго сегмента:

Кнопки для вырезания, копирования, вставки и поиска фрагментов кода.

Кнопки третьего сегмента:

Кнопки отмены действий и повторения отмененных действий.

Кнопки четвертого сегмента:

Кнопки для выполнения, остановки и прекращения выполнения процедуры, а также смены режима отображения экранной формы.

Кнопки пятого сегмента:

Кнопки, управляющие отображением окон проектов, свойств и просмотра объектов, а также панели инструментов.

Последняя кнопка — это обычная кнопка вызова справочной системы.

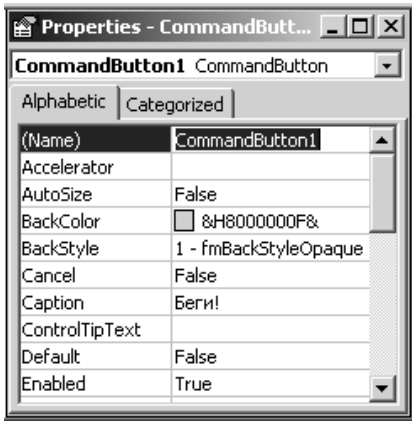
Окно проекта

Окно проекта активизируется выбором команды **Вид → Окно проекта** (View, Project window) или нажатием кнопки **Окно проекта**  . В окне проекта представлена иерархическая структура файлов, форм и модулей текущего проекта.

Окно свойств

В окне свойств перечисляются основные значения свойств выбранного объекта. Используя это окно, можно просматривать свойства и изменить их значения. Для просмотра свойств выбранного объекта надо выполнить команду **Вид → Окно свойств** (Properties Windows) или нажатием кнопки  .

Окно свойств состоит из двух составных частей: *верхней* и *рабочей*. В верхней части окна располагается раскрывающийся список, из которого можно выбрать любой элемент управления текущей формы или саму форму. Рабочая часть состоит из двух вкладок, в которых свойства располагаются **По алфавиту** (Alphabetic) и **По категориям** (Categorized).



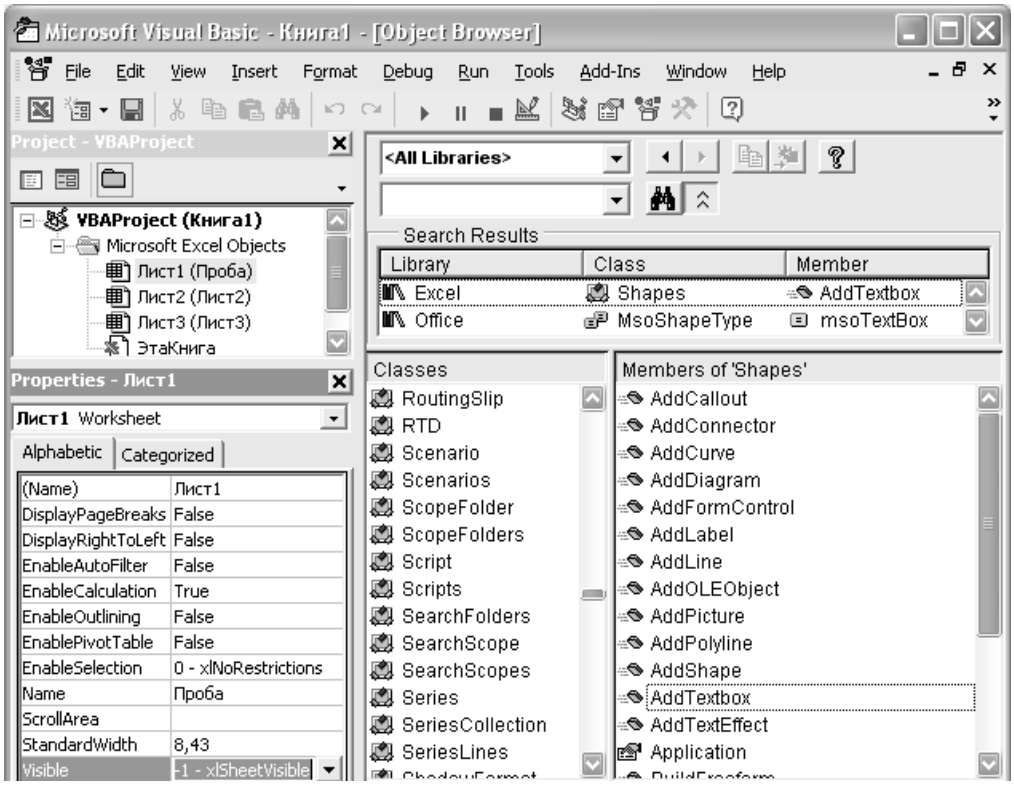
Окно для просмотра объектов (Object Browser)

Окно **Просмотр объектов** (Object Browser) вызывается командой **Вид → Просмотр объектов** (View, Object Browser) или нажатием кнопки  . В этом окне можно просматривать все объекты проекта. Здесь вы найдете все свойства, методы и события, связанные с любым объектом (см. с. 14).

Окно **Просмотр объектов** состоит из трех основных частей:

1. Раскрывающегося списка **Проект/Библиотека** в верхнем левом углу экрана. Например, библиотеки объектов Excel, VBA, Office и VBAProject (объекты пользовательского проекта).
2. Списка **Классы**. Выводятся все классы выбранной библиотеки.
3. Списка **Компоненты** (Members). Выводятся все компоненты выбранного класса.

Это окно предоставляет доступ ко всем объектам, свойствам, методам и событиям.



Проверочная работа № 1–1

Выберите правильный ответ (возможно несколько правильных ответов):

1. Объектом VBA являются:

✓ рабочая книга;
✓ рабочий лист;
✓ диапазон ячеек;
✓ массивы.

2. Щелчок мышью или нажатие клавиши, перемещение мыши или выход из программы, это действие называется:

откликом;
просмотром;
свойством;
✓ событием.

3. Для того чтобы подсчитать дискриминант в квадратном уравнении, надо его записать следующим образом:

$d = b2 - 4\ ac$
$d = b\hat{}2 - 4ac$
✓ $d = b\hat{}2 - 4*a*c$
$d = b\hat{}2 - 4*ac$

4. Продолжите определение:

Метод — это...

основной элемент языка VBA;
структура редактора VBA;
характеристика объекта;
✓ действия, совершаемые над объектами.

5. Чтобы стандартно изменить характеристику объекта, надо:
   написать новую программу;
   применить другое свойство;
   применить метод;
✓ изменить значение свойства.

6. Запишите конструкции, которые применяются для установки свойств объектов и доступа к их методам:

1. Запишите в ячейку B9 дату своего рождения, в ячейку B10 — свое имя.
2. Установите размер шрифта 16 в ячейках C4:F4, цвет — красный.
3. Покажите форму № 3.
4. Скройте форму № 2.
5. Выберите диапазон ячеек B2:D7.

Проверочная работа № 1–2

Выберите правильный ответ:

1. Очистить диапазон ячеек от A1 до C10:
```
Range("A1;C10").Clear
✓ Range("A1:C10").Clear
Range("A1—C10").Clear
Clear. Range("A1:C10")
```
2. Скрыть форму № 6:
```
UserForm6.Show
UserForm №6.Close
✓ UserForm6.Hide
UserForm №6.Show
```
3. Выйти из приложения:
```
Application.Hide
Application.Close
Application.Select
✓ Application.Quit
```
4. Выделить диапазон ячеек от D6 до E8 на листе “География”:
```
Sheets("География").Range("D6—E8").Show
Sheets "География".Range("D6:E8").Select
Sheets "География".Range("D6—E8").Select
✓ Sheets("География").Range("D6:E8").Select
```
5. Присвоить диапазону ячеек от A4 до A16, расположенных на листе “11 класс”, имя “Информатика”:
```
✓ Sheets("11класс").Range("A4:A16").Name = "Информатика"
Sheets("11класс").Range("A4—A16").Name = "Информатика"
Sheets("Информатика").Range("A4—A16").Name = "11класс"
Sheets("Информатика").Range("A4:A16").Name = "11класс"
```
6. Установить источником строк для поля со списком два в пользовательской форме три данные из диапазона от A12 до A15:
```
UserForm3.ComboBox2.Rowsource = "A12—A15"
UserForm2.ComboBox3.Rowsource = "A12:A15"
✓ UserForm3.ComboBox2.Rowsource = "A12:A15"
UserForm2.ComboBox2.Rowsource = "A12:A15"
```
7. Очистить значение поля для поля ввода со списком номер три в пользовательской форме номер два:
```
UserForm3.ComboBox2.Text = "Очистить"
✓ UserForm2.ComboBox3.Text = " "
UserForm3.ComboBox2.Capture = "0 "
UserForm2.ComboBox3.Text = "—"
```
8. Установить флажок номер пять в пользовательской форме номер два:
```
UserForm5.CheckBox2.Value = True
UserForm2.CheckBox2.Text = False
UserForm2.ComboBox5. Capture = " True "
✓ UserForm2.CheckBox5. Value = True
```

9. Сбросить флажок номер пять в пользовательской форме номер два:

```
UserForm5.CheckBox2. Value = True
✓ UserForm2. CheckBox5. Value = False
UserForm2.ComboBox5. Capture = " True "
UserForm2.CheckBox5. Text = False
```

10. Выбрать переключатель номер два в пользовательской форме номер три:

```
UserForm3.OptionButton2.Value = False
UserForm2. CheckBox3. Value = False
UserForm2.ComboBox5. Capture = True
✓ UserForm3.OptionButton2.Value = True
```

Практическая работа № 1–3

Установите соответствие между функцией и выражением:

Log(x)	Экспонента
Sin(x)	x преобразуется в целое
Sqr(x)	Натуральный логарифм
Abs(x)	Модуль (абсолютная величина числа)
Cos(x)	x преобразуется в число с плавающей точкой обычной точности
Exp(x)	Синус
CInt(x)	x преобразуется в строку
CSng(x)	Текущие дата и время
CStr(x)	Квадратный корень
Now	Косинус

Практическая работа № 1–4

1. Откройте новый файл.
2. Перейдите в редактор Visual Basic.
3. Нажмите клавишу F2 для открытия окна Object Browser.
4. В списке Classes (Классы) найдите объект Range и выделите его.
5. Прокрутите соседний список Members of 'Range' для просмотра свойств и методов объекта Range.
6. Выберите метод Activate.
7. Нажмите клавишу F1. Откроется окно справочной системы Visual Basic с описанием метода Activate.
8. Откройте пример с применением метода Activate к объекту Range. Запишите пример.
9. Для объекта Range выберите свойство Cells. Запишите, для каких объектов, кроме объекта Range, характерно это свойство.
10. Запишите метод, используемый для удаления содержимого диапазона ячеек.
11. Запишите метод, используемый для удаления рабочего листа в книге.
12. В окне проекта выберите Лист1. В окне свойств измените его имя на имя ПРОБА. Запишите название свойства.
13. В окне проекта выберите Лист2. В окне свойств подберите свойство, которое будет делать этот лист невидимым. Запишите название свойства и его значение.
14. В окне проекта выберите Лист3. В окне свойств подберите свойство, которое будет изменять ширину столбцов. Укажите значение — 15. Запишите название свойства.
15. Закончите предложение: окно Object Browser можно использовать для просмотра списков объектов, событий, свойств и...

Игра-конкурс “Инфознайка-2005”

Н.В. СОФРОНОВА, А.А. БЕЛЬЧУСОВ, Н.В. БАКШАЕВА,
Республика Чувашия

В Чувашской Республике 8 апреля 2005 г. была проведена игра-конкурс “Инфознайка”, в которой приняло участие более 500 школьников начальной, основной и полной ступеней образования из двенадцати районов (Алатырский, Вурнарский, Козловский, Красночетайский, Мариинско-Посадский, Моргаушский, Порецкий, Урмарский, Шумерлинский, Ядринский, Яльчикский, Янтиковский) и шести городов (Чебоксары, Новочебоксарск, Кугеси, Цивильск, Шумерля, Ядрин). Игра проводилась на трех уровнях:

- подготовительный (1—7-е классы);
- основной (7—11-е классы);
- углубленный (10—11-е классы).

Для каждого уровня были выбраны задачи, учитывающие степень подготовки учащихся в области информатики. Задачи подготовительного уровня рассчитаны на учеников, не изучавших информатику, либо изучавших ее в “безмашинном” варианте, что соответствует пропедевтическому уровню преподавания информатики. Задачи основного уровня рассчитаны на учащихся, изучающих информатику на базовом общеобразовательном уровне. Задания углубленного уровня — на учащихся профильных классов. Ученики самостоятельно выбирали уровень, по которому хотели получить задания. Дифференциация задач составляла от 10 до 50 баллов. Выбравшим подготовительный уровень было необходимо решить 24 задачи, основной — 28, задание углубленного уровня состояло из 30 задач. Для рассылки заданий и получения результатов были использованы как возможности электронной почты, так и бумажные комплекты, включающие бланки заданий и бланки ответов. На выполнение заданий отводилось 45 минут. Проверка полученных результатов осуществлялась с помощью автоматизированной системы, что позволило ускорить процесс оценки работ и подведение итогов. По результатам игры-конкурса 97 участников, набравших в общем зачете более 80% от возможного количества баллов, стали победителями и были награждены призами. Остальные участники получили сертификаты, где были указаны набранные ими очки.

Стоит отметить, что возможность участия школьников в подобных конкурсах значительно шире, чем в олимпиаде по информатике. Специфика олимпиад не позволяет учащимся, интересующимся теоретическими основами информатики, информационными и коммуникационными технологиями, показать свои достижения в этой области. В результате проведения игры-конкурса был получен объективный срез знаний и компетентности учащихся в предметной области “Информатика и ИКТ” по пропедевтическому, базовому и профильному курсу. Ниже приведены данные, отражающие успешность решения заданий в подготовительном, основном, углубленном уровне.

1. Подготовительный уровень

Задания, оцениваемые из 10 баллов

Наибольшее количество баллов учащиеся набрали при решении задач, в формулировке которых использовалась информация, зашифрованная в форме ребуса. Наименьшее количество баллов было получено при решении заданий, подготовленных для практико-ориентированных сред (например, в графическом редакторе MS Paint) и задач, связанных со свойствами информации.

Задания, оцениваемые из 20 баллов

Меньше всего допущено ошибок при решении задач на дешифрование информации, записанной с помощью кода, состоящего из 0 и 1; алгоритмов, связанных с вычисляемым условием и выбором дальнейших действий. Наиболее сложными оказались задания, рассчитанные на использование логических операций И, ИЛИ, НЕ при решении задач.

Задания, оцениваемые из 30 баллов

Разброс по баллам в этой категории задач наименьший, однако процент решений находится в интервале 52—61%, т.е. весьма невысок. Задания были представлены логическими задачами и задачами по расшифровке информации с последующим выбором решения.

Задания, оцениваемые из 40 баллов

Задания этой группы, вызвавшие наибольшее затруднение учащихся (31—53%), предусматривали вычисления в цикле, формирование оптимальных решений в графах, расшифровку сообщений на основе заданной последовательности действий.

2. Основной уровень

Задания, оцениваемые из 10 баллов

Учащиеся ответили правильно (свыше 90%) на вопросы, связанные с устройством компьютера и периферией, с антивирусной защитой и выбором программного обеспечения для подготовки реферативных работ.

Наибольшее затруднение (40—55%) вызвали вопросы, связанные с системой управления базами данных, использованием служб сети Интернет, выбором информационных моделей.

Задания, оцениваемые из 20 баллов

Трудность представляли задания, в которых требовались знания по теории логических высказываний (52%). Точное значение атрибутов графических файлов смогли указать лишь 68% отвечающих.

Задания, оцениваемые из 30 баллов

Общий уровень ответов на вопросы невысок (51—61%), а знания, которые требуют проявления уровня компетентности (оценка возможностей графического растрового редактора для выполнения операций масштабирования, например), показали только 28% учащихся.

Вызвали затруднения и вопросы, связанные с количеством информации, которую можно разместить на внешнем носителе. Только 56% учащихся смогли соотнести размеры файла и внешнего носителя, выбрать соответствующий множитель.

Задания, оцениваемые из 40 и 50 баллов

Те задачи, которые носили характер обобщающих теоретических, не вызвали особых затруднений и получили от 76 до 95% правильных ответов.

3. Углубленный уровень

На углубленном уровне были задания, на которые никто не смог ответить правильно (0%), и задания, по которым были получены 100%-ные результаты. Можно отметить, что учащиеся не знают истории создания компьютеров, не владеют приемами использования логических операций в программировании, не знакомы с возможностями текстового процессора MS Word по автоматизированной обработке данных. Около половины учащихся не смогли упростить сложную логическую функцию и составить логическую схему по заданной функции. Учащиеся определили (58—100%) по фрагментам программ, написанных на различных языках программирования, выполняемые в них действия. А вот для задания, связанного с языком Пролог, не было представлено ни одного правильного решения. Сложной оказалась и задача на кодирование информации последовательностью комбинаций битов (0%).

Ниже приведены примеры заданий, которые вызвали наименьшее и наибольшее затруднение учащихся в соответствующих уровнях.

Подготовительный уровень

A4 (10 баллов)

Количество правильных ответов составило 87% от общего числа выполнявших задание.

Расшифруй предложение, в котором перепутаны буквы в словах:

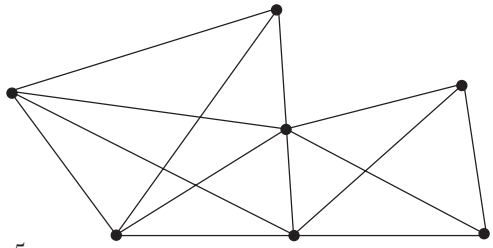
ВЕТИРП КИНСАТИЧУ ЫГРИ.

1) УГРИ ЧАСТО ПЛАВАЮТ
2) ПРИВЕТ УЧАСТНИК ИГРЫ
3) ВЕТРЫ УЧАСТИЛИСЬ С СЕВЕРА

A22 (40 баллов)

Количество правильных ответов составило 21% от общего числа выполнявших задание.

На рисунке зачеркни как можно больше отрезков в изображенном ниже графе так, чтобы в получившемся графе любую точку можно было соединить с любой другой точкой, двигаясь по отрезкам. Сколько отрезков ты зачеркнул?



1) 5;
2) 6;
3) 7;
4) 8.

Основной уровень

A21 (30 баллов)

Количество правильных ответов составило 28% от общего числа выполнявших задание.

В результате последовательного уменьшения (*рис.* 2), а затем увеличения (*рис.* 3) рисунка (*рис.* 1) были получены следующие изображения.

