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

ИНФОРМАТИКА

внутри номера CD и код доступа к электронной версии

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

НА ОБЛОЖКЕ

Поиск подстроки в строке — задача, с которой каждый пользователь компьютера имеет дело по много раз в день. Конечно, мало кого интересует, что там на самом деле происходит при нажатии на кнопку “Найти”. Лишь бы поиск был быстрым. Обычный линейный поиск для небольших текстов вполне годится, но его производительность не всегда может удовлетворить при решении производственных задач. Поэтому было придумано немалое количество быстрых и производительных алгоритмов. Один из самых известных из них назван в честь трех авторов — Д.Кнута, В.Пратта и Д.Морриса.

В НОМЕРЕ

НА ДИСКЕ

ЭЛЕКТРОННЫЕ МАТЕРИАЛЫ:

- Дистрибутивы свободно распространяемых программ, упомянутых и используемых в статьях номера
- Исходные файлы, упомянутые в статьях, коды программ
- Презентации ко всем статьям

ИНФОРМАТИКА

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

http://inf.1september.ru

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

РЕДАКЦИЯ:
гл. редактор С.Л. Островский
редакторы
Е.В. Андреева,
Д.М. Златопольский (редактор вкладки “В мир информатики”)
Дизайн макета И.Е. Лукьянов
верстка Н.И. Пронская
корректор Е.Л. Володина
секретарь Н.П. Медведева
Фото: фотобанк Shutterstock
Журнал распространяется по подписке
Цена свободная
Тираж 10140 экз.
Тел. редакции: (499) 249-48-96
E-mail: inf@1september.ru
http://inf.1september.ru

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

ГАЗЕТА ИЗДАТЕЛЬСКОГО ДОМА
Первое сентября – Е.Бирюкова

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

УЧРЕДИТЕЛЬ:
ООО “ЧИСТЫЕ ПРУДЫ”

Зарегистрировано ПИ № ФС77-44341 от 22.03.2011 в Министерстве РФ по делам печати
Подписано в печать: по графику 12.01.2012, фактически 12.01.2012
Заказ №
Отпечатано в ОАО “Чеховский полиграфический комбинат” ул. Полиграфистов, д. 1, Московская область, г. Чехов, 142300

АДРЕС ИЗДАТЕЛЯ:
ул. Киевская, д. 24, Москва, 121165
Тел./факс: (499) 249-31-38
Отдел рекламы: (499) 249-98-70
http://1september.ru

ИЗДАТЕЛЬСКАЯ ПОДПИСКА:
Телефон: (499) 249-47-58
E-mail: podpiska@1september.ru

Черный квадрат современных технологий

Современному человеку, к примеру, на отдыхе приходится прилагать немало усилий, чтобы хоть на какое-то время вырваться из липкой паутины информационного общества. Представьте себе, что идете вы, допустим, по лесу — черт-те где, в другой стране. Тишина звенящая, сугробы по пояс, следы — только звериные. Мечта! ☺ Но телефон-то в кармане работает. И не просто телефон, а со встроенным GPS-навигатором. Не выключишь — штука-то в лесу полезная. А раз GPS-навигатор есть, значит, и другие функции смартфона имеются. И пока вы дышите воздухом, ваша электронная почта исправно проверяется, в скайпе выставлен статус “доступен”, такие же статусы в Google Talk’е и в чате Facebook’а, пришедшую в голову “умную мысль” можно немедленно “твитнуть” — в общем, тянутся к вам ниточки Всемирной паутины ☺. Для многих из нас такая картина является вполне привычной — различаются лишь набор и количество сервисов, которыми мы пользуемся, но в конце концов не так уж важно, сколько ниточек к вам протянуто, хватит и одной.

Но что меня “добило” во время такой зимней прогулки, так это самый обычный аккуратный деревянный столбик, который спокойно стоял посреди леса. На столбике имелся QR-код, просканировав который можно было узнать не только об истории этого места, но и о ближайших достопримечательностях, включая (простите за подробности, но не могу не поделиться!) расположение ближайших туалетов ☺. Текст был на двух языках — на языке страны, установившей столбик, и на английском.

QR-коды — по меркам развития технологий, достаточно старое изобретение, но в России они стали широко использоваться не так давно. Характерные квадратики можно встретить на товарах, в магазинах, в гостиницах, на транспортных узлах, в рекламных материалах.

Аббревиатура “QR” образована от слов *quick response*, что переводится как “быстрый отклик”. Для сканирования таких кодов требуются специальные сканеры, но при этом алгоритмы так просты и хорошо разработаны, что функцией сканирования QR-кодов обладают даже обычные камеры мобильных телефонов. Иногда требуется установить для этого специальное ПО, а иногда не требуется и этого. Появились QR-коды в Японии в 2000 г., и сегодня на обычной улице обычного японского города их можно увидеть повсюду — куда ни глянь.

В QR-код можно уместить достаточно много информации. Максимальное количество символов, которые помещаются в один QR-код:

Цифры — 7089.

Цифры и буквы (включая кириллицу) — 4296.

Двоичный код — 2953 байта.

Иероглифы — 1817.

Наибольшее количество приложений с использованием QR-кодов имеется для мобильных устройств. В частности, широко распространены программы, которые отображают в виде кодов контакты из записной книжки телефона. Чтобы поделиться контактом, достаточно просто просканировать (обычной камерой телефона) изображение на экране.