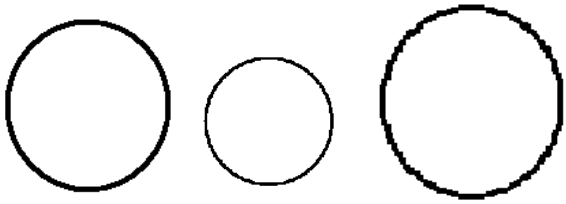


Рис. 1 *Рис.* 2 *Рис.* 3

Какое слово пропущено в предложении?

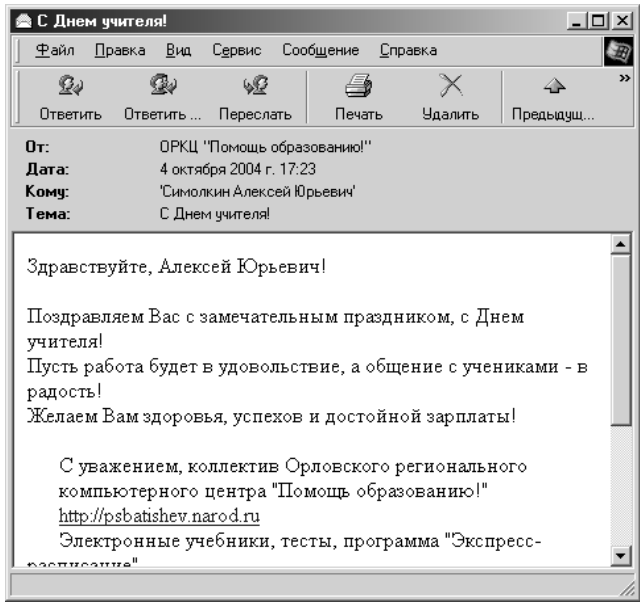
Масштабирование в _____ редакторе не является его сильной стороной.

1) Векторном;
2) графическом;
3) текстовом;
4) растровом.

А17 (20 баллов)

Количество правильных ответов составило 98% от общего числа выполнявших задание.

От кого пришло письмо?



1) от Алексея Юрьевича;
2) от ОРКЦ “Помощь образованию”;
3) от учителя;
4) от неизвестного адресата.

Углубленный уровень

А10 (10 баллов)

Количество правильных ответов составило 100% от общего числа выполнявших задание.

На каком языке программирования написан фрагмент программы?

1) Pascal;
2) QBasic;
3) VBA;
4) Delphi.

```
ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ
ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ
ŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸŸŸŸŸŸŸŸ"
ŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ"ŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸŸ"
ŸŸŸŸŸŸŸŸ
```

А29 (50 баллов)

Количество правильных ответов составило 0% от общего числа выполнявших задание.

Программа написана на языке Пролог.

```
большой(медведь).
большой(слон).
маленький(кот).
коричневый(медведь).
черный(кот).
серый(слон).
темный(Z) :- черный(Z).
темный(Z) :- коричневый(Z).
? темный(X), большой(X) % Кто одновременно темный и большой?
```

Как ответит пролог-система на поставленный вопрос?

1) X = слон;
2) X = медведь;
3) X = кот.

Подводя итоги, можно отметить, что следующие учебные разделы и темы требуют большего внимания в курсе обучения информатики: логические основы работы компьютера, моделирование и формализация, алгоритмизация, автоматизированные системы и устройства, искусственный интеллект.

Олимпиады по осени считают

Осень — напряженное время для студентов и школьников, а для участвующих в соревнованиях по программированию вдвойне, поскольку к обычным учебным заботам добавляются олимпиадные. В октябре–ноябре проходят региональные четверть- и полуфиналы ICPC — командного студенческого чемпионата мира по программированию ACM, школьные олимпиады различного уровня. Завершает различные отборочные соревнования неделя информатики, которая традиционно проводится по приказу Министерства образования и науки в конце ноября одновременно в Санкт-Петербурге и Барнауле.

В рамках недели информатики 27 ноября была проведена VI Всероссийская командная олимпиада школьников по программированию, а 30 ноября прошли полуфинальные соревнования Северо-Восточного Европейского региона студенческого чемпионата мира. В соревнованиях школьников участвовали 106 команд из России и Беларуси. Впервые победителем соревнований стала команда г. Новосибирска, соревновавшаяся в барнаульской подгруппе. В состав команды-победительницы входили:

Владислав Кузькоков (Лицей информационных технологий),

Степан Гатилов (лицей № 130),

Роман Сорокин (лицей НГТУ).

Тренируются ребята вместе со студентами Новосибирского государственного университета под руководством С.Н. Касьяновой, А.Г. Тамаревской, Т.Г. Чуриной, Т.В. Нестеренко, Е.А. Четвертакова. Следует отметить, что месяцем ранее эта же команда одержала победу в обоих турах студенческой Всесибирской командной олимпиады по программированию им. И.В. Поттосина (подробности на сайте *http://olimpic.nsu.ru*).

Второе место уже второй год подряд заняла команда лицея № 30 г. Санкт-Петербурга, тренирующаяся на базе Санкт-Петербургского государственного университета в составе:

Копелеович Сергей,

Петров Юрий,

Руцкий Владимир.

Третье место у команды ФТШ г. Санкт-Петербурга, тренирующейся на базе Санкт-Петербургского университета информационных технологий механики и оптики в составе:

Гладких Максим,

Давыдов Олег,

Сидоров Александр.

За выход в финал студенческого чемпионата мира боролись уже 172 команды из различных стран СНГ (за исключением Украины и республик Прибалтики). Победителем полуфинала стала команда Московского государственного университета им. М.В. Ломоносова в составе:

Попелышев Иван,

Румянцев Андрей,

Халявин Андрей.

Команда-победительница

Им, а также еще 8 командам различных вузов России предстоит 9—13 апреля в г. Сан-Антонио, штат Техас (США), участвовать в финале командного студенческого чемпионата мира по программированию. Пожелаем им достойно представить нашу страну!

О том, как можно достигнуть больших успехов в подобных соревнованиях, можно прочитать в статье чемпиона мира 2004 г. Сергея Оршанского (см. с. 21—26). Его советы могут оказаться полезными и для участников личных олимпиад по информатике, которые стартуют в январе в различных регионах России.

О решении олимпиадных задач по программированию формата ACM ICPC

С.А. Оршанский,
чемпион мира ACM ICPC 2004 г.,
Санкт-Петербург

Интересно читать мнение любых людей, добившихся выдающихся результатов в своей области. Статья С.А. Оршанского не является исключением.

А.А. Шалыто, д. т. н., профессор

1. Олимпиады по программированию

1.1. Соревнования по информатике и программированию

Олимпиады по информатике, как и олимпиады по математике, широко распространены и имеют достаточно долгую историю. Командный студенческий чемпионат мира по программированию ACM ICPC (*Association for Computing Machinery International Collegiate Programming Contest*) [1—4] проводится с 1977 года. Международная олимпиада школьников по информатике IOI (*International Olympiad in Informatics*) проводится с 1989 года. Эти олимпиады позволяют выявлять способности как в математике, так и в программировании, а также умение работать под стрессом в сжатых временных рамках. Указанные соревнования студентов традиционно являются командными, а школьников — личными. В России более долгую историю имеют олимпиады школьников по информатике. В книге [5] собраны все задачи московских олимпиад по программированию, которые прошли с 1980 по 1988 гг. Материалы для подготовки к школьным олимпиадам можно найти также в книгах [6—8].

Популярность соревнований по информатике и программированию стремительно растет. Их спонсорами выступают такие крупные корпорации, как *AT&T*, *Microsoft*, *IBM*, *Google*. Естественно, появились исследования о том, как эффективно участвовать в соревнованиях, готовиться к ним, многочисленные советы и рассказы очевидцев [9]. К этой категории относится и настоящая статья. Автор имеет большой опыт участия в соревнованиях по информатике и программированию, преимущественно — командных студенческих. В настоящей статье речь пойдет о решении задач чемпионата мира ACM ICPC или аналогичных соревнований.

Цель статьи — попытаться ответить на вопрос "Как решить задачу?" при условии, что она одна и решить ее надо достаточно быстро. При этом необходимо решить задачу наверняка, а не с вероятностью 50%. Аспекты командной борьбы, стратегия и тактика не рассматриваются, но некоторые соображения по этим вопросам будут приведены.

В работе излагаются некоторые общие принципы решения задач. В работе также затронут вопрос о минимальном круге идей и методов, которыми целесообразно владеть каждому участнику соревнований. Эти идеи и методы являются базовыми не только для подготовленных участников, но и для составителей задач. За это олимпиады иногда подвергаются критике. Однако в этом соревнования по программированию мало чем отличаются от других сфер человеческой деятельности. Если бы на разных этапах соревнования давали принципиально различные задачи, то отборочные туры потеряли бы смысл. Принципиально изменять характер задач из года в год на всех этапах — четвертьфиналах, полуфиналах и в финале — нереально. В этом и нет необходимости, поскольку неясно, кого в таком случае будет выявлять чемпионат. На сегодняшний день студенческий чемпионат мира отбирает лучших в командном решении задач формата ACM ICPC. На этих соревнованиях команда состоит из трех человек, ей предоставляется один компьютер на пять часов для решения 8—12 задач.

1.2. Об обмене опытом

В книге [4] можно прочитать легендарный текст Леонида Волкова и Никиты Шамгунова "Как стать чемпионом Урала по программированию". Многие участники олимпиад знают наизусть фразу из этого текста: "Я не знаю, как решать задачи. Я знаю только, что после того, как решишь их много, начинаешь делать это лучше, начинаешь лучше видеть возможные подходы к решению задач, начинаешь лучше их чувствовать".

Вероятно, многолетний опыт участия в командных соревнованиях по программированию позволяет мне высказать ряд конструктивных соображений по вопросу "Как решать задачи?". Не отвечая на вопрос "Как стать чемпионом мира по программированию?", я постараюсь сформулировать и объяснить, чем я руководствовался при решении задач. Видимо, такие принципы все-таки существуют, хотя они, к сожалению, и не работают без интуиции [10].

Сама постановка вопроса "Как решать задачи?" может показаться кощунственной, поскольку решение задач — процесс творческий [11]. Однако описание некоторых методов и приемов, помогающих лучше решать задачи, вполне может служить ответом на поставленный вопрос, не предполагая тем не менее погружения в недра сознания. Подобным образом занятия физкультурой являются лишь механизмом, стимулирующим возможности тела, но совсем не обязательно раскрывающим сущность этих возможностей. Еще одно подтверждение вышесказанному — работы Г.С. Альтшуллера по теории решения изобретательских задач (ТРИЗ) [12]. Мало кто усомнится, что решение изобретательских задач — процесс творческий, однако ТРИЗ

активно применяется на практике в самых разных сферах человеческой деятельности. Надо сказать, что многие идеи Г.С. Альтшуллера применимы не только для решения инженерных, но и научных и олимпиадных задач.

В заключение раздела отметим, что обычно в литературе приводятся задачи, а иногда — задачи с решениями [13—16]. Однако методика решения олимпиадных задач описана недостаточно подробно и в основном передается от предыдущего поколения олимпиадников к следующему в устной форме. В настоящей работе делается попытка хотя бы частично заполнить этот пробел.

1.3. Существующие наработки

Начнем с обзора существующих материалов, помогающих научиться решать задачи. Наиболее важную роль играет наличие собственно задач, на которых можно тренироваться. Каждый год в мире происходит огромное количество соревнований по программированию различного уровня. Стали появляться сайты с интерактивными архивами задач. В них не просто доступен текст задачи, как в библиотеках, но и имеется возможность послать решение на проверку и немедленно, почти как на настоящих соревнованиях, получить результат. На таких сайтах ведутся рейтинги участников по числу решенных задач, проводятся онлайн-соревнования, обычно носящие любительский или тренировочный характер.

Два крупнейших российских архива задач:

1. *Saratov State University :: Online Contester*. Онлайн-система тестирования олимпиадных задач Саратовского государственного университета. ***http://acm.sgu.ru/***

2. *Ural State University Problem Set Archive with Online Judge System*. Онлайн-система тестирования олимпиадных задач Уральского государственного университета. ***http://acm.timus.ru***

Как отмечалось выше, существует ряд публикаций [13—16] с разбором конкретных задач всероссийских и международных олимпиад. Также есть публикации, например [17], в которых излагаются конкретные методики, применяемые при решении задач, небольшие хитрости и приемы.

Идеи настоящего текста адресованы читателю, участвовавшему в нескольких соревнованиях, хотя бы тренировочных. Поэтому предполагается, что читатель представляет, что такое олимпиадные задачи рассматриваемого формата, знает, что они проверяются на тестах, что засчитывается только программа, прошедшая все тесты, и т.д. Информацию об этом можно найти в книге [2].

Решением задачи является программа, читающая входные данные и выводящая соответствующие выходные данные. Задача может быть либо решена полностью, либо не решена совсем, тогда как на школьных олимпиадах важно уметь писать программы, которые работают почти всегда или почти на всех наборах входных данных.

На сайтах ***http://acm.sgu.ru/***, ***http://acm.timus.ru*** и ***http://neerc.ifmo.ru/school/*** можно найти ссылки на другие архивы задач и иные материалы схожей тематики.

2. Общая схема решения задачи

Перед вами условие задачи. Эта задача как бы единственная, другой не дано. Поэтому требуется не оценивать ее, а решать, и именно ее. Это предположение делается для простоты, на практике все сложнее. Рассмотрим семь этапов, через которые проходит решение задачи. Конечно, некоторые из них могут пропускаться, смешиваться, распараллеливаться между членами команды, если олимпиада командная, и т.д.

Основное правило: можно пропускать не более одного этапа. Так, переход к третьему этапу ("построение общей схемы решения") должен происходить только после полного завершения первого этапа ("чтение условия"), а, например, переход к седьмому этапу ("посылка решения в жюри") — только после окончательного завершения пятого этапа ("реализация"). То же верно для второго и четвертого, третьего и пятого, четвертого и шестого этапов решения задачи. При этом соседние этапы частично пересекаются. Обязательно обдумайте это правило, прочитав про все семь этапов, — из него следует много важных выводов.

2.1. Чтение условия

На этом этапе необходимо внимательно прочесть условие, не пропуская ни одной фразы. Типичные проблемы:

- в обстановке соревнований сложно быть внимательным. Отведите достаточно времени на спокойное чтение условия. Отдохните полминуты, чтобы сконцентрироваться, но не спешите читать условие "наискосок". Неверное понимание условия может привести к тому, что вы будете решать совершенно другую задачу, а не ту, что сформулирована;
- обычно в условии есть так называемое "литературное введение", придающее задаче сюжет ("background"). Чтение такого описания обычно утомляет, отвлекает, а также расслабляет, так как авторам задач обычно не чуждо чувство юмора. Однако будьте осторожны: во введении может, прямо или косвенно, содержаться важная информация, касающаяся условия. Если "background" не вынесен в отдельный раздел, то его придется прочитать. Обычно это делают не очень внимательно, выискивая начало содержательного текста, который скорее всего пойдет потом без перерыва;
- незнание или плохое знание английского языка может помешать правильному пониманию условий задач, которые даже на российских студенческих олимпиадах формулируются на английском языке. Это делается для тренировки перед финалами олимпиад. Практически на всех официальных соревнованиях разрешено использовать словарь. Не пренебрегайте этой возможностью и переводите те слова, которые критичны для понимания смысла. Если условие плохо понятно в целом, переведите по словарю и те слова, которые на первый взгляд не важны для понимания смысла. Изучайте английский язык на досуге — пригодится;
- ключевое условие может быть спрятано, например, в формате выходных данных. Без этого условия задача может быть совершенно другой, но тоже вполне корректной. Конечно, на серьезных соревнованиях такого обычно не бывает, но все-таки будьте внимательны. Ошибки при чтении условия дорого обходятся.

2.2. Построение математической модели

На этом этапе необходимо понять, в чем заключается задача — построить ее математическую модель "в голове". Не думайте, что невыполнение этого этапа означает неправильное понимание. Можно внимательно прочитать текст, но не построить никакой математической модели. Попытка формализовать прочитанное часто выявляет множество нестыковок, возникших из-за важной фразы, пропущенной при чтении или неверно понятой. Хорошая проверка — внимательно рассмотреть приведенный пример входных и выходных данных и понять, почему выход соответствует входу.

Что значит "построить математическую модель"? Это означает достаточно формально и математически строго понять условие. С моей точки зрения, понять условие — это, как правило, научиться вручную, с помощью ручки и листа бумаги, находить ответ для простых наборов входных данных (тестов). Кстати, полезно не только научиться, но и проделать это для нескольких таких тестов. При этом улучшится понимание условия и, возможно, вскроется неправильное прочтение. Кроме того, могут прийти новые идеи, и в любом случае будут готовы тесты для дальнейшей проверки. Таким образом, для простых задач вы разберетесь, понимаете ли вы их условия, тогда как для сложных задач простых тестов может оказаться недостаточно.

Возможна и другая трактовка того, что такое построить математическую модель. Это значит придумать решение, которое будет работать на абстрактной виртуальной, математической машине при неограниченной памяти, неограниченном времени, неограниченном диапазоне переменных и отсутствии потерь точности в вещественной арифметике.

При этом требуется разработать более или менее формальный алгоритм решения задачи. Это не означает даже решения задачи на уровне идеи. Неэффективное решение задачи оказывается практически равносильным пониманию условия. Например, дана строка и требуется найти подстроку, оптимальную по некоторому параметру. Решение вытекает из способа ручного поиска ответа на маленьких примерах: перебрать все подстроки и выбрать оптимальную. Скорее всего вы не будете реализовывать это решение, но следует его придумать или наметить, или хотя бы почувствовать. Это и означает построить математическую модель в олимпиадном смысле — понять и осмыслить условие задачи.

Иногда условие включает неизвестное для вас понятие. Даже если оно там же и определяется, от вас могут потребоваться значительные усилия по пониманию, что же требуется сделать. Соответственно, чем больше идей и методов вы освоили при подготовке, тем меньше шансов встретить в условии малознакомое понятие. Более того, привыкая к основным идеям, отрабатывая различные приемы, вы будете все чаще встречать знакомые задачи, полностью или с небольшими вариациями совпадающие с задачами, которые вы видели раньше.

2.3. Построение общей схемы решения

Теперь следует перейти от понимания того, что необходимо сделать, к пониманию того, как это сделать. В этом разделе намечается эффективный алгоритм решения задачи и пути его реализации. Это наименее формализуемая часть всего процесса решения. В ней может заключаться вся суть задачи, но она может быть и тривиальной при достаточно сложной задаче в целом. В чем же заключается этот этап?