Кстати, немалая часть текста этой заметки может уместиться в один QR-код. Может быть, подумать о том, чтобы издавать журнал в QR-виде? ☺

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

О чем это?
В новом модульном учебнике базового курса информатики под редакцией А.В. Горячева (учебник получил гриф и будет издан к следующему учебному году) имеется необычный модуль “Управление личными проектами”. Таких материалов ранее не было в учебниках информатики, но важность их очевидна, а если не мы, то кто? ☺

И это все?!
На диске имеются дистрибутивы свободно распространяемых программ для работы с интеллект-картами и диаграммами Ганта. Также на диске имеются все файлы, упомянутые в статье.

Введение в управление проектами

С.Л. Островский,
Москва

Урок по теме “Цели”

Постановка проблемы урока

Беседовали как-то кот Матроскин с Шариком. Шарик и говорит: “Хочу я, Матроскин, завести корову. Что думаешь?”. — “Думаю я, что дело хорошее, но только неправильно ты, Шарик, цель формулируешь, неконкретно. Так мы еще долго без молока жить будем”.

Как вы думаете, почему Матроскин считает, что они с Шариком еще долго без молока жить будут? Что бы вы посоветовали Шарику?

Хочу!

Хотеть не вредно! Пожалуй, скорее вредно ничего не хотеть. Но с точки зрения целей данного урока важен лишь один вопрос: как хотеть правильно? Потому что целью мы будем считать именно правильное “хочу”. Например, “хочу прочитать книгу” практически никуда не годится и целью являться не может. Как и “хочу сходить в кино”. Но бывают “хочу” и еще хуже! Например, “хочу стать хорошим человеком”, несмотря на явно позитивную направленность желания, является никудышным “хочу”.

А вот “хочу прямо сейчас понять, зачем авторы “обижают” такие понятные и приятные хочу”, уже лучше! Так каким же должно быть правильное “хочу”?

SMART-хочу

Аббревиатура “SMART” образована от первых букв пяти английских слов:

- S — ***S****pecific* — конкретный;
- M — ***M****easurable* — измеримый;
- A — ***A****chievable* — достижимый;
- R — ***R****elevant* — насущный;
- T — ***T****imed* — определенный во времени.

В каждом случае в качестве перевода мы использовали наиболее подходящие по смыслу русские слова.

SMART-хочу, они же SMART-цели, они же просто цели, поскольку любая цель должна быть SMART, — широко распространенный критерий проверки целей на правильность. Правильная цель, как правило, обладает всеми перечисленными характеристиками. Говоря более формально, правильная цель описывается формулой “S and M and A and R and T”. Что же означают все эти пять критериев?

Конкретность

Требование конкретности цели подразумевает, что вы в точности понимаете, чего именно хотите. “Прочитать книгу” — не конкретная цель. Какую именно книгу? Детектив? Толстую? Красную? Ставя цель “прочитать книгу”, вы должны четко назвать самому себе ту самую конкретную книгу, которую хотите прочитать.

Измеримость

Требование измеримости предполагает, что вы умеете точно различать состояния, когда цель достигнута, а когда еще нет. Возможно, вы даже понимаете, что значит находиться “далеко от цели”, “на полпути” или “близко к цели”. Это не очень важно и не является обязательным. Главное — вам должны быть известны характеристики состояния, когда цель достигнута.

Например, если вашей целью является “прочитать данную книгу”, можно считать, что цель будет достигнута, когда вы закончите чтение последней страницы. Если цель — после окончания школы поступить в данное учебное заведение, вы можете выбрать критерием достижения, например, наличие вашей фамилии в списках зачисленных. Или наличие у вас в руках студенческого билета — как хотите. Лишь бы такой параметр вы для себя определили.

Достижимость

С достижимостью на самом деле все не так просто. Известны примеры, когда люди ставили себе, казалось бы, недостижимые цели и достигали их! Но ключевое слово в предыдущем предложении — “казалось бы”. Ведь если цели были достигнуты, значит, они были достижимыми! Достижимость не предполагает того, что мы должны отказываться от целей трудных и амбициозных. Но не следует формулировать явно недостижимые цели.

Хотите в течение ближайших десяти лет побить мировой рекорд в беге на 100 метров? Поставьте себе такую цель! Возможно, она кажется невероятной, но кто-то скорее всего побьет этот рекорд. Почему не вы? Но не стоит ставить целью пробежать стометровку быстрее одной секунды.

Раздумывая над достижимостью цели, полезно иметь в виду следующее: достижение вашей цели должно зависеть именно от вас. В формулировках целей следует избегать упоминаний о том, что “если случится то-то…”, “если кто-то сделает нечто…”. Подобные условия можно включать в стратегию (мы обсудим это понятие на следующем уроке), но не в формулировку цели.

Насущность

Требование насущности подразумевает, что вы должны думать о цели, как о вашей собственной цели. Даже если цель перед вами поставили. В последнем случае, с момента, когда вы согласились с поставленной целью (вынужденно, не вынужденно — не имеет значения), она становится вашей.