У каждого участника существует некоторое понятие о "кирпичиках", элементарных структурных единицах создаваемой программы — в нашем случае об алгоритмических единицах. Каждая такая единица характеризуется функциональностью, эффективностью, сложностью написания (количеством кода) и т.д. Различные "кирпичики" имеют разные возможности по модификации. На этом этапе необходимо построить решение из "кирпичиков". Участник, решающий задачу, не обязательно должен хорошо понимать и осознавать данный конкретный "кирпичик". Например, он может знать, что такое *Венгерский алгоритм*, какова его эффективность и насколько долго его писать. Однако он может в данный момент и не помнить деталей, а всего лишь быть уверенным в том, что он эти детали вспомнит, если потребуется, или попросит помощи у своего товарища по команде.

Итак, на этом этапе строится схема решения из "кирпичиков". При этом возникают следующие трудности:

- плохая стыковка. Интуитивная комбинация нескольких сложных блоков, внутренняя структура которых тяжело осознается человеком, может на первый взгляд решать задачу, однако при внимательном рассмотрении будут возникать проблемы. Может выясниться, что, например, у задачи, решаемой динамическим программированием, отсутствует свойство субоптимальности, или что маловероятный на первый взгляд "худший случай" возникает на любом достаточно большом наборе входных данных, или что-нибудь в том же духе. Эти проблемы сродни неправильному решению математической задачи;
- эффективность. Придуманная схема может оказаться неэффективной. При этом необходимо учитывать не только асимптотические оценки, но и игнорируемые в этих оценках константы. Самый надежный выход из этой ситуации — создание принципиально более эффективного решения всей задачи или какой-то подзадачи. Возможная альтернатива: "добивание" данного решения различными алгоритмическими и программистскими оптимизациями, что рискованно — можно потратить кучу времени, но так и не сдать задачу;
- другой тип задачи. Не все задачи решаются построением из "кирпичиков". Решение задачи может базироваться на математических идеях, которые необходимо просто придумать. В этом случае "кирпичиками" могут служить типичные приемы для подобных задач: отсортировать, начать с максимального/минимального элемента, использовать динамическое программирование и т.д.

При решении сложной задачи, к которой никак не подступиться, имеет смысл некоторое время "подолбить" ее стандартными методами. Это может не привести к решению, а может привести и к неправильному решению, внешне похожему на правильное. Однако такой подход оправдан хотя бы потому, что большинство задач только на первый взгляд — сложные. Практически невозможно придумывать к каждому соревнованию 8—12 задач, совершенно новых и непохожих на

задания прошлых лет. Если же такое случается, то часто подобные задачи остаются нерешенными никем;

• наличие нескольких решений. Синтез из "кирпичиков" может помочь быстро придумать решение, которое потом долго и нудно реализуется, тогда как для данной задачи может быть лучше немного подумать, и все существенно упростится.

2.4. Стыковка

Под стыковкой понимается уточнение решений, принятых на предыдущем этапе. Необходимо достаточно медленно и тщательно проговорить, из каких частей будет состоять программа, какие массивы и структуры будут выделены и т.д.

На этом этапе часто всплывают различные проблемы. Наиболее распространенная схема стыковки — один человек придумал общую схему решения задачи и рассказывает ее второму. Это неплохой метод, часто экономящий силы и время, но при этом возникают проблемы. Первая проблема: второй человек должен был сам, заранее и внимательно прочитать условие задачи. Вторая — он должен реально слушать первого, а не только делать вид, как это происходит в половине случаев.

Исключительно полезно писать решение задачи на бумаге. Лучше, чтобы это были не наброски, а большие законченные фрагменты или даже вся программа целиком, включая объявление всех переменных. Полезность написания кода на бумаге не в том, что перепечатывать код с бумаги на компьютер быстрее, чем писать "из головы", хотя это, конечно, и так. В написанном коде проще обнаружить, все ли необходимые переменные и процедуры используются. После этого не понадобится десять раз прокручивать программу на экране в поисках нужного места. Это экономит время. Но это также вторично.

Основная идея написания кода на бумаге в том, что этим форсируется завершение стыковки, происходит упорядочение мыслей. Записать код — значит четко сформулировать решение. Впрочем, мне встречались случаи, когда довольно смутные мысли оформлялись в виде кода, "примерно передающего идею", а потом практически подгонялись под ответ вариациями с индексами массива — заменами i на $i + 1$ и т.п. Иногда это приводит к успеху, но чаще угадать не удается, и требуется сконцентрироваться, додумать и сразу написать верный код. В большинстве случаев приходится "переобдумать" и переписать значительную часть программы — написание недообдуманной программы помогает, уменьшая нагрузку на мозг, но приводит к очень большой потере времени. Впрочем, большой опыт и высокая техника иногда позволяют "на лету" перекроить программу, делающую не то и не так, в правильно и быстро работающую. Фаза стыковки в этом случае может быть исключена вовсе. Описанная ситуация все-таки является "высшим пилотажем", требующим внимания всех трех участников команды или хотя бы двух из них, а также изрядной доли везения. Несмотря на то что я неоднократно участвовал в решении задач указанным образом, обычно лучше все-таки заранее подумать и разложить по полочкам все детали, чем потом спешно сшивать разрозненные куски в подобие решения, почему-то выдающее верные ответы на тесты.

Обычно пропуск стыковки и переход сразу к реализации возникал либо по причине усталости и нежелания сосредоточиться и тщательно продумать решение, либо из-за желания немедленно начать что-нибудь писать, чтобы не терять время и не повышать нервозность обстановки в команде из-за простоя компьютера.

Часто стыковка выявляет ошибки в решении, неэффективность решения и т.д. Она же наводит на мысль, как придумать другое, существенно более простое решение. Это еще один аргумент в пользу того, что десять минут размышления могут в дальнейшем сэкономить полчаса. На этапе стыковки необходимо "раскрыть кирпичики", вспомнить, как же пишутся эти якобы известные стандартные алгоритмы.

Вариант при нехватке времени: один из участников пишет основную часть на компьютере, а второй — стандартный алгоритм на бумаге, а потом "вбивает" его в компьютер. При этом необходимо тщательно согласовать интерфейс — не только для ускорения и упрощения "вбивания", но и в первую очередь для взаимопонимания того, что же собственно требуется. Второй участник, которого просят написать алгоритм на бумаге, в 80% случаев должен спросить: "А зачем в решении этот алгоритм?". Тут первый участник скорее всего начнет мяться и запинаться, и выяснится, что он придумал что-то сложное и громоздкое, а этот алгоритм нужен как подзадача чего-то другого, что решается само по себе значительно проще.

У разных участников соревнований, в том числе и успешных, различное отношение к этапу стыковки. Я считаю этот этап очень важным и настаиваю на его выполнении. Этот этап сложен, требует большой концентрации, но немного времени. Им часто пренебрегают, что при недостаточном опыте и интуиции может привести к печальным результатам. Пренебрегают обычно из-за усталости в конце соревнований или экономии сил в начале. Однако, если только вы не экономите время, переключаясь на другую задачу, я рекомендую уделить внимание стыковке.

Противоположная концепция — сразу начать писать. При достаточном опыте, интуиции, уверенности, что "задача простая, раз ее все сдают" или критическом недостатке времени в конце, когда необходимо хоть как-то попытаться ее решить, стыковку можно опустить, точнее, производить одновременно с написанием решения. Этот подход применяется повсеместно, но вряд ли его можно рекомендовать.

2.5. Реализация

На этом этапе собственно пишется программа. Иногда предпочтительнее программирование "сверху вниз", иногда — "снизу вверх", или их комбинация.

Первый подход используется, когда существует общее видение программы — тогда пишется основная часть, а функции и процедуры не реализуются, только согласовывается их интерфейс. Это делается достаточно быстро, если разбиение на подпрограммы достаточно удачно, и позволяет окончательно уложить мысли в голове. Также упрощается отладка, понимание и дописывание другими участниками.

Подход "снизу вверх" используется, когда требуется что-то писать, не очень понятно, что именно, а время

уходит. Тогда можно сначала написать то, что потребуется в любом случае, например — ввод входных данных, функции для работы с геометрией или арифметику повышенной точности. Хорошо, если в команде устоялись реализации стандартных алгоритмов, конвенции о названиях переменных и функций и т.д. При необходимости потратьте полминуты и допишите комментарии о том, что за значения хранятся в каждой из переменных, что возвращает та или иная функция и т.д.

На практике, конечно, используются смешанные подходы.

2.6. Тестирование и отладка

Добившись того, чтобы программа компилировалась, необходимо убедиться в ее правильности. Проблемы могут быть в мелких ошибках, допущенных в процессе написания: перепутанные имена переменных, неверный знак в формуле и т.д. Решение может быть принципиально неправильным или неэффективным. Размер массивов может быть недостаточным или, напротив, чрезмерным, что будет вызывать ошибку "превышен предел памяти".

Поэтому программу необходимо тестировать, если, конечно, речь идет не о последней минуте соревнований. Программу, дописанную за три минуты до конца, не следует тестировать только при условии, что тестирующая система работает очень нестабильно, и на посылку решения в жюри одной минуты мало.

Первое правило тестирования — проверяйте задачу на тесте (наборе входных данных) из примера. Какой бы правильной ни казалась ваша программа, каким бы простым ни был тест из примера, все равно в половине случаев тест из примера не пройдет. Все-таки решение пишется в обстановке нервного напряжения и на скорость. Далее, не ленитесь придумывать свои тесты. Вводите много "маленьких" тестов. Старайтесь не стирать тест, однажды введенный в компьютер. Если вы хотите слегка изменить его, предварительно скопируйте — пусть лучше будет два теста.

Второе правило — внимательно проверяйте, что программа выдала на тесте. Очень часто, когда программа правильно работала на девяти тестах, придуманных командой, но выдает неправильный ответ на десятом, команда этого не замечает, поскольку запускает программу на десятом тесте только для "очистки совести".

Отметим, что в достаточно сложных задачах помогает встраивание в программу элементов автоматической проверки. Очень полезна процедура ŸŸŸŸŸŸ или ее аналоги. Например, после сортировки можно проверить, действительно ли массив отсортирован. Если это не так, пусть программа завершается с ошибкой. Если задача достаточно сложная и включает не только сортировку, такие предосторожности практически наверняка окупятся. Единственная проблема — не недостаток времени, а лень, ведь на тренировках все десять раз писали тот или иной алгоритм. К сожалению, никакая тренировка не гарантирует безошибочного написания на соревнованиях даже таких простых алгоритмов, как, например, алгоритм Евклида, алгоритм Дийкстры или двоичный поиск.

Кроме маленьких тестов, необходимо всегда проверять решение на так называемом "максимальном тесте". Для каждой задачи следует, если это возможно, сгенерировать максимальный тест — полностью случайный большой тест с максимальными ограничениями. Сгенерированный таким образом тест не во всех задачах будет худшим по времени работы программы. Далеко не во всех — и хорошо бы понимать, в каких. Но на практике вы будете поражены, увидев, насколько часто такой тест приводит к ошибке времени выполнения. Конечно, ответ к такому тесту проверить нелегко, но часто по нему легко понять, что он — неправильный. Добавление в программу многочисленных проверок, лучше всего — исчерпывающих (если задача большая и трудная), облегчает использование максимальных тестов.

Вечная дилемма: что лучше — тестировать или проверять логику? Если в программе есть небольшая часть, вызывающая большие сомнения, лучше ее обдумать и сразу написать правильно. Потому что если в программе есть и другие ошибки, тестирование может оказаться долгим и утомительным. Если вы не тестируете программу, распечатайте решение и проверяйте его по распечатке, освободив компьютер для других участников команды.

Необходимо выдерживать баланс между этими подходами. Всегда следует проверять программу на нескольких небольших тестах. В случае неверного ответа снова имеется альтернатива: отлаживать программу или искать ошибки, внимательно читая ее код (как правило, опять же по распечатке). Помните, что для отладки часто достаточно трех—пяти минут, тогда как поиск ошибок на бумаге легко может затянуться на полчаса. Это занимает одного из членов команды, нервирует всех троих и к тому же увеличивает штрафное время. Выбирать следует по обстоятельствам.

По ходу соревнований приходится регулярно принимать тактические решения. Проверять решение дальше или отправить его в жюри? Тестировать, отлаживать программу или искать ошибки по распечатке? Писать решение задачи в одиночку или вдвоем? Важно правильно расставлять приоритеты. Что важнее в последний час соревнований — надежное решение одной задачи или рискованная попытка решить еще две?

На практике сложно уделить тактике достаточно внимания. Опишу единственный метод, который оказывался полезным во всех без исключения случаях. В самом начале соревнования возьмите чистый лист бумаги и выпишите на нем в столбик буквы, соответствующие задачам. В дальнейшем вычеркивайте решенные задачи. Удобно также помечать задачи, по которым есть идеи, или решения к которым частично написаны. Обнаружив задачу, на решение которой заведомо не хватит времени, также вычеркните ее из списка.

2.7. Посылка на проверку в жюри

Не забывайте про отладочную информацию, включение/выключение оптимизации и проверок переполнения арифметики, стека, выхода за границы массива — если вы, конечно, не пишете на языке *Java*. Существуют сторонники макроса *DEBUG* в олимпиадном программировании, которым можно отметить, какие части программы выполняются при отладке, а с какими программа посылается в жюри. Иногда его использование оправданно, но обычно очень уж лень им пользоваться. Впрочем, это приво-

дит к многочисленным "забыл включить проверку переполнения", "забыл убрать отладочную информацию" и т.д. А когда из жюри приходит сообщение "Неверный ответ", отладочная информация возвращается обратно, и затем, после исправления ошибки, ее опять забывают убрать. Та же история происходит, когда требуется выводить информацию в стандартный вывод и читать ее из стандартного ввода, а для отладки используется файл.

Не смотрите, как ваши товарищи по команде посылают программу в жюри, как они нажимают кнопки в клиенте и т.д. Это отнимает ваше время и внимание, а они нервничают, что вы не готовитесь и не обдумываете другую задачу. Впрочем, если уж вы наблюдаете за ними, проконтролируйте, чтобы они выбрали нужный файл, нужный язык программирования и нужную задачу, включили/отключили проверки и т.д.

Не смотрите просто так в монитор, ожидая ответа. Сервер, проверяющий решения участников, может быть перегружен. Он может быть временно в нерабочем состоянии — это типично для соревнований самого разного уровня. Тестируйте эту задачу, пишите другую, думайте над третьей и т.д. В конце соревнований участники часто вносят "случайные изменения" и посылают программу в жюри еще раз, вносят и посылают и т.д. Однако, если осталось хотя бы пять минут, лучше придумайте пару небольших содержательных тестов и проверьте программу на них. Если обнаружится ошибка — сконцентрируйтесь и исправьте ее, а не просто добейтесь верного ответа на данном конкретном примере. Помните, что в программе часто есть повторяющиеся места, поэтому одна и та же ошибка могла быть допущена многократно.

Довольно часто обнаруживается фрагмент написанной программы, который может быть улучшен. Приемлемость существующей реализации может зависеть от трактовки условия задачи, нетривиальных свойств использованных алгоритмов, даже от некоторой доли везения, поскольку иногда участники допускают столь необычные и редко проявляющиеся ошибки, что заготовленные тесты эти ошибки не выявляют. Вы можете улучшить этот сомнительный фрагмент и послать программу на проверку, однако подобные действия часто, наоборот, приводят к увеличению количества ошибок в программе. Поэтому перед внесением исправлений, в необходимости которых вы не уверены, сделайте резервную копию всего решения. Обнаружив впоследствии действительно серьезную ошибку, вы, вероятно, захотите к этой копии вернуться.

3. Заключение

В данной работе рассмотрены общие принципы решения задач в рамках командных студенческих соревнований по программированию формата ACM ICPC. Выполнена попытка формализовать и описать процесс решения задачи вместе с ключевыми аспектами для каждого этапа. Рассмотрен пример. Приведены комментарии и рекомендации.

Конечно, никакая инструкция не заменит реального опыта. Однако на фортепиано почему-то принято учиться играть у преподавателя, и так начинали практически все великие композиторы. Соответственно, чтение этих рекомендаций должно быть не попыткой применить к себе чужие методы, а содержательным восприятием чужого опыта, изложенным в виде набора рекомендаций. В процессе моей олимпиадной карьеры я много обсуждал различные тактические и стратегические аспекты, точки зрения изменялись на диаметрально противоположные в течение дня. Могло быть и так, что в один сезон мы придерживались одной тактики, а в другой — другой. Тактика зависит от опыта, целей, соперников.

Важно тренироваться самостоятельно, особенно если вы чувствуете, что сильно отстаете хотя бы от одного из своих напарников. Важно быть заинтересованным.

Если вы систематически изучаете эту проблему, то, следовательно, вы всерьез нацелены на решение задач. Необходимо втянуться в этот процесс, жить решением задач, может быть, только им. Необходимо получать удовольствие от этого, чтобы можно было проснуться ночью и пойти решать задачи, быть готовым обсуждать задачи в любое время — во всяком случае, такое помешательство должно быть на начальном этапе карьеры. Конечно, придется потратить сотни часов и решить сотни задач, в том числе самостоятельно. Впрочем, это еще не гарантирует результата. Удачи вам!

Литература

1. *Асанов М.О., Парфенов В.Г.* Финальные соревнования чемпионата мира по программированию. Потрясающий успех петербургских команд // Компьютерные инструменты в образовании № 2, 2001, ***http://ict.edu.ru/lib***.

2. *Богатырев Р.* К истории чемпионатов мира ACM по программированию // Мир ПК — Диск № 6, 2004. ***http://is.ifmo.ru/belletristic/acmhist.rdf***.

3. *Богатырев Р.* Нас не догонят?! // Мир ПК — Диск № 5, 2005. ***http://is.ifmo.ru/belletristic/acm2005.rdf***.

4. *Брудно А.Л., Каплан Л.И.* Московские олимпиады по программированию. М.: Наука, 1990.

5. *Беров В.И., Лапунов А.В., Матюхин В.А., Пономарев А.Е.* Особенности национальных задач по информатике. Киров: Триада-С, 2000.

6. *Кирюхин В., Лапунов А., Окулов С.* Задачи по информатике. Международные олимпиады 1989—1996 гг. М.: ABF, 1996.

7. *Овсянников А., Овсянникова Т., Марченко А., Прохоров Р.* Избранные задачи олимпиад по информатике. М.: Тровант, 1997.

8. *Скиена С., Ревилла М.* Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. М.: Кудиц-Образ, 2005.

9. *Адамар Ж.* Исследование психологии процесса изобретения в области математики. М.: МЦНМО, 2001.

10. *Пуанкаре А.* О науке. М.: Наука, 1983.

11. Сайт "Генрих Саулович Альтшуллер, автор ТРИЗ, РТВ и ТРТЛ". ***http://www.altshuller.ru***.