Вам нужно прочитать данную книгу к уроку литературы? Это — ваша цель. Считаете, что не ваша? Нет проблем, не читайте. Смело получайте двойку! (Тс-с-с, учитель литературы нас не слышит?) Но если решили, что должны, — прочь сомнения, цель теперь ваша!

Измеримость во времени

Вам нужно прочитать данную книгу к ближайшему уроку литературы? Это и есть измеримость цели во времени. Нет смысла ставить цели, даже правильные в остальных отношениях, “вообще”. Всегда должно быть определено время их достижения.

Не навреди

Последний критерий правильности цели не относится к SMART-критериям, но его, возможно, следовало бы поставить на первое место. Достижение вами цели не должно принести кому-то вреда.

Обобщение новых знаний

Не всякое “хочу!” является целью. Цели должны удовлетворять SMART-критериям.

Применение знаний

- В электронной таблице с названием “Хочу” (точное расположение файла вам укажет учитель) находятся “правильные” и “неправильные” цели. Справа от цели расположены столбцы с заголовками S, M, A, R, T, соответствующие SMART-критериям. Проанализируйте каждую цель и занесите в соответствующие столбцы значения 0 или 1 в зависимости от того, подходит ли данная цель под соответствующий критерий.
- В электронной таблице с названием “Ремонт целей” (точное расположение файла вам укажет учитель) находятся “неправильные” цели. Поработайте в парах. Обсудите, почему именно данные цели неправильные, какие из SMART-критериев для них не выполняются. Отремонтируйте их таким образом, чтобы они стали правильными.

Урок по теме “Стратегии”

Постановка проблемы урока

Шарик научился правильно ставить цели. И цель он выбрал себе достойную: завести настоящую голландскую корову. А так как молока очень хочется, сделать это достаточно быстро, максимум за пару месяцев. Сказано — сделано… Сделано? А как? Легко сказать! Заказать по почте у почтальона Печкина? Самому ехать в Голландию? Отправить в Голландию Матроскина? Как вы думаете, какой вопрос мы будем рассматривать на уроке?

Из точки А в точку Б

На прошлом уроке мы научились правильно ставить цели. После того как цель поставлена, необходимо определиться с тем, каким *путем* мы *планируем* ее достигнуть. Цель всегда расположена в будущем, это точка Б, которой необходимо достичь. А мы всегда живем в настоящем — в точке А, из которой мы начнем путь. Пути достижения цели могут быть разными. Избранный путь и есть *стратегия* достижения цели.

Допустим, ваша цель — в течение жизни совершить один космический полет. Каким образом можно достичь этой цели? Вот несколько вариантов:

- стать военным летчиком, попасть в отряд космонавтов;
- стать инженером космических аппаратов, инженеры тоже летают в космос;
- стать врачом и заниматься проблемами космической медицины. Врачи тоже совершают космические полеты;
- стать бизнесменом, заработать огромные деньги и купить место космического туриста.

Возможно, имеются и другие варианты, мы хотели показать главное — путей достижения одной и той же цели может быть много, и пути эти могут быть очень разными.

Для иллюстрации понятия стратегии часто используют наглядный визуальный образ, подобный показанному на *рис*. 1.

Рис. 1. *Обычная дорога в горах Норвегии*

Это очень хорошая картинка, надо только понимать все упрощения данной модели. Прежде всего она двухмерна — перед нами картинка на плоскости. Даже если представлять себе, что на этой иллюстрации изображен наш привычный трехмерный мир, ситуация принципиально не меняется — в реальной жизни измерений (существенных параметров), как правило, значительно больше. Но ось времени имеется всегда.

А и Б сидели на трубе. Вместе

Итак, точка Б — желаемая цель, точка А — наше настоящее. После того как мы выстроим стратегию достижения точки Б, эта стратегия свяжет их. А до этого? На самом деле связь между точками А и Б должна быть ясна еще на этапе целеполагания. Среди SMART-критериев “правильных” целей имеется критерий достижимости. Именно он и связывает точки А и Б еще до выработки стратегии. Если мы думаем, что цель достижима, значит, на этапе целеполагания мы считаем, что в принципе можем попасть из точки А в точку Б.

От Я до А. И далее в Б… или в $Б_1$

Где я? Обсуждая цель, как желаемую точку Б, мы нередко считаем, что уж с точкой А проблем нет. Действительно, какие могут быть проблемы? Вот мы, здесь, в точке А. Но точно ли мы понимаем, где мы? Правильно ли мы оцениваем параметры точки А, а значит, свои ресурсы и возможности? Четкое понимание не только финишной, но и стартовой точки — важнейшая составляющая процесса разработки стратегии.

Нередко при детальном изучении точки А становится понятно, что необходимо скорректировать и целевую точку Б по причине ее недостижимости. Это нормальный процесс! Если мы собираемся за лето прочитать N книг суммарным объемом M страниц, а при оценке личных ресурсов понимаем, что при текущей скорости чтения не успеваем прочитать более 0.01*M в день, то за 90 летних дней вы никак не сможете достичь цели.

Хотя… Может быть, имеет смысл пойти на курсы скорочтения, закончить их в течение десяти дней и после этого читать уже в два раза быстрее — по 0.02*M страниц в день. И тогда оставшихся 80 дней вполне хватит для того, чтобы прочитать требуемый объем.

Это очень простой пример, но он имеет прямое отношение к выработке стратегии, включая постановку цели и оценку ресурсов. В данном случае нам не пришлось корректировать точку Б, но в реальности цели чаще корректируются, чем остаются неизменными.

Хотя… а точно ли мы сформулировали цель? Действительно ли она в том, чтобы именно прочитать то, что написано на указанных страницах? Может быть, нам на самом деле требуется лишь узнать о том, что там написано? Меняет ли это дело? Ведь в таком случае можно задуматься об использовании аудиокниг. И тогда уже можно исходить не из скорости чтения, а из параметров прослушивания аудиозаписей. Вот, возможно, мы и пришли к корректировке цели.

SWOT-анализ: первая помощь голове

Вдумчивое целеполагание и тщательный подход к выработке стратегии достижения окупятся результатом. Но правильное планирование — весьма ресурсоемкий процесс. От него просто голова пухнет! Куда как проще: *просто чего-то захотел*, решил, что, *наверное*, *вот так это получится* — и сломя голову вперед!

Для того чтобы облегчить работу по уточнению целей, оценке текущего состояния и выработке стратегий, придумано немало интеллектуальных инструментов — специальных технологий умственного труда. Практически все эти инструменты были придуманы в докомпьютерные времена и требуют лишь чистого листа бумаги (иногда большого) и карандаша (иногда карандашей разных цветов). Применение же компьютерных технологий позволяет повысить эффективность и технологичность самих инструментов.

Мы познакомимся с одним из самых простых, но, одновременно, самым известным и мощным инструментом из указанных — SWOT-анализом. Аббревиатура “SWOT” образована от первых букв четырех английских слов:

- S — ***S**trength* — сила;
- W — ***W**eakness* — слабость;
- O — ***O**pportunity* — возможность;
- T — ***T**hreat* — угроза.

В докомпьютерные времена основным инструментом SWOT-анализа была обычная табличка 2 × 2 — на листе бумаге, на доске, на флип-чарте. Вот как выглядит типичная “бумажная” таблица для SWOT-анализа:

Цель	
S	W
O	T

Обратите внимание, что **SWOT-анализ всегда проводится “под” заранее сформулированную цель**. В процессе SWOT-анализа цель может уточняться и корректироваться, но нельзя использовать эту технологию без предварительного понимания цели.

В четырех ячейках таблицы необходимо выписать, соответственно, **свои** сильные и слабые стороны и **внешние** (предоставляемые извне) возможности и угрозы с точки зрения достижения поставленной цели. Еще раз: цель — это самое важное. Одно и то же может быть “плюсом” или “минусом”, сильной или слабой стороной. Если вы хотите стать баскетболистом, то рост за два метра — ваша сильная сторона, если танкистом — слабая.

Сегодня SWOT-анализ часто выполняют посредством компьютерных инструментов. Есть даже специализированные программы для SWOT-анализа, но многие предпочитают пользоваться универсальным инструментом — интеллект-картами.

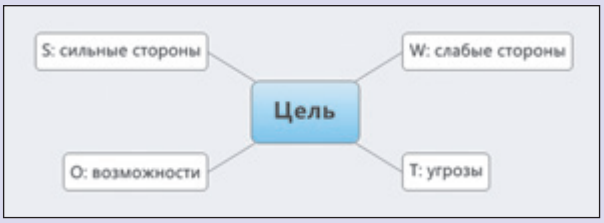


Рис. 2. Заготовка для проведения SWOT-анализа с использованием интеллект-карт

Интеллект-карты

Интеллект-карты являются одновременно и тривиальным, и мощнейшим инструментом, помогающим упорядочить процессы мышления и принятия решений. Как и многие другие средства интенсификации умственного труда, интеллект-карты были изначально придуманы в бумажном исполнении — первая книга Тони Бьюзена, в которой описывалась технология интеллект-карт, вышла в 1974 г. Сами по себе интеллект-карты не делают ничего. Они служат лишь для записи и визуализации наших мыслей и ассоциаций. Наверняка практически каждый, раздумывая над тем или иным вопросом, не раз рисовал на обычном листочке подобие интеллект-карты, но просто не знал, что “говорит прозой”.

С формальной точки зрения интеллект-карта представляет собой дерево с корнем в центре карты. Большинство современных компьютерных программ построения интеллект-карт умеют представлять карту и в виде настоящего дерева, но в классических картах корень расположен именно в центре.