12. *Андреева Е.В.* Решение задач XIII междунардной олимпиады // Информатика № 37, 40, 42—44, 2001.

13. *Окулов С.М., Шулятников Д.В.* Разбор задач международной олимпиады 2000 года // Информатика № 12, 2001.

14. *Станкевич А.С.* Решение задач I Всероссийской командной олимпиады по программированию // Информатика № 12, 2001.

15. *Станкевич А.С.* II Всероссийская командная олимпиада школьников по программированию // Информатика № 12, 2002.

16. *Андреева Е.В.* Олимпиады по информатике. Путь к вершине // Информатика № 38, 40, 42, 44, 46, 48/2001; № 6, 8, 10, 12, 14, 16/2002.

Математика текстов

А.Л. Семенов,
Москва

Когда-то в детской энциклопедии я прочитал, что математика — это наука о числах и фигурах. "Числа и фигуры" — это название замечательной книжки о разных разделах математики, с которой и вам стоит познакомиться. Однако позже я понял, что в математике есть много такого, что можно только условно назвать числами или фигурами, например, топологические пространства и абстрактные алгебры. Еще в восьмом классе я побывал на лекции Андрея Андреевича Маркова для участников олимпиады и узнал, что можно строить всю математику, начиная с представления о *текстах* — последовательностях (цепочках) символов.

Впрочем, название "*математика текстов*" нетрадиционно, обычно употребляется название "*математическая логика и теория алгоритмов*". Вам, наверное, даже более знаком термин "*информатика*". На уроках информатики в школе проходят разные вещи: так, если в школе есть компьютерный класс, то детей учат работать с компьютером. Но я употребил слово "информатика" в другом смысле. Наука, которую можно назвать также "*математическая информатика*", или "*теоретическая информатика*", в школьном курсе информатики изучается, но совсем немного. (Когда-то я участвовал в написании первого в нашей стране учебника по информатике для всех школьников 10-х и 11-х классов, и мы включили в него немного математики. Так происходит и сейчас. Но часто математику там трудно увидеть, и требуется некоторое усилие, чтобы понять, где же математическая суть в этой информатике.) Но все это: математика текстов, математическая логика и теория алгоритмов, математическая информатика — относится к одной и той же области математики.

Математика текстов, в частности *математических* текстов (для этой математики еще придумали название "*метаматематика*"), появилась раньше, чем компьютеры, она существовала еще в прошлом веке. Необходимость и полезность этой науки хорошо осознавал один из величайших математиков конца XIX — начала XX века Давид Гильберт. Но именно с возникновением компьютеров математика текстов стала особенно важной областью математики.

Метаматематика изучает, в частности, устройство доказательств и аксиоматических систем в различных разделах математики, таких, как арифметика, геометрия, теория множеств: что в этих системах можно доказать, чего доказать невозможно и т.д. Но в этой статье речь пойдет о более понятных и наглядных вещах.

Начнем с нескольких примеров.

Логические парадоксы

Возьмем, например, такой простой текст:

Утверждение, записанное в последней и предпоследней строках на третьей странице этой газеты, ложно.

Многие из вас, наверное, встречали в тех или иных книжках похожий текст. Напомню, что проблема (кстати, вполне математическая) состоит в том, чтобы понять, истинно это утверждение или ложно. Подумайте над этим.

Вот еще один пример.

Некоторые русские слова и словосочетания описывают числа (например, "сто двадцать восемь", "миллион" и т.п.). Рассмотрим такое число: наименьшее натуральное число, которое нельзя описать текстом короче 100 символов (1) (символами мы считаем буквы русского языка, цифры, знаки препинания, пробелы).

С одной стороны, ясно, что существуют числа, которые нельзя описать текстом короче 100 символов — просто потому, что таких текстов конечное число, а чисел бесконечно много. С другой стороны, наименьшее такое число описано текстом (1), в котором меньше 100 символов.

И, наконец, еще один пример.

Однажды учитель математики объявил ученикам, что на следующей неделе он проведет контрольную работу. При этом накануне контрольной они не будут знать точно, что контрольная завтра. (Уроки математики проходят каждый день с понедельника по пятницу.)

Ясно, что в пятницу контрольной быть не может: тогда в четверг вечером школьники точно будут знать, что контрольная завтра. Но тогда в среду вечером они точно будут знать, что контрольная в четверг, ведь в пятницу ее быть не может. Значит, и в четверг не может быть контрольной... Словом, ученики пришли к выводу, что контрольной вообще не будет, и успокоились. А контрольная произошла в среду.

Не пытаясь до конца выяснить причину странности, парадоксальности приведенных примеров, заметим только, что парадокс возникал при попытке применить в рассуждении конструкцию

"с одной стороны..., с другой стороны...",

и при этом получались взаимно противоречащие друг другу утверждения.

Оказывается, эти конструкции могут быть применены к важным математическим построениям, уже не парадоксальным, хотя часто и демонстрирующим невозможность чего-то.

Диагональный метод Кантора

Рассмотрим квадратную таблицу $n \times n$ клеток, в которой некоторые клетки заштрихованы, а остальные

клетки — белые. Пусть нам нужно построить такую строку, которой нет в этой таблице. Давайте последовательно выпишем в строку клетки, которые в таблице стоят по диагонали (*рис.* 1); полученную строку обозначим через d. Строка d может присутствовать в нашей таблице, а может и не присутствовать. Поменяем в этой строке цвета, то есть сделаем белые клетки черными, а черные — белыми; переделанную таким образом строку обозначим через $\overline{d}$. Вот строки $\overline{d}$ в нашей таблице точно нет: от первой строки она отличается первой клеткой, от второй строки — второй клеткой, ..., от n-й строки — n-й клеткой.

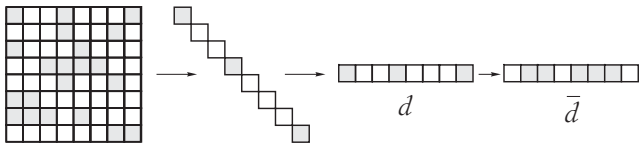


Рис. 1

Рассмотрим теперь бесконечную таблицу. Как и раньше, мы можем построить строку d из клеток, стоящих на диагонали (только теперь она получится бесконечной), и получить из нее строку $\overline{d}$ (*рис.* 2). Так же, как в конечном случае, доказывается, что бесконечной строки, совпадающей с $\overline{d}$, в таблице нет.

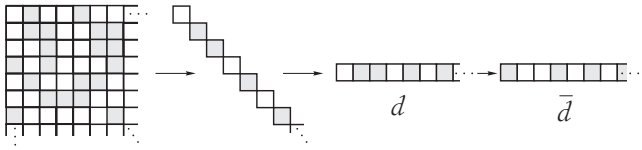


Рис. 2

Что же в нашей конструкции замечательного? Да, мы научились строить бесконечную строку, которой нет в нашей таблице. (Такое построение и называется *диагональным.*) Какие из этого можно извлечь последствия? Самое знаменитое из них получится, если мы предположим, что в нашей таблице имеются вообще все бесконечные строки. В этом случае наше построение сразу приводит к ложности такого предположения.

Мы установили следующий замечательный факт.

Факт. Невозможно все бесконечные последовательности из нулей и единиц выписать в таблицу.

Иными словами, все такие последовательности нельзя пересчитать, перенумеровать, решить, какая последовательность будет первой строкой в нашей таблице, какая — второй и т.д. Говоря научно, невозможно установить взаимно-однозначное соответствие между натуральными числами и всеми последовательностями из нулей и единиц.

Эту знаменитую теорему впервые доказал в XIX веке Георг Кантор, основатель теории множеств.

Контрольный вопрос. Можно ли выписать в бесконечную таблицу все строчки, в которых лишь конечное число черных клеток, а остальные — белые?

Программы

Некоторые тексты, в частности тексты на русском языке, являются *программами*, или *алгоритмами*, — предписаниями, которые можно выполнить. То, к чему применяется программа, тоже текст. Например, алгоритм сложения двух натуральных чисел "столбиком" — это следующий текст на русском языке:

1. Запиши два числа друг под другом "столбиком".
2. Если числа различаются по длине, дополни короткое слева нулями, чтобы их длины сравнялись.
3. Запомни 0 в уме.
4. Рассмотри самую правую пару цифр (по вертикали), которая еще не была рассмотрена. Сложи эти цифры и добавь то, что в уме. Если сумма меньше 10, напиши ее под взятой парой, запомни в уме 0; если сумма больше или равна 10, запиши под взятой парой последнюю цифру суммы и запомни в уме 1.
5. Если все цифры слагаемых рассмотрены и в уме 0, то результат уже получен в нижней строке; если все цифры слагаемых рассмотрены и в уме 1, допиши справа 1 к нижней строке, и это — результат; если не все цифры слагаемых рассмотрены, перейди к пункту 4.

Применяется эта программа к тексту, который представляет собой два числа, записанные в десятичной системе счисления и разделенные запятой и пробелом, например, к такому:

4850493897329785961, 29785565428497

Бывают программы, распознающие некоторое свойство. Такие программы на любом тексте дают ответ либо "да", либо "нет". Например, можно себе представить программу, которая определяет, просто ли данное число.

Еще один пример программы:

1. К данному тексту припиши в конце букву "А".

Это очень простая программа, она заканчивает работу на любом тексте всего за один шаг. Бывают программы, которые не заканчивают работу, как, например, такая:

1. К данному тексту припиши в конце букву "А".
2. Перейди к пункту 1.

Она припишет к тексту букву "А" в конце, потом еще раз припишет букву "А", потом еще раз припишет букву "А" и т.д.

Встречаются и более сложные случаи, когда программа на одних текстах заканчивает работу, а на других — нет. Если программа $П$ на тексте T не заканчивает работу, будем это обозначать так: $П(T) = \infty$. Если же программа $П$ заканчивает работу на тексте T, будем писать: $П(T) = !$.

Самоприменимые программы

Поскольку каждая программа представляет собой текст, то мы можем запустить ее на самой себе, на собственном тексте. Программа $П$ называется *самопри-*

менимой, если $П(П) = !$. А можно ли по программе $П$ узнать, является ли она самоприменимой?

Если просто запустить программу $П$ на тексте $П$, то со временем может выясниться, что она закончила работу. Но ни в какой конечный момент времени, если программа еще работает, мы не можем быть уверены, что она никогда не закончит работать. Ведь может случиться так, что программа работает несколько суток, решая какую-нибудь сложную задачу, и это не означает, что она собирается работать бесконечно долго; может быть, она закончит работу через месяц или через миллион лет.

Но, может быть, есть какой-нибудь другой способ, читая тексты программы, понять, что она не заканчивает работать на самой себе?

Допустим, что существует программа S, которая, получая текст программы $П$, распознает, является ли $П$ самоприменимой, и дает ответ либо "да", либо "нет". Тогда мы можем построить программу $\tilde{S}$, которая, получив на входе программу $П$, запускает программу S и, если $S(П) =$ "да", работает бесконечно долго (выполняет какую-нибудь бесконечную процедуру, например, бесконечное число раз умножает 0 на 0), а если $S(П) =$ "нет", тоже говорит "нет", то есть заканчивает работу. Таким образом, программа $\tilde{S}$ не оканчивает работу на самоприменимых программах и заканчивает — на несамоприменимых.

Заметим, что конструкция здесь тоже "диагональная", как и в случае таблицы с закрашенными клетками. Только сейчас мы рассматриваем таблицу, в которой и по горизонтали, и по вертикали выписаны все программы (точнее, все конечные последовательности символов того языка, на котором мы решили записывать программы (табл. 1), при этом, если текст не является программой, мы можем считать, что он просто ничего не делает с исходными данными). В клетки таблицы мы записываем результат работы программы (стоящей в заголовке столбца) на исходном тексте (стоящем в заголовке строки).

Таблица 1

	А	Б	АА	АБ	БА	ББ	В	ВАБ	...
А	АБ	"да"	А	)	р?	∞	А	А	
Б	!,7	"да"	Б	Г	ё	∞	Б	Б	
АА	у,	"нет"	АА	Ж	”	∞	АА	АА	
АБ	3,	∞	АБ	,	—“	∞	АБ	АБ	
БА	Ыы	"да"	БА	∞	∞	∞	БА	БА	
ББ	1 1	"нет"	ББ	2ы3	фщ	∞	ББ	ББ	
В	Й	"нет"	В	(1(	ч)!	∞	В	В	
ВАБ	Т	"нет"	ВАБ	2Ц1	600:	∞	ВАБ	ВАБ	
...									

Если программа S существует, то существует и программа $\tilde{S}$. Предположим, что $\tilde{S}(\tilde{S}) = !$, то есть $\tilde{S}$ заканчивает работу. Но тогда $S(\tilde{S}) =$ "нет" по определению $\tilde{S}$, следовательно, $\tilde{S}$ несамоприменима. И наоборот, если $\tilde{S}(\tilde{S}) = \infty$, то $S(\tilde{S}) =$ "да" и $\tilde{S}$ самоприменима. Противоречие.

Итак, мы установили следующий факт.

Факт. Не существует программы, которая распознавала бы самоприменимые программы.

Это один из фуудаментальных фактов теории алгоритмов, а конструкция, на которой основано его доказательство, — одна из важнейших в этой теории.

Конечно, вопрос о самоприменимости программ кажется довольно экзотическим: не так уж часто мы применяем программу к самой себе. А вот применение программы к другой программе — обычное дело: операционная система всякого реального компьютера (а она, конечно, является программой) выполняет другие программы, заданные человеком, то есть применяется к ним.

Заметим, что и примеры, приводимые ранее, когда не удавалось определить истинность простого на вид утверждения, выглядят "далекими от жизни", и неясно, стоит ли им придавать серьезное значение. Попробуем объяснить, почему дело обстоит серьезно. Первая причина состоит в том, что один контрпример — в математике уже опровержение общее теории или метода. Вторая причина в том, что как-то отделить "диагональные" случаи и "недиагональные" не удается, и это может быть доказано после соответствующего уточнения понятий. Однако во многих случаях математики спрашивают друг друга: "А есть ли «естественный» недиагональный пример?".

Универсальная программа

Иногда возникает необходимость в программах, которые применяются к парам текстов. Каким образом можно из пары текстов сделать один? Записать два текста рядом, один за другим? К сожалению, после этого уже невозможно будет определить, где кончается первый текст и начинается второй. Чередовать буквы первого и второго текстов? Но если тексты разной длины, снова не удастся разделить полученный текст на два исходных.

Попробуем теперь применить более сложный прием. Запишем первый текст, удваивая каждую его букву. Например, текст

Универсальная**программа

(знак "**" обозначает пробел) запишется так:

УУнниивверрссааллььннаая****ппррооггрраамммммаа

После первого текста запишем *разделитель* "аб". Потом запишем второй текст без изменения. Допустим, что второй текст выглядит так:

Сложность**текста

Тогда окончательный результат будет таким:

УУнниивввеерррссаалллььннааяя****ппррроогггрраамммммаааабСложность** текста

Покажем, что по такой записи можно однозначно восстановить оба текста. Будем читать из полученной строки по два символа. Если прочитанные символы совпадают, то они представляют один символ первого текста (*рис.* 3). Если в некоторый момент мы натолкнемся на два различных символа, то это означает, что мы дошли до разделителя "аб", который при восстановлении текстов нужно выбросить. После разделителя прочитаем по одному символу второй текст.

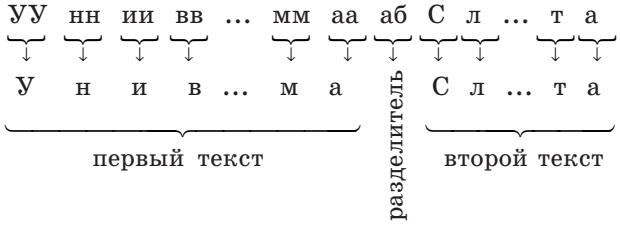


Рис. 3

Оказывается, существуют способы кодирования текстов одним текстом, причем можно делать это экономно, добавляя не слишком много новых букв, так чтобы длина кода была довольно близка к сумме длин исходных текстов.

Задача. Насколько длина кода пары текстов может быть близка к сумме их длин?

Будем считать, что мы уже научились из двух текстов делать один. После этого можно построить программу U, которая применяется к паре текстов $П$ и X, а результатом ее работы будет $П(X)$. Это тоже один из важнейших фактов теории алгоритма.

Факт. Существует *универсальная* программа, которая, получая на вход программу $П$ и текст X, применяет $П$ к X.

Это утверждение не кажется очень сложным, хотя вполне формальные доказательства его довольно громоздки.

Сложность текста. Случайные и неслучайные последовательности

Не так давно, в середине 60-х годов XX века, было выяснено, что математика текстов может быть использована для прояснения наших представлений о природе случайностей. Начнем с обсуждения того, что нам самим кажется случайным, а что неслучайным.

Предположим, что мы бросаем монету и записываем результаты в виде последовательности из нулей и единиц: если выпадет орел, пишем ноль, а если решка — единицу. Может ли получиться такая последовательность:

01101100101110101101001000000111101100111?(2)

Наверное, может. А такая:

00000000000000000000000000000000000000000?(3)

Вряд ли.

Почему, глядя на эти последовательности, мы сразу понимаем: так бывает, а вот так — не бывает? Первый приходящий в голову ответ: потому что первая последовательность "вероятна", а вторая — "невероятна". Однако обе последовательности имеют одинаковую вероятность! (Равную 2^{-n}, где n — длина последовательности.) Чем случайные последовательности отличаются от неслучайных? Попытка прояснить этот вопрос привела к такому важному понятию, как *сложность конечной последовательности*, или *сложность текста*.

Сейчас мы временно отложим вопрос о случайности и поговорим о сложности текстов.

Бывает, что короткий текст описывает длинный. Например, текст

миллион слов "да",

состоящий из 17 символов, является описанием текста длиной два миллиона символов.

Пусть $П$ — какая-нибудь программа. *Сложностью* текста S при способе описания $П$ называется наименьшая длина текста T такого, что $П(T) = S$. Текст T при этом называется описанием текста S.

Иными словами, мы берем текст S, рассматриваем все его описания, из которых программа $П$ может получить S, и среди них ищем самое короткое. Например,

миллион слов "да" и пятьсот слов "дада"

— два описания одного текста, но первое описание короче.

Заметим, что для последовательности (3) мы сразу можем указать короткое описание, в отличие от последовательности (2). Естественно ожидать, что у большинства последовательностей нет коротких описаний. Такие последовательности и называются *случайными*.

Это понятие случайности (его в 60-х годах ввел Андрей Николаевич Колмогоров) оказывается очень интересным и продуктивным.