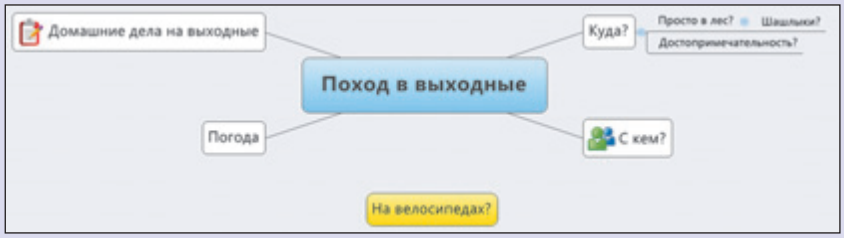


Рис. 3а. Фрагмент интеллект-карты для планирования похода. Классическая карта

Рис. 3б. Фрагмент интеллект-карты для планирования похода. Схема

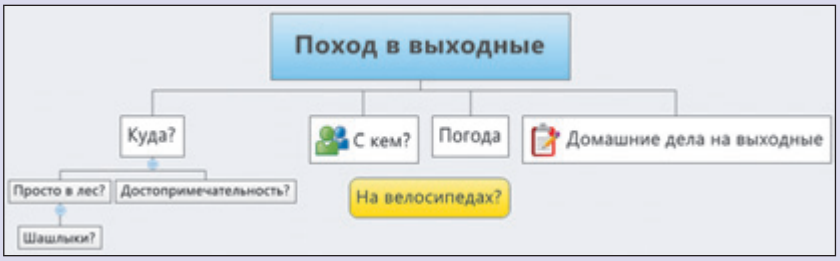


Рис. 3в. Фрагмент интеллект-карты для планирования похода. Дерево в виде "организационной диаграммы"

Рис. 3г. Фрагмент интеллект-карты для планирования похода. Дерево в виде "дерева каталогов"

Помимо узлов дерева, на карте могут быть так называемые "свободные узлы" (пример — узел "На велосипедах?"). Особенно удобно использовать их на начальном этапе размышлений, когда структура карты еще не сложилась, а надо просто поскорее записать некоторую мысль "на будущее". Свободные узлы в любой момент можно включить в дерево, равно как и любой узел можно "вынуть" из дерева или переставить на другое место.

Пример SWOT-анализа

Вернемся к SWOT-анализу. Рассмотрим реальный пример личного SWOT-анализа, который провел в начале учебного года ученик 9-го класса школы, имевший целью сдать один из международных экзаменов на знание английского языка — экзамен FCE (***First Certificate English***).

Рис. 4. Пример личного SWOT-анализа

На конкретном примере еще раз обратим внимание на следующее:

- SWOT-анализ выполнялся под заранее заданную цель;
- Сильные стороны (S) и слабости (W) — внутренние свойства, личные качества человека;
- Возможности (O) и угрозы (T) — внешние обстоятельства;
- Одно и то же может быть как сильной, так и слабой стороной, как возможностью, так и угрозой. В данном случае человек отнес текущий уровень знания языка к своим сильным сторонам, имея в виду, что впереди еще почти целый год. Если бы за окном была весна, а экзамен сдавать летом, то скорее всего такой уровень стал бы уже слабой стороной.

От SWOT-анализа к стратегии

Сам по себе SWOT-анализ не преподносит готовую стратегию "на блюдечке" — он лишь делает ясной и полной картину. При решении бизнес-задач за анализом следует целый ряд обязательных шагов. Но личный SWOT-анализ нередко позволяет сразу определиться со стратегией достижения цели — полной картины часто бывает достаточно для принятия решения. Тем не менее в любом случае, даже если после анализа стратегия ясна, полезно, выявив собственные слабости и внешние угрозы, понять, каким образом можно им противодействовать.

Возможно, вам будет интересно, какую стратегию избрал наш девятиклассник? Он принял решение в первую очередь освоить навыки самоорганизации, приобрести литературу по подготовке к экзамену и готовиться самостоятельно, активно используя Интернет и смотря телевизионные программы на английском языке. Кроме того, он решил скор-

Слабость	Что делать?
Очень плохо понимаю язык “на слух”	Слушать песни на английском, слушать радио на английском, смотреть телепередачи на английском
Плохо говорю на языке	Найти англоязычного собеседника в Интернете для разговоров по Скайпу
Плохо пишу на языке	Найти англоязычного друга по переписке, участвовать в англоязычных форумах в Интернете
Большая загруженность в школе, мало свободного времени	Эффективнее использовать время, освоить навыки организации времени
Плохие навыки самоорганизации, не умею самостоятельно планировать свою работу	Освоить навыки самоорганизации

Угроза	Что делать?
Ограниченный бюджет, родители не могут выделить более $1000 в год	Изыскать возможность работы в летний период

ректировать цель и сдавать экзамен не будущим летом, а через год. А следующим летом в период каникул поработать. В результате денег, которые смогли выделить родители и заработанных им самим, хватило для поездки за границу для изучения языка на заключительной стадии подготовки методом погружения. А это, в свою очередь, позволило вместо экзамена начального уровня FCE сдать экзамен следующего уровня CAE (***C**ertificate in **A**dvanced **E**nglish*). Такова реальная история, которая начиналась с личного SWOT-анализа.

Стратегия не высечена в камне

В заключение еще раз обратим внимание на то, о чем мы уже говорили. Ни цели, ни стратегии не высечены в камне. Можно и нужно менять при необходимости и то и другое. Инструменты, подобные рассмотренному SWOT-анализу, не принимают за вас решения, но позволяют вам самим принимать взвешенные и осознанные решения.

Обобщение новых знаний

Стратегия — путь достижения цели. Для выработки стратегии необходимо не только ясно сформулировать цель, но и объективно оценить текущее положение дел. Для структуризации собственных мыслей при разработке стратегий можно использовать SWOT-анализ. При проведении личного SWOT-анализа важно следить за тем, чтобы сильные и слабые стороны были личными свойствами человека, а возможности и угрозы — внешними факторами. Результатом SWOT-анализа может стать корректировка цели. Также цели могут корректироваться в процессе реализации самой стратегии.