Докажите, что существует способ, который дает почти самые короткие описания для всех последовательностей. Более подробно, существует такой способ K, что сложность любого текста S при способе описания K меньше, чем его сложность при любом другом способе описания $П$ плюс некоторая постоянная, зависящая только от $П$.

(Эту теорему доказал А.Н. Колмогоров.)

Построение этого способа описания K использует конструкцию универсальной программы, о которой говорилось ранее. Попробуйте сами построить этот способ. У вас получится!

НАЧАЛКА
газета-клуб для всех, кто учит информатике маленьких детей

№ 1 (1–15 января)

Двухкомпонентный курс информатики для начальной школы

А.В. ГОРЯЧЕВ

1. О причинах популярности курса "Информатика в играх и задачах"

В свое время авторы курса "Информатика в играх и задачах" обратили внимание на то, что при объектно-ориентированном анализе и проектировании требуется целенаправленное применение таких умственных действий, как абстрагирование, декомпозиция, построение иерархий — действий, которые относятся к так называемым "когнитивным", или познавательным, действиям. Известно, что младший школьный возраст является возрастом особой восприимчивости детей к развитию логического мышления, в этом возрасте происходит переход к иерархиям понятий и их многоступенчатой систематизации.

Учитывая тезис Выготского о том, что обучение опережает развитие, мы предположили, что обучение элементам объектно-ориентированного, алгоритмического и логического анализа поможет интеллектуальному развитию младших школьников, а это отразится на успеваемости по базовым дисциплинам. Наши ожидания подтвердились. Более того, мы обратили внимание на то, что анализ как этап профессиональной деятельности системных аналитиков, рассматриваемый отдельно, имеет несомненную общеобразовательную ценность и его можно трактовать как этап формализации знаний. Обучение объектно-ориентированному, алгоритмическому и логическому анализу можно рассматривать как обучение автоформализации знаний. Под автоформализацией знаний понимается компактное и однозначное представление человеком своих (именно своих) персональных знаний. Формализация знаний необходима для возможности обмена знаниями, для создания и развития так называемой "экономики знаний". К методам формализации и автоформализации знаний относятся классификация и систематизация знаний. Как правило, классификационные системы имеют иерархическую или древовидную структуру.

К умениям младших школьников, связанным с объектно-ориентированным, алгоритмическим и логическим анализом, мы отнесли умения:

- выделять классы (группы однородных предметов);
- находить общее в составных частях и действиях у всех предметов из одного класса (группы однородных предметов);
- называть общие признаки предметов из одного класса (группы однородных предметов) и значения признаков у разных предметов из этого класса;
- определять составные части предметов, а также, в свою очередь, состав этих составных частей и т.д.;
- приводить примеры последовательности действий в быту, сказках, понимать построчную запись алгоритмов и запись с помощью блок-схем;
- выполнять простые алгоритмы и составлять свои по аналогии;
- выполнять алгоритмы с ветвлениями, повторениями, параметрами, обратные заданному;
- выбирать схему, правильно изображающую предложенную ситуацию;
- изображать множества с разным взаимным расположением;

• находить на рисунке область пересечения двух множеств и называть элементы из этой области;

• отличать высказывания от других предложений, приводить примеры высказываний, определять истинные и ложные высказывания;

• записывать выводы в виде правил "если-то";

• по заданной ситуации составлять короткие цепочки правил "если-то".

Многие учителя начальных классов, пропустив слова про объектно-ориентированный анализ как непонятные, подтвердят вам значимость перечисленных умений. Более того, многие из них скажут, что всегда старались (насколько это было возможным) развивать у детей эти умения. То есть систематизированное формирование перечисленных умений многие учителя начальной школы воспринимали и воспринимают как что-то хорошо знакомое, но упорядоченное и разложенное "по полочкам".

Честно признаемся, "полочки" были выбраны непроизвольно. В нашем наборе "полочек" проявилась предметная специфика. Результаты анализа фиксируются в виде схем. Мы предлагали схемы, принятые или аналогичные принятым в информатике: схемы состава, наследования, отношений, алгоритмов, логического вывода. А для учителей начальной школы эти схемы выглядели и выглядят просто как один из видов универсальных схем для описания декларативных и процедурных знаний.

Итак, мы показали пользу, которую обучение элементам объектно-ориентированного, алгоритмического и логического анализа приносит развитию младших школьников, а следовательно, и всему учебному процессу, базовым учебным дисциплинам.

Рассмотрим пропедевтическую роль элементов объектно-ориентированного, алгоритмического и логического анализа при обучении информатике. В литературе по специальности неоднократно отмечалось, что объектно-ориентированный подход является по своей сути моделированием, а объекты — абстрактными моделями тех предметов и явлений, которые они представляют. То есть, занимаясь объектно-ориентированным анализом, переходя от рассматриваемых предметов к их представлению в виде таблиц и схем, школьники на практике учатся строить информационные модели. Таким образом, в начальной школе проходит пропедевтика важнейшей линии информатики — формализации и моделирования. Кроме того, при внимательном рассмотрении перечня перечисленных умений нетрудно обнаружить представление информации в виде схем и таблиц, а также пропедевтику алгоритмов и элементов формальной логики. Все эти темы традиционно относятся к наиболее логически сложным при изучении информатики.

Итак, описанный выше подход к пропедевтике курса информатики создает основу для последующего изучения логически наиболее сложных разделов базового курса информатики: формализация и моделирование, алгоритмы, формальная логика. Безусловно, это не полный перечень разделов информатики. Например, мы не считаем самостоятельной ценностью заучивание детьми новых терминов ("информация", "прием", "передача", "приемник", "источник" и т.д.) для обозначения хорошо известных процессов, таких, как чтение, слушание, общение, обмен сведениями. В нашем курсе нет практики работы на компьютере, которая вполне доступна и явно целесообразна для младших школьников. Этот сознательный выбор ограничения области пропедевтики информатики определяется рядом причин:

1. Логика построения необязательного пропедевтического курса. Поскольку мы имеем дело с необязательным курсом, то в базовом курсе информатики мы должны вводить предметные понятия, исходя из предположения, что они вводятся впервые. Может, конечно, неоднократный ввод одних и тех же понятий и имеет смысл, но, на наш взгляд, гораздо целесообразнее сконцентрироваться на своевременном развитии мышления и формировании необходимых умственных действий.

2. Вынужденное разделение пропедевтики информатики и практики работы на компьютере. Начальная школа обладает особенностью, которой нет на других этапах школьного обучения: один учитель ведет уроки по всем базовым учебным дисциплинам. Было бы целесообразно использовать эту ситуацию для обеспечения переноса умственных действий, формируемых на уроках информатики, на уроки по базовым дисциплинам. Основу такого переноса и создает ведение уроков информатики учителем начальных классов. До тех пор, пока учитель начальных классов повсеместно не будет обладать способностью проводить полноценные уроки по информационным технологиям, мы вынуждены рассматривать пропедевтику информатики и пропедевтику информационных технологий в начальной школе как две отдельные сущности, как два отдельных урока с разными учителями и в разных классах (обычном и компьютерном). Конечно, желательно предусматривать и вариант объединения этих уроков.

3. Невозможность объять необъятное. Как говорится в известном стихотворении, "больших семь шапок из овцы не выкроишь никак". Известный недостаток одночасовых предметов — это нагруз-

ка в основном на память, перегрузка новыми понятиями (введение новых понятий без достаточного времени на освоение их в действиях). Поэтому пропедевтика чего-либо в одночасовом предмете предполагает разумное самоограничение, выбор "направления главного удара", оставляя ряд тем на изучение в базовом курсе.

Выбор основного направления пропедевтики в виде освоения базовых, фундаментальных умственных действий полностью соответствует деятельностному подходу в обучении, сущность которого заключается в рассмотрении процесса учения как деятельности. Для преподавателя это означает, что в процессе обучения, при передаче опыта общественно-исторической практики он должен решать задачу формирования у обучаемых умения осуществлять деятельность. Следовательно, целью обучения также является деятельность или действия и операции, с помощью которых она реализуется и которые направлены на решение специфических для учения задач. Систему операций, которая обеспечивает решение задач определенного типа, называют способом действий. Таким образом, при деятельностном подходе конечной целью обучения является формирование способа действий. При этом всякое обучение основам наук в то же время является и обучением соответствующим умственным действиям, а формирование умственного действия невозможно без усвоения определенных знаний. При деятельностном подходе проектирование учебной деятельности начинают с ее анализа и выделения системы умений: базовых (познавательных и организационных), межпредметных и предметных. Затем планируется последовательность освоения этих умений, начиная с базовых. Именно такую картину мы и наблюдаем в курсе "Информатика в играх и задачах". В нем ставится задача формирования базовых познавательных умений для учебного предмета "Информатика и ИКТ".

Нельзя не упомянуть о возможностях, которые курс "Информатика в играх и задачах" открывает перед школами, специализирующимися на информатике и информационных технологиях. Для таких школ обучение объектно-ориентированному анализу выступает не только как обучение автоформализации знаний, но и как первый этап обучения современной парадигме объектно-ориентированного программирования.

Выводы

Основной причиной популярности курса среди учителей начальной школы является его ориентация на развитие базовых познавательных умственных действий в информатике, относящихся к общеучебным. Эта ориентация положительно влияет на развитие школьников и способствует повышению успеваемости по базовым дисциплинам.

Основной причиной популярности курса среди учителей информатики является его ориентация на пропедевтику логически сложных разделов информатики, позволяющая более быстро и полно осваивать в основной школе логически сложные разделы базового курса информатики, такие, как формализация и моделирование, алгоритмы, формальная логика.

Перечисленные выше причины являются не единственным фактором популярности курса. При его создании ставилась задача: говорить о сложном просто и понятно, в удобной для учителя и ученика форме. В методических пособиях подробно описаны сценарии всех уроков, минимизировано число новых терминов, проанализированы типовые ошибки детей и предложены варианты реакции учителя. В состав комплекта входят контрольные работы. Разработаны наглядные пособия. В течение 5—6 лет мы работали в режиме обратной связи с учителями, ведущими уроки, и постоянно корректировали содержание учебников. Кроме того, мы подбирали занимательные и интересные формы проведения занятий, в том числе задачи и учебные игры.

2. О новом стандарте для начальной школы

В новом стандарте для начальной школы есть два нововведения, имеющих отношение к пропедевтике учебного предмета "Информатика и ИКТ", — это перечень общеучебных умений и раздел образовательной области "Технологии" под названием "Практика работы на компьютере (использование информационных технологий)". Раздел, связанный с изучением информационных технологий, не является для школ обязательным, а вводится в обучение "при наличии материально-технических средств".

Рассмотрим, как соотносятся эти нововведения с содержанием курса "Информатика в играх и задачах". Перечень общеучебных умений примерно на две трети состоит из умений, которые можно назвать информационными (и треть организационных умений). Курс "Информатика в играх и задачах" можно считать более или менее близким перечню информационных умений (за исключением тех умений, которые явно относятся к предмету "Чтение"). Более того, можно считать, что в "Информатике в играх и задачах" мы имеем дело с расширением перечня общеучебных умений в направ-

лении большей интеллектуализации, логической сложности. Это привычная картина — расширение минимумов, заложенных в стандартах, в тех или иных авторских учебных программах.

В то же время, содержание курса "Информатика в играх и задачах" практически не имеет точек соприкосновения со стандартом для практики работы на компьютере (использования информационных технологий). Если задуматься о причинах этого, то можно обратить внимание на соотношение информатики и информационных технологий в базовом курсе информатики. По отношению к информационным технологиям информатика выступает как совокупность теоретических основ: логических, алгоритмических, математических, информационных (в узком смысле), этических и так далее. Причем не всегда одновременно с изучаемыми основами изучаются информационные технологии — их приложения. Порой эти основы изучаются впрок. Более того, некоторые разделы информатики могут начать использоваться в информационных технологиях уже за порогом школы. В любом случае информатика — это фундамент применения информационных технологий. В курсе "Информатика в играх и задачах" мы имеем дело с изучением этого фундамента и не касаемся его приложений.

Для каждого логически сложного раздела информатики, безусловно, существуют опирающиеся на них информационные технологии, но не в начальной школе. А в начальной школе ученикам вполне доступно создание таких информационных объектов, как рисунки, анимация, текстовые документы с таблицами, схемами и изображениями, поиск информации и организация найденных сведений (персональное информационное пространство).

Авторам курса "Информатика в играх и задачах" приходилось слышать, что наш курс не соответствует новому стандарту по информатике для начальной школы. В этом утверждении присутствует серьезная ошибка: не существует стандарта информатики для начальной школы. В новом стандарте есть раздел "Информационные технологии" в образовательной области "Технологии". Именно ему не соответствует курс "Информатика в играх и задачах", что неудивительно. Точно так же наш курс не соответствует стандарту по физической культуре. Просто он не является ни курсом физкультуры, ни разделом труда ("Технологий"). Это пропедевтический курс именно информатики — предмета, не входящего в федеральную часть базисного учебного плана для начальной школы и изучаемого в школах за счет регионального или школьного компонента БУП.

3. О дополнении "Информатики в играх и задачах" информационными технологиями

Поскольку в новом стандарте для начальной школы появился раздел образовательной области "Технологии" под названием "Практика работы на компьютере (использование информационных технологий)", нашему авторскому коллективу было предложено разработать учебник по информационным технологиям для проведения занятий на уроках технологий (труда) в 3-х и 4-х классах. Предполагалось, что новый учебник по информационным технологиям будет полностью соответствовать новому стандарту по технологиям. Кроме того, нами было решено обеспечить сочетанием нового учебника и комплектом "Информатика в играх и задачах" пропедевтику всех разделов базового курса информатики.

Поскольку в разных школах по-прежнему разные условия для освоения информационных технологий и разные возможности выделения часов для изучения информатики, нами было решено рассматривать новую разработку как второй компонент пропедевтического курса учебного предмета "Информатика и ИКТ". Мы считали, что, исходя из конкретных условий в конкретной школе, учитель может собрать из двух компонентов оптимальное сочетание для своих классов. Идеальным вариантом мы считаем сочетание отдельного урока информатики и изучение информационных технологий на части уроков технологий в 3-х и 4-х классах, как это предусмотрено федеральным базисным учебным планом.

Мы считали, что новый учебник должен решить задачу пропедевтики разделов информатики, не охваченных комплектом "Информатика в играх и задачах", не через изучение теории этих разделов, а через накопление учениками опыта информационной деятельности. Мы предпочитали не включать в учебник рассказы о разных видах информации, с которой можно работать на компьютере, а дать школьнику практическую возможность попробовать своими руками действия с графикой, анимацией, интерактивной анимацией, текстами, схемами, таблицами, с поиском и организацией информации.

Поскольку информационные технологии изучаются по новому стандарту в образовательной области "Технологии", этот факт должен был быть отражен в новом учебнике. Другими словами, у нас должен получиться не учебник информатики, а учебник по одному из разделов "Технологии", который должен следовать общим принципам, относящимся ко всем

учебникам образовательной области "Технологии". Именно поэтому в нем нашли свое место страницы о соотношении материальных и информационных технологий, о профессиях, связанных с информационными технологиями, о возможности применения информационных технологий в школе, о правилах безопасности при работе в компьютерном классе.

Известно, что школьников легче учить сначала создавать на компьютере рисунки, а уже затем переходить к работе с текстами. Мы следовали этому годами проверенному правилу. В 3-м классе по нашей программе школьники будут иметь дело с графикой, анимацией, интерактивной анимацией. В 4-м классе ученики перейдут к созданию текстов, текстов с изображениями, таблицами и схемами, будут заниматься практическим поиском и организацией информации.

Создавая новый учебник, мы столкнулись с двумя проблемами:

1) неопределенность с числом часов, выделяемых на изучение информационных технологий, из общего числа часов, отводимых на изучение технологий,

2) необходимость предусмотреть наличие разных платформ (PC+Windows, PC+Linux, Mac) и привычку работы учителей с полюбившимися программными пакетами.

Для решения проблемы с неопределенностью часов нами было решено делать учебник с модульной структурой. В зависимости от числа имеющихся часов учитель может выбирать для изучения тот или иной набор модулей.

Более оригинальное решение было принято нами для решения проблемы разных платформ и разных программных пакетов. Перед нами стояло классическое противоречие ТРИЗ (теории решения изобретательских задач): в учебнике должны быть описания конкретных кнопок, чтобы дети могли этому научиться, и в учебнике не должны быть описания конкретных кнопок, потому что у разных программ на разных платформах эти кнопки разные. Для разрешения этого противоречия мы применили классический прием ТРИЗ — разделение в пространстве. Мы выпускаем учебник и справочники-практикумы по отдельным программам.

В учебнике мы приводим общую часть описания работы с графикой, анимацией или текстами и даем обзор (неполный) инструментальных программ, с помощью которых можно выполнять эту работу. В обзорах программ мы стараемся представить программы для разных платформ. Кроме того, в этих обзорах мы стараемся указать наиболее популярные программы. Например, программа ЛогоМиры включена у нас в три обзора: программы для работы с графикой, программы для создания анимации и программы для создания интерактивной анимации. Другой пример: программа Роботландия (под DOS) включена нами в обзор программ для работы с графикой.

В справочниках-практикумах мы знакомим школьников с назначением программ, описываем способы выполнения отдельных операций и приводим упражнения для освоения этих операций. Мы не стали заниматься разработкой своих программ, а выбрали из уже созданных те, которые показались нам наиболее привлекательными. Например, для третьего класса мы готовим справочники-практикумы для программ "TuxPaint", "Конструктор мультфильмов", "Конструктор игр", "FloorPlan 3D".

Число справочников-практикумов по конкретным программам будет расти. На первом этапе мы ограничились одним справочником-практикумом для каждого модуля в учебнике. Если учитель предпочитает использовать привычную для него программу, для которой у нас еще нет справочника-практикума, он просто обходится без него, как и поступал раньше.

Поскольку новый учебник относится к учебникам образовательной системы "Школа-2100", разработанной под руководством академика А.А. Леонтьева, в него включены специальные средства реализации целей нашей образовательной системы. В частности, речь идет о такой цели образовательной системы, как выращивание функционально грамотной личности. Под функционально грамотной личностью мы понимаем личность, которая способна использовать все постоянно приобретаемые в течение жизни знания, умения и навыки для решения максимально широкого диапазона жизненных задач в различных сферах человеческой деятельности, общения и социальных отношений. Для родителей мы даем более простое и менее научное определение: функционально грамотная личность — это человек, умеющий действовать, решать возникающие в жизни задачи, оставаясь при этом человеком. Именно как средство достижения этой цели в новых учебниках образовательной системы "Школа-2100" можно встретить составленные по единой форме описания жизненных ситуаций, для решения которых необходимо применять изучаемые умения. Описания жизненных ситуаций, для решения которых необходимо применение изучаемых информационных технологий, можно встретить и в нашем учебнике.