Применение знаний

В файле с названием “SWOT.xmind” (точное расположение файла вам укажет учитель) находится заготовка для проведения SWOT-анализа, показанная на *рис*. 2. Поработайте в парах. Сформулируйте по одной цели для каждого из вас (на выбор). Работайте с целями последовательно — сначала с одной, затем с другой. Проверьте, что цель удовлетворяет SMART-критериям, и при необходимости скорректируйте ее. Проведите SWOT-анализ для выработки стратегии достижения цели. По результатам анализа при необходимости скорректируйте цель и предложите стратегию ее достижения.

Урок по теме “Производственные проекты”

Постановка проблемы урока

Задумали Шарик с Матроскиным новый дом строить. Даже Печкин им помочь вызвался. И дядя Федор обещал на недельку из города приехать. Сели кошка с собакой и стали думать — это же сколько всего предусмотреть надо! И кому что делать? И как не запутаться? Как вы думаете, какой вопрос мы будем рассматривать на уроке?

Вместо определения

На “бытовом уровне” под проектом часто понимают “достаточно большую” задачу, в которой выделены различные подзадачи. Конечно, указанных свойств для “настоящего” проекта мало. Но что такое “настоящий” проект?

Имеется несколько авторитетных источников, в которых даются неформальные, но достаточно строгие определения проектов. В России широко используются Национальные требования к компетенции менеджера проекта (НТК), разработанные российской национальной ассоциацией управления проектами, в мире — PMBOK — свод знаний в области управления проектами, разработанный Институтом управления проектами США. Указанные документы имеют много общего. Тем не менее определения проектов в них чуть разнятся.

Согласно НТК, проект представляет собой “уникальный комплекс взаимосвязанных мероприятий, направленных на создание продукта или услуги в условиях заданных требований и ограничений”.

В PMBOK под проектом понимается “временное предприятие (мероприятие), предназначенное для создания уникального продукта, услуги или результата”.

Наш модуль посвящен личным проектам. Известно, однако, что лучше всех личные проекты

реализуют обычно профессиональные руководители — менеджеры, которые привыкли управлять производственными проектами и с теми же требованиями и инструментами подходят к организации личных проектов. Поэтому и мы познакомимся с основами управления производственными проектами.

Скажем сразу — авторы не имеют возможности и не делают попытки уместить всю науку и практику управления проектами в рамки одного параграфа. Современный сертифицированный менеджер проекта имеет высшее образование, не менее 4500 часов работы в области управления проектами и несколько достаточно сложных сертификационных экзаменов за плечами. Но основы науки управления проектами, ключевые понятия и инструменты можно рассмотреть и в рамках одного параграфа.

Итак, вернемся к определениям понятия "проект". О чем в них говорится?

Во-первых, проект всегда имеет **цель** (одну или несколько — не так важно) и **результат** — проектный продукт на выходе. Во-вторых, проект всегда явление временное — у него есть фиксированные **начало** и **конец**. Это важно: текущая производственная деятельность предприятий не является проектной. Если завод постоянно выпускает некоторую продукцию, это не имеет отношения к проектной деятельности. Но если внедряется новая линия, новый производственный цикл, новый продукт — это проект. Отметим сразу, что имеются предприятия, для которых вся текущая деятельность проектная. Например, конструкторские бюро, разрабатывающие уникальные изделия.

Важным свойством проекта является наличие **ресурсов**. Обсуждать что-либо вообще имеет смысл лишь в условиях ограниченности ресурсов. Есть шутливый, но очень справедливый закон Лермана: любую техническую проблему можно решить, имея достаточно времени и денег. И из него есть следствие: вам никогда не будет хватать либо времени, либо денег.

Наконец, любой проект всегда несет в себе **риски**, которые обязательно учитываются при планировании. Этот факт не отражен в определении, но риски — важнейшая составляющая проекта.

Диаграммы Ганта

Для планирования проектов используются различные профессиональные инструменты. Мы познакомимся лишь с одним из них, зато самым, пожалуй, известным — редкий план проекта обходится без диаграмм Ганта. Генри Гант был менеджером и занимался процессами организации производства. Ключевая и новаторская в то время идея Ганта — описание данных и процессов посредством графических средств. В законченном и дошедшем до нашего времени виде диаграммы Ганта были придуманы им во время службы военным консультантом во время Первой мировой войны. Первая созданная Гантом "настоящая" диаграмма изображала пять изделий военных материалов и планирование требуемых заказов, выполнение заказов, а также наличие остатков на складе. В управленческой мысли того периода графические пособия Ганта были революционными для планирования управления и самого управления. На готовой графической форме менеджеры могли увидеть, как выполнять планы и предпринять необходимые действия, чтобы контролировать их выполнение вовремя и их бюджет. Все последующие доски и диаграммы контроля по производству были заимствованы у Ганта.

Проиллюстрируем идею Ганта на "плохом", но предельно простом примере: чтение большой книги. Допустим, мы имеем книгу из пяти глав и читаем по одной главе в день по рабочим дням, начиная с 1 сентября. В выходные — отдыхаем. Простейшая диаграмма Ганта, иллюстрирующая реализацию такого проекта, показана на *рис.* 5.