Мы надеемся, что предлагаемый нами двухкомпонентный "конструктор" позволит учителям собирать из него такие конструкции, с помощью которых они смогут комфортно и результативно учить детей.

Второе заседание Большого московского семинара: Алексей Львович Семенов о курсе "Информатика для начальной школы"

8 декабря прошлого года в гостеприимном РГСУ прошло второе заседание Большого московского семинара по методике раннего обучения информатике. Слушали Алексея Львовича Семенова, который рассказал о курсе информатики для начальной школы, который под его руководством разрабатывается с 1986 г. Отметим, что в отличие от большинства "именованных" курсов обсуждаемый называется просто и скромно — "Информатика для начальной школы".

Если бы мы поставили достаточно авантюрную задачу обозначить каждый курс информатики для началки каким-либо одним ключевым словом, то для "Роботландии" этим словом, наверное, было бы "алгоритмизация", для "Информатики в играх и задачах" — "системология", а для "Информатики для начальной школы", вне всяких сомнений, — "математика". Даже в тезисах доклада Алексея Львовича тема была обозначена как "Информатическая математика в начальной школе".

Что же это за странный курс то ли информатики, то ли математики? И какую такую математику можно изучать в начальной школе, когда дети только-только учатся элементарным арифметическим операциям? Алексей Львович рассказал об удивительном факте: когда он с коллегами размышлял над тем, куда бы в школьном курсе "пристроить" ряд важных разделов современной математики, например, теорию алгоритмов и теорию множеств, обнаружилось, что самое естественное место для них именно в начальной школе!

При этом, разумеется, важнейшую роль играет вводимая система понятий, которая, с одной стороны, не должна профанировать серьезную "взрослую" математику, которая за всем этим стоит, а с другой — быть понятной и доступной даже первоклашкам. Нам представляется, что авторы курса обе задачи решили успешно.

Важнейший математический и информатический объект — конечная последовательность — в терминах "Информатики для начальной школы" называется (и это совершенно естественно и для детей, и для взрослых!) "цепочкой". Цепочки можно строить из любых объектов — бусин, геометрических фигур, цифр, букв, даже животных. Над цепочками можно выполнять разнообразные операции и решать множество содержательных (в основе своей совершенно "взрослых") задач. Например, индуктивно строить цепочку объектов: за синим — красный, за красным — зеленый, за зеленым — квадратный и т.д.

Второй важнейший объект, который вводят авторы курса, — "мешок". Мешок — это почти множество. Почти, потому что при добавлении во множество объекта, который в нем содержится, множество не изменится, а при добавлении чего-либо в мешок его содержимое естественным образом увеличится (строго говоря, мешки — это мультимножества, но эти рассуждения, конечно, не касаются ни детей, ни учителей).

Рассказ об интересном семинаре, конечно, можно бы продолжать долго (в частности, речь шла и о логике, и о играх, и о многом другом), но, конечно, в рамки этой краткой "отчетной" заметки не вместить главного — рабочую атмосферу настоящего методического семинара, которого так не хватало уже давно. Напоминаем, что семинар открытый, он проводится по вторым и четвертым четвергам каждого месяца в РГСУ (возможны и выездные заседания). Оперативная информация о семинаре отражается на сайте *http://ito.edu.ru/sp/*. Приглашаем вас, уважаемые коллеги!

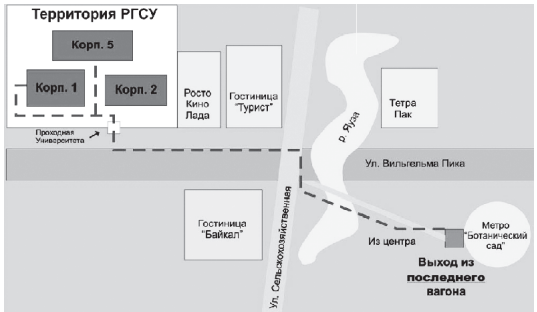
Адрес и контактные реквизиты

107150, Москва, ул. Вильгельма Пика, 4, РГСУ, кафедра социальной и педагогической информатики, комн. 323.

Первин Юрий Абрамович

Тел.: (08535) 3-00-35, 8-905-130-85-63

E-mail: *pervin@robot.botik.ru*



В мир информатики
88 (1—15 января)
Газета для пытливых учеников и их талантливых учителей

Блуждания "случайного" жука

Жук начинает движение из центра квадрата размером 10 × 10 см, подвешенного горизонтально над землей. За каждую секунду жук продвигается на 1 см в произвольном направлении. Какова вероятность того, что к моменту времени T *жук упадет на землю?*

Значение вероятности, упомянутой в вопросе, можно было бы определить, проведя некоторое достаточно большое число экспериментов n и подсчитывая количество случаев m, когда жук падает с квадрата в бак. Тогда искомое значение вероятности было бы равно m / n. Однако ясно, что это сделать невозможно. Здесь нам на помощь приходит так называемое "имитационное моделирование". Можно разработать математическую модель процесса перемещения жука и на ее основе разработать программу, в которой многократно смоделировать "путешествие" бедного насекомого, а затем по полученным результатам рассчитать нужную нам вероятность.

Составим процедуру ОдинЭксперимент (на школьном алгоритмическом языке [1]), в которой моделируется одно такое путешествие. В ней используем следующие величины:

t — время в секундах, прошедшее с начала движения жука;

угол — величина, характеризующая направление перемещения жука на очередном "шаге" — ее значение определяется случайным образом [2]: *угол* = rnd(1) * 360;

x и *y* — координаты насекомого в момент времени *t*;

упал — величина логического типа, фиксирующая факт "падения" жука;

длина — длина стороны квадрата, по которому перемещается насекомое.

Положение жука в момент времени *t* можно определять через положение на предыдущем шаге:

$$x = x + \cos(угол);$$
$$y = y + \sin(угол).$$

Заметим, что расчет положения следующим образом:

$$x = x + \cos(\mathrm{rnd}(1) * 360);$$
$$y = y + \sin(\mathrm{rnd}(1) * 360)$$

приведет к ошибочным результатам — в этом случае координаты будут рассчитаны применительно к разным значениям направления перемещения.

Обсудим также вопрос о выборе системы отсчета координат. Если началом координат считать нижний левый угол упомянутого в условии задачи квадрата (при виде на него сверху), то условие, при котором жук упадет в бак, запишется следующим образом:

$x < 0$ или $x >$ *длина* или $y < 0$ или $y >$ *длина*

Если же начало координат разместить в центре квадрата, то это условие можно записать так:

$x >$ *длина* / 2 **или** $x < -$*длина* / 2 **или**
$y >$ *длина* / 2 **или** $y < -$*длина* / 2

или короче:

abs(x) > *длина* / 2 **или** abs(y) > *длина* / 2,

— где abs — абсолютная величина числа, указанного в скобках.

В процедуре используем второй вариант:

```
алг ОдинЭксперимент
нач цел длина, t, вещ x, y, угол, лог упал
  длина := 10
  x := 0
  y := 0
  t := 0
  упал := нет
  нц
    t := t + 1
    угол := rnd(1) * 360
    x := x + cos(угол)
    y := y + sin(угол)
    если abs(x) > длина / 2 или
      abs(y) > длина / 2
      то
       упал := да
    все
  кц_при t = 60 или упал
```

```
  если упал
  то
    жук_упал := да
  иначе
    жук_упал := нет
  все
кон
```

Нетрудно увидеть, что:

1) в процедуре моделируется перемещение жука в течение 60 сек. или до падения его с квадрата (используется оператор цикла с постусловием);

2) в результате выполнения процедуры присваивается значение величине логического типа *жук_упал*, которая должна быть описана как глобальная.

С использованием созданной процедуры программа для вычисления вероятности падения насекомого имеет вид:

```
лог жук_упал
алг ОпределениеВероятностиПаденияЖука
нач цел i, n, всего_упал, вещ вероятность
  n := 10000
  всего_упал := 0
  нц для i от 1 до n
    ОдинЭксперимент
    если жук_упал
    то
      всего_упал := всего_упал + 1
    все
  кц
  вероятность := всего_упал / n
  вывод нс, "Вероятность падения жука
             равна ", вероятность
кон
```

— где n — количество экспериментов (смысл остальных переменных величин можно легко понять по их именам).

Задания для самостоятельной работы

1. С помощью аналогичной программы на известном вам языке программирования определите вероятность падения при $T = 60$ сек.

2. Исследуйте зависимость этой вероятности от значений T и размера квадрата. В частности, определите:

1) при каком размере квадрата вероятность падения при $T = 60$ сек. будет равна 1;

2) при каком значении T вероятность падения будет равна 0,5, если длина стороны квадрата — 8 см.

3. Определите среднее расстояние, на которое удалится жук от центра через $T = 60$ сек.

Указания по выполнению

1. Оформите процедуру ОдинЭксперимент таким образом, чтобы она меняла значения двух глобальных величин: *жук_упал* (см. выше) и *расстояние_до_центра*, равную расстоянию от жука до центра квадрата в ситуации, когда после проведения эксперимента насекомое осталось на квадрате.

2. В основной части программы определите сумму значений *расстояние_до_центра* для только что указанных ситуаций и количество экспериментов, в результате которых жук остался на квадрате.

4. Если вы знакомы с графическими операторами, получите траекторию движения насекомого.

Ответы[1] присылайте в редакцию (можно выполнять не все задания).

Исследуемый процесс перемещения жука по квадрату можно смоделировать также средствами электронной таблицы Microsoft Excel. О том, как это сделать, мы расскажем в следующем выпуске.

Литература

1. Почему школьный алгоритмический? / "В мир информатики" № 38 ("Информатика" № 39/2004).

2. Случайные числа в программах / "В мир информатики" № 33 ("Информатика" № 34/2004).

Games.exe

Фокус с 16 буквами

Написав по кругу 16 букв, как показано на рисунке, предложите своему товарищу загадать некоторую букву. Потом попросите его считать буквы "про себя" от задуманной буквы (включая и ее) по часовой стрелке до тех пор, пока он не дойдет в счете до 40. Затем пусть он скажет вам букву, на которую пришелся счет 40.

Вы можете узнать загаданную букву по букве, объявленной вашим товарищем. Для этого после того как будет указана буква, на которую пришелся счет 40, вы должны, начиная с нее, считать против часовой стрелки до 8. Загаданной будет буква, на которую пришелся счет 8.

Пример. Пусть ваш товарищ загадал букву Л. Тогда, начав считать с нее по часовой стрелке, при счете 40 он попадет на букву Б. Считая от нее против часовой стрелки, при счете восемь попадем на букву Л.

Почему для отгадывания нужно поступать именно так? Ответ, пожалуйста, присылайте в редакцию.

[1] Все задания можно выполнить также, используя среду программирования на основе языка Лого.

Ответы, решения, разъяснения

к задачам, опубликованным в газете "В мир информатики" ранее

Статья "Бедный торговец" ("В мир информатики" № 57 / "Информатика" № 10/2005)

Задача 2

Определить, какие гири надо класть вместе с грузом, а какие — на свободную чашку, чтобы при грузе массой *W* кг весы были в равновесии, можно двумя способами, которые условно назовем "практическим" и "теоретическим" (почему так — узнаете, ознакомившись с каждым из них).

1. Первый способ моделирует действия человека, пытающегося определить вес груза, используя гири весом 1, 3, 9, ... кг. (Пытались ли вы, уважаемый читатель, решить эту задачу? Если нет — попробуйте сделать это, прежде чем продолжите чтение.) Один из возможных путей решения состоит в следующем. Сначала надо определить самую легкую гирю, вес которой больше или равен весу груза. Рассмотрим в качестве примера два случая:

1) вес груза — 11 кг;

2) вес груза — 21 кг.

В обоих случаях вес гири, о которой шла речь только что, равен 27 кг. Сразу же заметим, что это не означает, что обязательно нужно будет использовать эту гирю — в первом варианте задачу можно решить и без нее. А когда нужно задействовать и гирю в 27 кг? Ответ — когда вес груза больше суммы весов всех гирь, вес которых меньше найденного значения (27 кг). Для нашего случая — больше, чем 1 + 3 + 9 = 13 кг.

Итак, в первом случае мы должны, используя гири весом 1, 3 и 9 кг (возможно, не все), как-то расположить их на чашечных весах, чтобы разность общего веса гирь на свободной чашке и на чашке, где находится груз, была равна 11 кг. При этом точно нужно гирю весом 9 кг установить на свободной чашке.

Во втором случае следует разместить гирю в 27 кг на свободной чашке весов, а затем так разместить гири меньшего веса (также возможно не все из них), чтобы разность общего веса гирь на чашке, где находится груз, и общего веса гирь на свободной чашке (без учета гири в 27 кг) была равна 6 кг.

После этого мы приходим к задаче подбора комбинации гирь для воображаемого груза массой 2 кг (11−9) и −6 кг (27−21) соответственно, которая решается аналогично.

Все действия повторяются до тех пор, пока оставшийся вес груза, применительно к которому нужно подобрать набор гирь, не станет равен нулю.

Прежде чем представлять программу, моделирующую описанные действия, заметим:

1) в программе используется не величина, соответствующая значению 27 в рассмотренных примерах, а величина *макс_вес_гири* — вес самой тяжелой гири, не превышающий вес груза (в нашем случае это 9), при этом 27 может быть получено умножением значения этой величины на 3;

2) при отрицательных значениях веса воображаемого груза (имя этой величины — *осталось*) понятия "свободная чашка" и "чашка с грузом" становятся противоположными по отношению к реальному грузу.

```
алг Подбор_гирь
нач цел W, макс_вес_гири,
        общий_вес_всех_гирь, осталось
  вывод нс, "Задайте вес груза"
  ввод W
  макс_вес_гири := МаксСтепень3(W)
  общий_вес_всех_гирь :=
                    ОбщийВес(макс_вес_гири)
  осталось := W
  нц пока осталось <> 0
    макс_вес_гири :=
          МаксСтепень3(цел(abs(осталось)))
    общий_вес_всех_гирь :=
                    ОбщийВес(макс_вес_гири)
    если abs(осталось) > общий_вес_всех_гирь
    то
     если осталось > 0
     то
      вывод нс, "Гирю массой ",
                макс_вес_гири * 3,
                " — на свободную чашку"
                осталось := осталось
                — макс_вес_гири * 3
     иначе
      вывод нс, "Гирю массой ",
                макс_вес_гири * 3,
                " — на чашку с грузом"
                осталось :=
                осталось +
                макс_вес_гири * 3
     все
    иначе
     если осталось > 0
     то
      вывод нс, "Гирю массой ",
                макс_вес_гири,
                " — на свободную чашку"
                осталось :=
                осталось — макс_вес_гири
     иначе
      вывод нс, "Гирю массой ",
               макс_вес_гири,
               " — на чашку с грузом"
               осталось := осталось +
               макс_вес_гири
     все
    все
  кц
кон
```

где `МаксСтепень3` — функция, определяющая максимальный вес гири, не превышающий некоторое значение *n*; `ОбщийВес` — функция, рассчитывающая общий вес всех гирь, масса которых не превышает некоторого значения массы гири *m*. Соответствующие функции разработайте самостоятельно.

2. Второй способ разработал Илья Митин из г. Зеленокумска. Он предложил составить таблицу, в которой указывается, как используется та или иная гиря, чтобы при грузе массой *W* кг весы были в равновесии:

Гири	Масса груза *W*, кг																						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...	22	23	24	25	...
1 кг	с	г	—	с	г	—	с	г	—	с	г	—	с	г	—	с	г	...	с	г	—		...
3 кг	—	с	с	с	г	г	г	—	—	—	с	с	с	г	г	г	—	...	с	г	г	г	...
9 кг	—	—	—	—	с	с	с	с	с	с	с	с	с	г	г	г	г	...	г	—	—	—	...
27 кг	—	—	—	—	—	—	—	—	—	—	—	—	—	с	с	с	с	...	с	с		с	...
...																							

В таблице буква “с” обозначает, что соответствующая гиря размещается на свободной чашке весов, буква “г” — на чашке с грузом, символ “—” говорит о том, что такая гиря при грузе массой *W* не используется.

Проанализировав таблицу, можно увидеть, что для каждой гири есть повторяющиеся последовательности символов. Так, для гири в 1 кг это “с г —”, для гири в 3 кг — “с с с г г г — — — ”, для гири в 9 кг это последовательность из девяти букв “с”, девяти букв “г” и девяти символов “—” и т.д. А это означает, что если мы для каждой гири узнаем соответствующий ей символ в таблице при заданном значении *W*, то мы сможем сказать, как должна использоваться эта гиря при таком грузе. Соответствующую программу подготовьте самостоятельно[2]. При этом следует учесть, что:

1) каждая гиря начинает применяться при весе груза, большем значения, исследовавшегося в задаче 1;

2) длина повторяющейся последовательности символов для каждой гири равна ее утроенному весу (см. чуть выше).

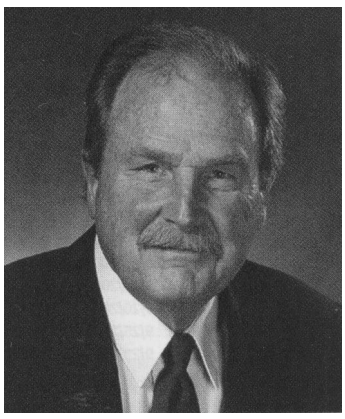
Правильное решение задачи об участниках завтрака в одном американском университете прислали также:

— Глижинский Дмитрий, г. Бендеры, Республика Молдова, гимназия № 2, учитель **Глижинская С.Л.**;

— Дмитриева Алена, основная школа села Васильевка Мелеузовского р-на, Республика Башкортостан, учитель **Зайцева Л.Н.**

Отметим, что Алена использовала для решения не табличный метод, а метод графов.

Личности



Дэвид Хаффман

В 2005 году исполнилось 80 лет со дня рождения Дэвида Хаффмана — автора известного алгоритма сжатия текстовых данных. Он родился в 2005 году в американском штате Огайо. В 18 лет Дэвид получил степень бакалавра электротехники в университете штата. Основную концепцию кодирования данных Хаффман разработал во время Второй мировой войны, когда служил на эсминце офицером-связистом. Изначально алгоритм предназначался для кодирования радиосообщений. Напомним [1], что главная идея алгоритма состоит в том, что наиболее часто используемые в тексте символы кодируются более короткой последовательностью единиц и нулей, а менее часто используемые — более длинной последовательностью. Помимо создания метода минимально-избыточных кодов, Хаффман сделал важный шаг в развитии многих перспективных областей электроники. За свою деятельность он получил множество наград, в том числе медаль Ричарда Хамминга от Института инженеров электричества и электроники (IIEE) за исключительный вклад в теорию информации. Дэвид Хаффман скончался в октябре 1999 года.

Литература

1. *Устинов П.* Методы сжатия данных. / “В мир информатики” № 6—8 (“Информатика” № 38—40/2003).

[2] Текст программы, пожалуйста, присылайте в редакцию.

Полезные советы для работы в Microsoft Word

П.В. Родионов,
село Комсомольское,
Республика Чувашия

Как удалить файл из Word

В пункте **Файл** главного меню текстового редактора Microsoft Word команда **Удалить файл** отсутствует. Но это не означает, что нельзя удалить файл, не выходя из редактора, — это можно сделать из диалогового окна **Открытие документа**.

Для этого воспользуйтесь командой **Файл** | **Открыть**, затем с помощью списка **Папка** перейдите в папку, где лежат документы, которые нужно удалить. Выделите удаляемый файл, щелкнув на нем мышью, и нажмите клавишу Del или щелкните правой кнопкой мыши и в появившемся контекстном меню выберите пункт **Удалить**.

Значок "евро" в тексте

Значок "евро" (€), в отличие от знака доллара, отсутствует на клавиатуре. Для того чтобы вставить значок в документ, необходимо при включенном режиме NumLock и нажатой клавише Alt набрать число 0136 на дополнительной цифровой части клавиатуры.

Как заменить стандартную кнопку печати

Использование стандартной кнопки печати (с изображением принтера) на панели инструментов очень удобно — при щелчке на данной кнопке все страницы документа отправляются на печать. В большинстве случаев этот вариант является оптимальным.

Однако бывают и другие ситуации, когда приходится печатать лишь отдельные страницы из документа или (и) несколько его экземпляров. Тогда в обоих случаях придется прибегнуть к команде **Файл** | **Печать**. В то же время нажимать на кнопку намного быстрее, чем выбирать команды из меню. Поэтому разумно заменить стандартную кнопку печати на ту, которая по действию была бы идентична команде **Файл** | **Печать**, то есть вызывала диалоговое окно печати, а не отправляла бы весь текущий документ на печать.

Для этого воспользуйтесь командой **Вид** | **Панели инструментов** | **Настройка** и выберите вкладку **Команды**. Щелкните правой кнопкой мыши на кнопке **Печать** из стандартного главного меню и выберите команду **Удалить**. Затем в категории **Файл** открытого до этого окна пролистайте список команд, найдите команду **Печать** и перетащите ее иконку на панель инструментов.

Печать больших документов

Чтобы лишний раз не перепечатывать очень большой многостраничный документ (реферат и т.п.), не помешает перед печатью лишний раз проверить, как располагаются, например, таблицы, графики и иллюстрации: не попадают ли они на разрывы страниц, начинаются ли разделы с новой страницы и т.п. Это можно сделать как просмотрев документ на экране (что довольно долго), так и распечатав на принтере его так называемую "сжатую" копию.

Во втором случае распечатывается весь документ, но на каждом листе бумаги размещается не одна страница, а несколько: 2, 4, 6, 8 или 16 страниц. Для того чтобы воспользоваться данной возможностью, выберите команду **Файл** | **Печать** и в разделе **Масштаб** укажите нужное количество страниц и размер бумаги — в результате вы получите своего рода мозаику из страниц документа. Данный прием позволяет сэкономить время и бумагу при печати черновиков документов.

Microsoft Word углубленно

"Ломаем" голову

Шутники и серьезные

Школьники для игры разбились на две "партии": на "серьезных", отвечающих правильно на любой вопрос, и на "шутников", дающих на любой вопрос только неправильные ответы.

Учитель, узнав об этом, спросил у Иванова, серьезный ли он человек или шутник. Не расслышав ответа Иванова, он спросил у Петрова и Сидорова (сидевших рядом с Ивановым и все слышавших): "Что ответил мне Иванов?". Петров сказал: "Иванов ответил, что он серьезный человек". Сидоров же сказал: "Иванов ответил, что он шутник". Кем были Петров и Сидоров?

Древнеегипетская задача

В древнем папирусе, написанном египетским писцом Ахмесом, есть задача: "Разделить семь хлебов между восемью людьми". Как решить такую задачу, сделав при этом минимальное число разрезов?

Ответы присылайте в редакцию.

Внимание! Конкурс

Итоги конкурса № 40 для учащихся

Напомним, что необходимо было привести пословицы и поговорки, в которых имеются слова, являющиеся названием цифр двоичной системы счисления.

Ответы прислали:

— Абалаков Александр и Полушкин Вадим, Кемлянская средняя школа Ичалковского р-на Республики Мордовия, учитель **Силантьев О.П.**;

— Абдуллин Рамиль, средняя школа деревни Старый Бабич Кармаскалинского р-на Республики Башкортостан, учитель **Абдуллин Р.Ф.**;

— Абдрахманов Артур, г. Учалы, Республика Башкортостан, школа № 2, учитель **Галимова З.Н.**;

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

— Аглямов Айнур, г. Бирск, Республика Башкортостан, школа № 1, учитель **Зонова И.В.**;

— Баштанова Марина, г. Братск, лицей № 2, учитель **Коновалова Е.А.**;

— Букаева Любовь, Фирсова Елена и Чапкевич Михаил, г. Орел, лицей № 4, учитель **Чапкевич И.М.**;

— Виссель Юлия, деревня Краснодубье Торопецкого р-на Тверской обл., Воронцовская школа, учитель **Громова Г.В.**;

— Волынцева Анна, Губанова Марина, Звягина Ксения, Идиятов Владимир, Косячкова Марина, Лю Кристина, Масловская Наталья, Мошкина Екатерина, Муллагалиева Адиля, Сенокосова Любовь и Шалагина Анна, г. Ревда Свердловской обл., школа № 28, учитель **Кольцова Е.М.**;

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

— Голованов Сергей, Иванова Наталия, Клементьева Анастасия и Погорелова И., г. Нижний Новгород, школа № 77, учитель **Занозина Г.В.**;

— Гордиенко Е., средняя школа села Гилевка Алтайского края, учитель **Збарах В.Ф.**;

— Граждан Артем, Клипов Игорь, Кожаева Елена, Козлова Виктория и Крадинов Александр, г. Балашов Саратовской обл., гуманитарно-педагогический лицей-интернат, учитель **Сухорукова Е.В.**;

— Гайсин Рашит и Гайсина Галия, г. Уфа, Республика Башкортостан, школа № 18, учитель **Искандарова А.Р.**;

— Зарубина Раиса, Мурашева Екатерина, Ознобихина Олеся и Тюрикова Елена, г. Иркутск, школа № 75, учитель **Выборова Е.С.**;

— Иванов Евгений, г. Озерск Челябинской обл., школа № 32, учитель **Любова Н.А.**;

— Исакова Олеся, средняя школа поселка Лимбяяха Ямало-Ненецкого автономного округа, Тюменская обл., учитель **Исакова И.С.**;

— Камалетдинова Лейсан, г. Нижнекамск, Республика Татарстан, школа № 27, учитель **Павлова В.Н.**;

— Карпенко Валерий, г. Кайеркан, Красноярский край, школа № 37, учитель **Стец В.Ю.**;

— Киселева Ксения и Кудинова Юлия, г. Воронеж, лицей № 2, учитель **Комбарова С.И.**;

— Котельникова Наталия и Минеева Светлана, г. Новокузнецк, школа № 100, учитель **Папшева В.В.**;

— Кривов Дмитрий, Кудлацкий Александр и Ткаченко Анна, средняя школа села Подсосное Назаровского р-на Красноярского края, учитель **Кривова Н.П.**;

— Кулешов Станислав, г. Омск, школа № 28, учитель **Козлова М.Л.**;

— Михолап Роман, г. Киров, школа № 28, учитель **Михолап Э.Л.**;

— Молодцов Сергей, Новобурасский р-н Саратовской обл., Ириновская средняя школа, учитель **Брунов А.С.**;

— Монахова Анна, село Спасское Спасского р-на Приморского края, школа № 8, учитель **Мирошник Л.В.**;

— Петренко Ирина, Попова Анжела, Радзеновская Анастасия, Суворова Полина и Суплотова Любовь, средняя школа села Гонжа Амурской обл., учитель **Комарова В.В.**;

— Пивоварова Елена, поселок Сосьва Серовского р-на Свердловской обл., школа № 1, учитель **Бедрина Н.Т.**;

— Пирогова Светлана, Свердловская обл., Артинский р-н, поселок Арти, школа № 1, учитель **Васильева Е.Л.**;

— Пономарева Дарья, г. Струнино Владимирской обл., школа № 11, учитель **Волков Ю.П.**;

— Резепина Светлана и Хохлова Алиса, средняя школа села Ваганово Октябрьского р-на Челябинской обл., учитель **Сараева Н.Н.**;

— Роот Мария, г. Холмск Сахалинской обл., лицей "Надежда", учитель **Романькова С.Н.**;

— Сизов Леонид, село Комсомольское, Республика Чувашия, школа № 1, учитель **Родионов П.В.**;

— Ситало Ольга, г. Прокопьевск Кемеровской обл. (номер школы и фамилия учителя информатики в письме, к сожалению, не указаны);

— Стафеева Елизавета, г. Новоуральск Свердловской обл., школа № 58, учитель **Стафеева Н.А.**;

— Урдиханова Алина, г. Жуков Калужской обл., школа № 2, учитель **Урдиханова И.Н.**;

— Хаустова Кристина, Вознесенская основная школа Красногвардейского р-на Оренбургской обл., учитель **Гриднев А.Б.**;

— Шамсутдинова Альбина, г. Лениногорск, Республика Татарстан, школа № 8, учитель **Кашапова Р.Х.**;

— Шестаков Михаил и Шнейвейс Анастасия, г. Холмск Сахалинской обл., лицей "Надежда", учитель **Романькова С.Н.**

Пример	Фамилия и имя читателя
Ноль без палочки (ничего не стоящий, не значащий человек[3])	Баштанова Марина, Губанова Марина, Иванов Евгений, Камалетдинова Лейсан, Котельникова Наталия, Радзеновская Анастасия, Сизов Леонид, Стафеева Елизавета, Суплотова Любовь, Хаустова Кристина, Чапкевич Михаил
Ноль внимания (полное равнодушие, безразличие со стороны кого-либо к кому-либо или чему-либо)	Камалетдинова Лейсан, Котельникова Наталия, Сизов Леонид
Абсолютный нуль, круглый ноль (человек ничтожный, совершенно бесполезный в каком-либо отношении)	Граждан Артем, Клипов Игорь, Кожаева Елена, Козлова Виктория, Котельникова Наталия, Крадинов Александр и Сизов Леонид
Голова с куль, а разума с нуль	Гордиенко Е., Пирогова Светлана и Шамсутдинова Альбина
Временами и ноль кажется осью, вокруг которой обращается мир чисел	Камалетдинова Лейсан
Нуль денег, нуль хлопот	Роот Мария
Разума много, да денег ноль	Роот Мария
Ноль гроша в кармане	Роот Мария
Если бы да кабы все на ноль делилось бы	Сизов Леонид
Круглый ноль такой хорошенький, но не значит ничегошеньки	Пономарева Дарья
Подстричь под ноль	Стафеева Елизавета, а также ребята из г. Балашова Саратовской обл.
Единица — вздор, единица — ноль (В.Маяковский)	Сизов Леонид
Курение и алкоголь — здоровья ноль	Граждан Артем, Кожаева Елена, Козлова Виктория, Клипов Игорь и Крадинов Александр

В большом числе ответов были приведены пословицы и поговорки, в которых имелись слова *одна*, *одно*, *одному*, *первый*, *раз*, *пусто* и т.п., не являющиеся названием цифр двоичной системы счисления. Благодаря всех читателей за проведенную работу, заметим, что такие примеры при подведении итогов конкурса не учитывались. Также не учитывались примеры со словами, обозначающими числа, в записи которых используются цифры двоичной системы счисления (*сто*, *тысяча* и т.п.).

Всего участники конкурса привели около 500 (!) примеров пословиц, поговорок, крылатых фраз и т.п. со словами *один* и *ноль*.

84 примера будут опубликованы в брошюре серии "Библиотечка «Первого сентября»", которую все подписчики газеты "Информатика" получат в ближайшее время. В момент подготовки брошюры ее автору не были известны пословицы и поговорки, в которых имеется слово *ноль*. А вот читателям газеты-вкладки "В мир информатики" такие примеры известны — некоторые из них привели ряд крылатых слов и фразеологизмов со словами *ноль* и *нуль* (см. таблицу).

Отметим также ответ Олеси Исаковой (ей всего 7 лет!), которая привела очень интересную эфиопскую пословицу: "Лучше два быка, чем *один* бык. Лучше *один* осел, чем два осла". В ней слово *один* встречается дважды. Это имеет место и во фразеологизме, который привели Погорелова И. и Шамсутдинова Альбина: "Разговор *один* на *один*".

Мы начинаем публиковать примеры, присланные читателями[4] (кроме тех, которые будут представлены в упомянутой выше брошюре).

Аминь, да не ходи *один*!

Артель воюет, а *один* горюет.

Артель расходчика кормит. *Один* мира не съест.

Бабушке *один* только дедушка не внук.

Бела овца, сера овца, а все *один* овечий дух.

Блин добро не *один*. В блин ломоть завернуть.

Бог дал два уха, а *один* язык.

Бог *один* видел, а люди знают.

Бог *один*, да молельщики не *один*аковы.

Бур черт, сер черт — все *один* бес.

В *один* динамик влетело, в другой вылетело. (Прислал Сизов Леонид.)

В *один* уголек дуют. (Привела Иванова Наталия.)

В семи дворах *один* топор.

В чистом поле теснота: *один* кашу варит, да и ту пролил.

Все *один* бог, что у нас, что у них.

Все равны бобры, *один* я соболек.

Всех в *один* кулак сожми, так только то и выжмешь, что съедено.

Вся порода в инохода, *один* дедушка рысак.

Глупый погрешит *один*, а умный соблазнит многих.

Гуляй нечетом: *один* день на разживу покидай.

[3] Комментарии привела Наталия Котельникова из г. Новокузнецка.

[4] Если вы не найдете в перечне предложенные вами примеры, пожалуйста, сообщите об этом в редакцию — мы их обязательно опубликуем с указанием вашей фамилии.

Две бараньи головы в *один* котел не лезут.

Двое — не то, что *один*: подумаем, да и лошадь продадим.

Двое (семеро) одного обедать не ждут, а смелый и *один* ест.

Девка не мак: в *один* день не облетит.

Дело делать и за себя не сможет, а кричит *один* за всех.

Десять раз смеряй, *один* раз отрежь.

До расправы двое похваляются, а после расправы — *один*.

Добр дворянин, что ездит не *один* (а с приносом).

Добрый мордвин живет и *один*.

Дружно — не грузно, а *один* и у каши загинет.

Дурак *один* в воду камень бросит, а двое умных не вытащат.

Дураков в *один* бы косяк сбить, да с кнутищем пастуха приставить.

Его *один* ленивый не бьет.

Ежедён не будешь умён. Дён-то много, а ум-эт *один*.

Если по-русски скроен, и *один* в поле воин.

Живем хорошо: два кваса наладили, *один* как вода, другой еще пожиже.

Живет *один*, как бирюк, как медведь в берлоге, как бухало, как перст, как божедом.

Живет *один*, как божедом.

Жизнь дает *один* только бог, а отнимает всякая гадина.

Жизнь и доверие теряют *один* раз.

Житье хорошее: семерых в *один* кафтан согнали.

Займует *один*, а отдает всем.

Земляк, избил всех в *один* синяк.

И *один* в поле воин, если он советский воин (советский солдат и *один* в поле воин).

И царь в *один* кус не обедает.

Из ничего *один* только бог свет создал.

Из пяти пальцев не вижу ни одного, а *один* в глазах семерит.

Из экого носу выйдет триста тесу, девяносто горбин да охлопень *один*. (Прислал Михолап Роман.)

Кабы у меня брюхо из семи овчин, все *один* бы съел.

Каждый кружится по-своему: *один* кругом, другой через голову.

Коза бела, коза сера, а все дух *один*.

Коли с правдой, так не *один*.

Кто взял, на том *один* грех; кто потерял, на том сто.

Летели три ворона, кричали в три голоса; *один* кричит: "Я Петр"; другой кричит: "Я Филипп"; третий кричит: "Я сам велик".

Лжей много, а правда одна. Ложью как хошь верти, а правде путь *один*.

Лучше десять раз поворотить, чем *один* раз на мель стать.

Лучше *один* друг, чем золота сундук. (Привела Радзеновская Анастасия.)

Лучше *один* раз подраться, чем весь век ссориться.

Лучше семь раз гореть, чем *один* раз овдоветь.

Любит поп блин — да и ел бы *один*.

Мастер *один*, а подносчиков десять.

Мерин с волком тягался — у мерина *один* хвост остался.

Мир — волна: что *один*, то и все.

Муж и жена — одно дело, одно тело, *один* дух.

На весь мир *один* песок сеют.

На гору десятеро тянут, под гору *один* столкнет.

На семи верстах *один* с денежкой Аким.

На сходке народу бывает много, а *один* голова.

Не богат дворянин, а едет не *один*.

Не все рви в *один* раз, покидай в запас на завтра.

Не *один* кулак бьет, бьет и козырь.

Ни кола, ни двора, *один* серенький коток, на шее беленький платок.

Один балахон, да и тот из торговой бани унесен.

Один бог, *один* государь.

Один брат сыт и крепок, другой брат жидок и редок.

Один брат, *один* свет милый. Живут, как брат с сестрой.

Один бульон, поедим и опять дольем. (Прислала Лю Кристина.)

Один в воду, другой ко дну — обоим равно.

Один в воду, другой под лед — оба упрямы: со дна не бывали. (Привел Михолап Роман.)

Один в Кесово, другой в Разлисово (Кесово и Разлисово — до революции — два села Кашинского уезда, Тверская губерния).

Один внизу, семь на возу кричат: "Не заваливай!".

Один воин десять рот водит.

Один воин тысячи водит, а бог и тысячи, и воина водит.

Один волк, да и тот в кусту голодный воет.

Один ворочаешь — ошибок наворочаешь.

Один враг сделает больше вреда, чем сто друзей — пользы.

Один всего не узнает. Одному все мудрости не пройти.

Один всеми доблестьми не овладеет.

Один вяжется, другой сватается.

Один глаз на мельницу, другой на кузницу.

Один глаз на печь, другой в Галич.

Один глаз на полицу, другой в солоницу.