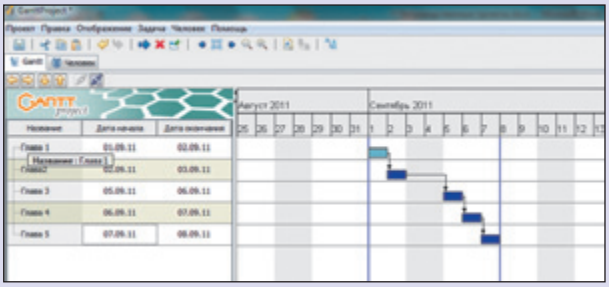


Рис. 5. *"Проект" по прочтению книги*

На диаграмме Ганта каждая задача представлена отрезком (прямоугольником, блоком — как удобнее визуализировать). В верхней части всегда располагается шкала времени. Связанные задачи, т.е. задачи, к выполнению которых можно приступить только после завершения предыдущих, показаны соединительными линиями со стрелочками. Вот и все главные элементы диаграмм Ганта.

• Введите в программу Gantt Project "проект" по прочтению книги, показанный на *рис.* 5.

Рассмотрим следующий пример. Допустим, вам надо прочесть две книги, в одной пять глав, в другой — шесть. И вы избрали следующий алгоритм — каждый рабочий день вы будете читать по одной главе из каждой книги. (Понятно, что на последний день останется только одна глава.) Этот план можно следующим образом визуализировать посредством диаграммы Ганта (см. *рис.* 6).

В этом примере внутри основного проекта были выделены два подпроекта.

В чем удобство диаграмм Ганта? Во-первых, они очень наглядны, а для сложных проектов это важ-

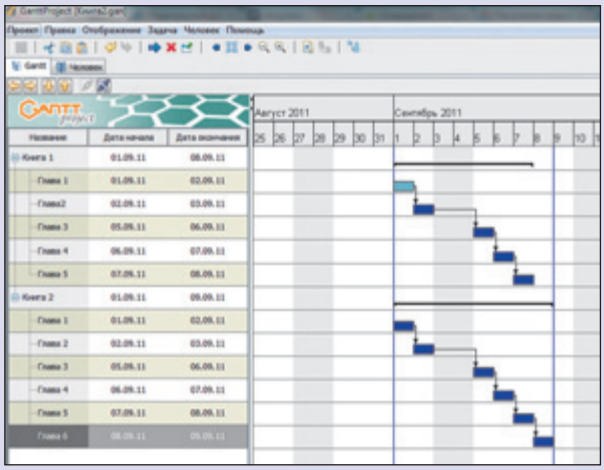


Рис. 6. “Проект” по прочтению двух книг

но — менеджеру необходимо иметь ясную картину развертки проекта по времени. Во-вторых, современные программные средства для построения диаграмм Ганта автоматически перестраивают диаграммы при изменении параметров задач проекта, например, при непредвиденных задержках. Если первая глава второй книги оказалась такой длинной (или занудной), что вместо одного рабочего дня вы потратили на нее три, диаграмма может быть автоматически перестроена (*рис*. 7).

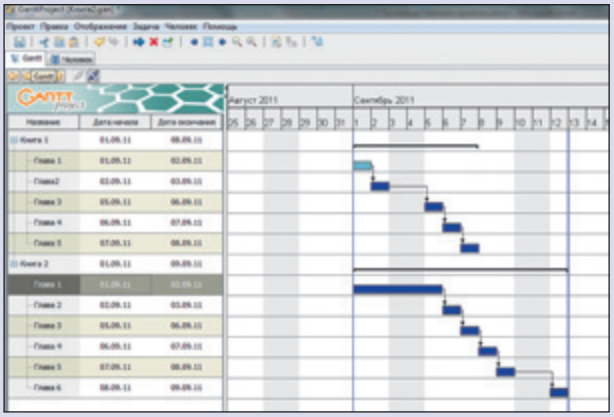


Рис. 7. Последствия задержки решения задачи по прочтению главы 1 книги 2

Пока мы справлялись с книгами в одиночку. А если мы такие хитрые, что договорились с другом о следующем: каждый прочитает по книге, а потом быстро “в двух словах” расскажет другому ее содержание. Правда ведь, здорово!

Все компьютерные программы, визуализирующие диаграммы Ганта, имеют разделы для управления ресурсами. Базовая идея простая: сначала надо описать в проекте некоторый ресурс, а потом его можно использовать для выполнения тех или иных задач. В программе Gantt Project раздел с ресурсами называется “Человек”, но на самом деле в этом разделе могут быть и группы людей — “бригады”. В данном случае для реализации проекта нам действительно нужны два человека.