Один глаз, да и тот на Арзамас.

Один глупый камень в море бросит, а сто умных не вынут.

Один говорит — красно, а двое — пестро.

Один говорит — пьян, другой говорит — пьян, как третий скажет — пьян, так поди ложись спать.

Один год десять лет (или весь век) заел.

Один горшок, а много ложек. (Привела Гайзер Яна.)

Один грешок, и тот с мешок.

Один гусь травы (поля) не вытопчет.

Один день год на кости накинул.
Один день у меня век заел.
Один другому не указ. (Прислал Кривов Дмитрий.)
Один дурак может больше спросить, что десять умных ответить.
Один дурак, а умных пятерых ссорит.
Один женился — свет увидал; другой женился — с головою пропал.
Один жил — в лаптях ходил, в колхоз вошел — сапоги нашел. (Привела Виссель Юлия.)
Один за всех, а за тебя — весь цех.
Один и у каши загинет.
Один и дома горюет, а двое и в поле воюют.
Один и камень не поднимешь, а миром и город передвинешь.
Один и камень не сдвинешь, а артельно и гору поднимешь.
Один и у каши не спор.
Один и у каши сирота.
Один карась сорвется, другой сорвется, третий, бог даст, и попадется.
Один мосол во щах не варят.
Один мутит, другой пресноводит (верховодит).
Один на семерых.
Один не годится, другой хоть брось, третий маленько похуже обоих.
Один Нестор и у каши не спор. (Прислала Виссель Юлия.)
Один не тянет, а другой не везет.
Один нынче лучше двух завтра. (Привела Суплотова Любовь.)
Один одним душой. (Прислала Гайзер Яна.)
Один одного стоит.
Один осенний день год кормит. (Привела Минеева Светлана.)
Один осудит, а двое рассудят.
Один от Золотой Орды, другой от рыжей бороды.
Один отец, *один* и норовец.
Один под овин, а два под дорогу (о снопах).
Один поет — другой подвывает.
Один поет — другой подыгрывает.
Один приказчик — *один* вор; два приказчика — два вора.
Один против всех не сголдишь (не сговоришь).
Один проторил тропу.
Один раз не в счет.
Один раз перевозиться — полпожара выстоять; два раза перевозиться — весь пожар выстоять.
Один раз скажи, два раза послушай (два раза послушай, один скажи).
Один раз родился, а дважды помер.
Один раз солгал — навек вруном стал. (Прислала Минеева Светлана.)
Один раз украл (соврал), а на век вором (лжецом) стал.
Один разумный согрешит, да многих глупых соблазнит.
Один рот — и тот дерет. Одна была у волка песенка, и ту переняли (о позевоте).
Один рот, да и тот много врет, а если бы два?
Один рубит, семеро в кулаки трубят.
Один рубль — *один* ум, два рубля — два ума.
Один рубль наживает, а два проживает. (Прислала Тюрикова Елена.)
Один рядовой, да и тот кривой (о гарнизоне).
Один свечку, другой свечку, и попу на овечку.
Один сегодня лучше чем два завтра. (Прислала Пивоварова Елена.)

В следующем выпуске нашей газеты мы продолжим публикацию пословиц и поговорок, присланных читателями. Здесь же в заключение предлагаем читателям:

1) решить задачу, приведенную Мариной Баштановой, в которой, кстати, имеется слово, о котором шла речь в условии конкурса: *Летела стая гусей, навстречу им один гусь: "Здравствуйте, говорит, сто гусей!" — "Нет, нас не сто гусей: кабы было еще столько, да полстолька, да четверть столька, да ты, гусь, так бы нас было сто гусей". Сколько гусей было в стае?*

2) решить два ребуса, подготовленные ребятами из г. Балашова:

3) отгадать загадки, приведенные в ответах Марины Баштановой и других участников конкурса:

— между двух светил я в середине *один*;

— *один* говорит: "побежим, побежим"; другой говорит: "постоим, постоим", третий говорит: "пошатаемся, пошатаемся";

— *один* говорит: "побежим", другой говорит: "полежим", третий говорит: "покачаемся";

— *один* говорит: "побежим", другой говорит: "полежим", третий говорит: "покачаемся";

— сто *один* брат, все в *один* ряд, вместе связаны стоят;

— все паны скинули кафтаны (жупаны), *один* пан не скинул кафтан;

— все капралы поскидали кафтаны, *один* капрал не скинул кафтан;

— весь мир подпоясан, *один* староста распоясан;

— велико поле колыбанское, много на нем скота астраханского, *один* пастух, как ягодка;

— на море, на коробанском, много скота тараканского, *один* пастух королецкий;

— поле поливанское, много скота ивановского, *один* пастырь и два яхонта;

— широко поле карагайское, на нем много скота тараканского, *один* пастух, ровно ягодка;

— есть поле сиянское, в нем много скота монастырского, *один* пастух, словно ягодка;

— земля бела, семена черны: пятеро пашут, двое блюдут, *один* управляет;

— расстилается по двору белое сукно: конь его топчет, *один* ходит, другой водит, черные птицы на него садятся;

— ни небо, ни земля, собой бела: двое глядят, трое работают, *один* повелевает;

— ни небо, ни земля, видением бела; трое по ней ходят, одного водят; два соглядают, *один* повелевает;

— два-ста пудаста, четыре-ста копыласта, *один* болтун да два ухтахта;

— два-ста рогаста, четыре-ста ходаста, *один* махтун и два ухтахта;

— два-ста рогаста, четыре-ста ходаста, *один* хтырок да два ухтырка;

— две головы, *один* хвост, две руки и шесть ног;

— две ковырки, две подковырки, *один* вертун, два войка, третья маковка;

— четверо стелют, двое светят, *один* лежит, никого не пустит;

— четыре стучихи, четыре гремихи, два богомола, *один* вихлец;

— четыре ходастых, два бодастых, *один* хлебестун;

— четыре четырки, две растопырки, *один* вертун да два яхонта;

— шесть ног, две головы и *один* хвост;

— и рать и воеводу в *один* мах перевалял;

— стоит море на пяти столбах, двое об нем спорят, *один* говорит: радость моя; другой говорит: пагуба моя;

4) перевести на русский язык пословицы и поговорки на английском и немецком языках, которые привели учащиеся из г. Балашова:

— на английском языке:

- in the country of the blind *one*-eyed man is a king;
- *one* link broken, the whole chain is broken;
- *one* man, no man;
- *one* man's meat is another man's poison;
- *one* scabby sheep will mar a whole flock;
- *one* swallow does not make a summer;
- *one* today is worth two tomorrow;
- *one* beats the bush, and another catches the bird;
- *one* chick keeps a hen busy;
- *one* drop of poison infects the whole tun of wine;
- *one* fire drives out another;
- *one* good turn deserves another;
- *one* law for the rich, and another for the poor;
- *one* lie makes many;

— на немецком языке:

- man soll nicht zwei Hasen auf *einmal* jagen;
- *eine* Stunde Schlaf vor Mitternacht ist besser als zwei danach;
- *einmal* ist keinmal;

5) повторить быстро 10 раз скороговорку "В *один*, Клим, клин колоти" (☺).

Ответы присылайте в редакцию.

Ответы на вопрос конкурса № 38 прислал также Глижинский Дмитрий, г. Бендеры, Республика Молдова, гимназия № 2, учитель **Глижинская С.Л.**

Конкурс № 41 для учащихся. Тур 6 — заключительный

(чемпионат СНГ среди школьников по переливанию воды (☺))

Составьте алгоритмы решения следующих задач, которые должен выполнить Водомер:

1. Объем емкости A равен 3 л, емкости B — 9 л, емкости C — 10 л, надо получить 8 л в емкости B.

2. Объем емкости A равен 3 л, емкости B — 9 л, емкости C — 11 л, надо отмерить 7 л, использовав при этом минимальное количество воды.

3. Объем емкости A равен 4 л, емкости B — 5 л, емкости C — 9 л, надо отмерить 7 л.

4. Объем емкости A равен 4 л, емкости B — 5 л, емкости C — 10 л, надо отмерить 7 л, использовав при этом минимальное количество воды.

5. Объем емкости A равен 4 л, емкости B — 8 л, емкости C — 9 л, надо отмерить 3 л.

6. Объем емкости A равен 3 л, емкости B — 5 л, емкости C — 11 л, надо получить 4 л в емкости B, использовав при этом минимальное количество воды.

7. Объем емкости A равен 5 л, емкости B — 16 л, надо отмерить 9 л.

8. Объем емкости A равен 5 л, емкости B — 16 л, надо отмерить 8 л.

Ответ отправьте в редакцию до 10 февраля 2006 г. по адресу: 121165, Москва, ул. Киевская, д. 24, "Первое сентября", "Информатика" или по электронной почте: *inf@1september.ru*. Пожалуйста, четко укажите в ответе свои фамилию и имя, населенный пункт, номер и адрес школы, фамилию, имя и отчество учителя информатики.

ДОРОГИЕ РЕБЯТА И ПРЕПОДАВАТЕЛИ!

Журнал “Компьютерные инструменты в образовании” и Институт продуктивного обучения Российской академии образования объявляют третий конкурс **“КИО” (Конструируй, Исследуй, Оптимизируй). Третий конкурс пройдет с 24 по 26 февраля 2006 г.** В этом году участникам конкурса будут предложены задания двух уровней. I уровень будет доступен учащимся младших и средних классов (до 7-го класса включительно). Задания II уровня будут предложены учащимся 8–11-х классов, студентам колледжей и техникумов, а также тем, кто считает себя более подготовленным к участию в конкурсе. Ограничений по возрасту для участников II уровня нет.

Цель конкурса — посредством работы с компьютерными моделями и инструментами познакомить участников с проблемами из различных областей знаний. Для пытливых участников конкурс может стать началом серьезной исследовательской работы.

Содержание конкурса. Всем участникам будет выдано программное обеспечение для проведения компьютерных экспериментов с предложенными моделями задач.

Задачи носят исследовательский характер, то есть решений у задачи в привычном смысле слова может быть много, но нужно постараться найти самое лучшее из них по указанному в задаче признаку. Таким образом, каждый участник конкурса сможет предложить свое решение.

Для некоторых задач до сих пор не известны полные решения, поэтому победители будут определяться по тому, насколько найденные ими решения будут лучше решений, присланных другими участниками.

Имена победителей и их результаты будут опубликованы в журнале “Компьютерные инструменты в образовании”, а также на сайте конкурса.

Каждый участник конкурса получит электронный сертификат, заверенный электронной подписью Оргкомитета. Все победители будут награждены призами и дипломами.

Правила регистрации участников конкурса:

Регистрация участников конкурса производится на сайте конкурса *http://www.ipo.spb.ru/kio/.*

• Участвовать в конкурсе могут учащиеся всех образовательных учреждений без ограничения возраста.

• Участники конкурса должны иметь выход в Интернет или доступ к электронной почте.

• Участие в конкурсе платное. Стоимость регистрационного взноса — 50 рублей для одного участника. Если заявка подается списком на группу, состоящую из 5 и более участников, то стоимость регистрационного взноса на одного человека — 40 рублей. Если заявка подается списком на группу, состоящую из 10 и более участников, то стоимость регистрационного взноса на одного человека — 30 рублей.

• При подаче заявки рекомендуем пользоваться специальной регистрационной программой, которую можно будет получить на сайте конкурса.

• Победители конкурса “КИО-2005” приглашаются к участию в конкурсе без регистрационного взноса.

• Регистрация участников, имеющих доступ в Интернет, производится на сайте конкурса: *http://www.ipo.spb.ru/kio/.*

• Те, кто не имеет прямого выхода в Интернет, могут воспользоваться программой подготовки регистрационного файла, которую можно получить на сайте конкурса или по электронной почте, выслав запрос по адресу: *kio@ipo.spb.ru.*

Регистрационный файл надо отправить по этому же адресу.

В заявке должно быть указано:

1. Фамилия, имя, отчество.
2. Класс, школа.
3. Почтовый адрес.
4. Телефон.
5. Электронный адрес.
6. Имеется ли возможность выхода в Интернет.
7. Уровень конкурса (I, II).
8. Номер платежного поручения (либо другую информацию, подтверждающую перевод денег).

Список всех зарегистрированных участников будет доступен для ознакомления на сайте конкурса.

• Регистрация участников заканчивается в **23 часа 59 минут** в день, предшествующий началу конкурса.

• Всем участникам конкурса (то есть подавшим заявки), имеющим выход в Интернет, сообщается пароль, с помощью которого на сайте конкурса можно будет взять задания с программными инструментами для их выполнения, а также сдать решения. В тех случаях, когда нет выхода в Интернет, задания с программными инструментами будут высылаться (а решения приниматься) по электронной почте.

• Последний срок приема решений: 23 часа 59 минут последнего дня конкурса.

Примечание. Зарегистрированные участники, не имеющие электронной почты и выхода в Интернет, могут получить диски с заданиями в редакции журнала.

Каждый участник конкурса получит результаты своего участия в конкурсе. Кроме того, результаты конкурса будут опубликованы на сайте конкурса.

Всю информацию о конкурсе можно получить по телефону (812) 234-68-40, по электронной почте *kio@ipo.spb.ru* и в редакции журнала “Компьютерные инструменты в образовании”.

Учителю информатики: памятные даты января

Окончание. См. с. 1–2

matica", переводила на французский язык работы русских математиков, в частности, П.Л. Чебышёва.

Затронутые ею вопросы математики и механики развивались потом многими учеными. Например, Н.Е. Жуковский дал геометрическое истолкование случая Ковалевской в задачах о вращении твердого тела, а Н.Б. Делоне построил модель гироскопа Ковалевской. Проводил интересные исследования и А.М. Ляпунов.


Джон Эккерт

В январе 1944 года один из создателей первого успешно функционировавшего электронного цифрового компьютера (ENIAC) Джон Эккерт выдвинул идею хранимой программы.

Суть этой революционной для компьютерной техники идеи в том, что "*программы ЭВМ должны храниться в ее внутренней памяти наравне с исходными данными и промежуточными результатами вычислений* и представляться числовым кодом, что позволяет выполнять над ними те же операции, что и над числами" [6].

Литература

1. *Самарин М.С.* Вольт, ампер, ом и другие. Единицы физических величин в технике связи. М.: Радио и связь, 1988.

2. *Храмов Ю.А.* Физики. Биографический справочник. Изд. 2-е. М.: Наука, Гл. редакция физико-математической литературы, 1983.

3. *Ляпунов А.А.* Основные проблемы машинного перевода // Вопросы языкознания № 5, 1956 (совместно с П.С. Кузнецовым и А.А. Реформатским). Вкл. в сб.: *Ляпунов А.А.* Проблемы теоретической и прикладной кибернетики. М.: Наука, 1980.

4. *Кнут Д.* Искусство программирования: Пер. с англ. Т. 1: Основные алгоритмы. Изд. 3-е, т. 2: Получисленные алгоритмы. Изд. 3-е, т. 3: Сортировка и поиск. Изд 2-е. М.: Издательский дом "Вильямс", 2000.

5. *Воронцова Л.А.* Софья Ковалевская. М.: Молодая гвардия, 1959.

6. *Полунов Ю.Л.* От абака до компьютера: судьбы людей и машин. Книга для чтения по истории вычислительной техники в двух томах. Т. I. М.: Издательско-торговый дом "Русская Редакция", 2004.

7. *http://www.ict.nsc.ru/ws/Lyap2001/2263/*

ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»
главный редактор – А.С. Соловейчик

ГАЗЕТЫ ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября
гл. ред. — Е.В. Бирюкова, индекс подписки — 32024;
Английский язык
гл. ред. — Е.В. Громушкина, индекс подписки — 32025;
Библиотека в школе
гл. ред. — О.К. Громова, индекс подписки — 33376;
Биология
гл. ред. — Н.Г. Иванова, индекс подписки — 32026;
География
гл. ред. — О.Н. Коротова, индекс подписки — 32027;
Дошкольное образование
гл. ред. — М.С. Аромштам, индекс подписки — 33373;
Здоровье детей
гл. ред. — Н.В. Сёмина, индекс подписки — 32033;
Информатика
гл. ред. — С.Л. Островский, индекс подписки — 32291;
Искусство
гл. ред. — М.Н. Сартан, индекс подписки — 32584;
История
гл. ред. — А.Л. Савельев, индекс подписки — 32028;
Литература
отв. сек. — С.Ф. Дмитренко, индекс подписки — 32029;
Математика
и.о. гл. ред. — Л.О. Рослова, индекс подписки — 32030;
Начальная школа
гл. ред. — М.В. Соловейчик, индекс подписки — 32031;
Немецкий язык
гл. ред. — М.Д. Бузоева, индекс подписки — 32292;
Русский язык
гл. ред. — Л.А. Гончар, индекс подписки — 32383;
Спорт в школе
гл. ред. — О.М. Леонтьева, индекс подписки — 32384;
Управление школой
гл. редактор — Я.А. Сартан, индекс подписки — 32652;
Физика
гл. ред. — Н.Д. Козлова, индекс подписки — 32032;
Французский язык
гл. ред. — Г.А. Чесновицкая, индекс подписки — 33371;
Химия
гл. ред. — О.Г. Блохина, индекс подписки — 32034;
Школьный психолог
гл. ред. — И.В. Вачков, индекс подписки — 32898.

Гл. редактор
С.Л. Островский
Зам. гл. редактора
А.И. Сенокосов
Редакция
Е.В. Андреева
Д.М. Златопольский (редактор вкладки "В мир информатики")
Л.Н. Картвелишвили
С.Б. Кишкина
Н.П. Медведева
Ю.А. Первин (редактор вкладки "Началка")
Корректор Е.Л. Володина
Дизайн и верстка Н.И. Пронская

Адрес редакции и издателя:
Киевская, 24, Москва, 121165
тел. 249-48-96
Отдел рекламы: 249-98-70

Учредитель: ООО "Чистые пруды"

Зарегистрировано в Министерстве РФ по делам печати. ПИ № 77-7230 от 12.04.2001.
Отпечатано в ОИД "Медиа-Пресса", ул. Правды, 24, Москва, ГСП-3, А-40, 125993
Тираж 6000 экз.
Срок подписания в печать по графику 15.12.2005.
Номер подписан 12.12.2005.
Заказ № 615501
Цена свободная

ИНДЕКС ПОДПИСКИ
для индивидуальных подписчиков 32291
комплекта изданий 32744

Тел.: (095) 249-31-38, 249-33-86. Факс (095)249-31-84

Internet: inf@1september.ru
WWW: http://www.1september.ru

ИЗДАТЕЛЬСКАЯ ПОДПИСКА Тел.: (095) 249-47-58 E-mail: podpiska@1september.ru