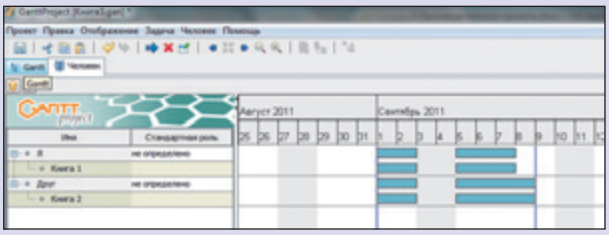


Рис. 8. Ресурсы проекта

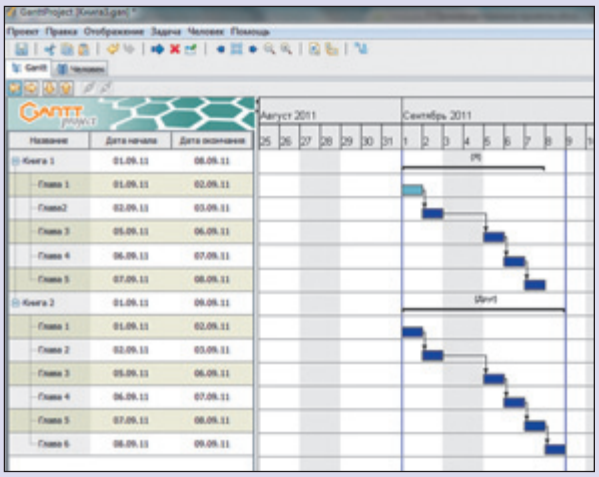


Рис. 9. Использование ресурсов проекта

• Введите в программу Gantt Project проект, показанный на *рис*. 9.

В заключение рассмотрим небольшой производственный проект по возведению трехэтажного дома. Допустим, вы руководите этим проектом и у вас имеются три бригады — “Возведение”, “Коммуникации” и “Отделка”. Сдавать дом комиссии вы будете сами. Бригада “Возведение” строит дом этаж за этажом. На возведение фундамента ей требуется 10 рабочих дней, каждого из трех этажей — по 5, крыша возводится за 3 дня. Отделочники приступают к работе, когда очередной этаж возведен. Фундамент не отделывается, на отделку каждого из этажей уходит по 10 дней, на чердак — 3 дня. Коммуникации подводятся независимо. Для подведения электричества и воды необходимо по 5 дней, на газ и канализацию — по 10. Считая, что к работе можно приступить завтра, введите в программу Gantt Project план возведения дома.

Обобщение новых знаний

Производственным проектом называют “уникальный комплекс взаимосвязанных мероприятий, направленных на создание продукта или услуги в условиях заданных требований и ограничений”. Важными свойствами проекта являются цели, сроки, ресурсы, риски. Для визуализации планов реализации проектов используют диаграммы Ганта, в которых указываются задачи, входящие в проект, их сроки и задействованные ресурсы.

О чем это?
Есть известный жизненный анекдот про то, как один юрист угостил другого апельсином: “Настоящим я передаю вам все принадлежащие мне права, требования, преимущества и другие интересы на собственность, именуемую апельсин, совместно со всей его кожурой, мякотью, соком и семечками, с правом выжимать, разрезать, замораживать и иначе употреблять, используя для этого любого рода приспособления, как существующие в настоящее время, так и изобретенные позднее, или без использования упомянутых приспособлений, а также передавать ранее именованную собственность третьим лицам с кожурой, мякотью, соком и семечками или без оных…”. Так вот, статья про это. Только применительно к IT.

И это все?!
На диске имеется презентация, которая поможет при использовании материалов статьи на уроке.

Основы информационной безопасности: обзорный урок

О.Б. Богомолова, д. п. н., учитель информатики и математики, Восточный округ г. Москвы

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

Объявленная тема урока относится к разряду достаточно сложных не столько для учеников, сколько для учителя. Как правило, учителя информатики располагают компетенциями в технической сфере (вычислительная техника и другие средства информационных и коммуникационных технологий) и в области теоретической информатики, но недостаточно “подкованы” в вопросах юридического характера.

Предлагаемый вниманию читателей материал к уроку содержит конкретные определения понятий, ссылки на статьи действующих федеральных законов, касающиеся вопросов информационной безопасности, а также классификацию программного обеспечения с точки зрения принципов распространения программ и описание типовых принципов коммерческой и свободной лицензионных политик. Авторы постарались собрать достаточное количество примеров для иллюстрации различных правонарушений в сфере информационной безопасности.

Авторы надеются, что этот материал поможет учителям информатики построить урок по данной теме, дополнив представленные теоретические материалы практическими и контрольными заданиями. Так, например, в качестве практического задания можно предложить учащимся проанализировать текст конкретной пользовательской лицензии и заполнить таблицу, определяющую права и обязанности сторон лицензионного договора (а также определить сами эти стороны), действия, которые пользователь может или же, наоборот, не имеет права производить с приобретенной им копией программы, меры ответственности за несоблюдение условий лицензионного договора и т.д. Практическое задание может также предполагать сравнительный анализ типовой коммерческой лицензии и лицензии GNU GPL по указанным выше позициям сравнения и т.д.

Что такое информационная безопасность

Под информационной безопасностью понимается состояние защищенности информационной среды. Соответственно, защита информации понимается как

действия по предотвращению возможного повреждения или уничтожения информации, а также несанкционированного доступа к ней (но вместе с тем — обеспечение беспрепятственного доступа к информации со стороны легитимных пользователей). Особо может рассматриваться информационная безопасность организации (как состояние защищенности информационной среды некоторой организации, например, образовательного учреждения), а также информационная безопасность государства (как состояние сохранности информационных ресурсов государства и защищенности законных прав личности и общества в информационной сфере).

В современных условиях информационная безопасность общества и государства может рассматриваться как совокупность информационно-технической и информационно-психологической (психофизической) безопасности.

Меры по обеспечению информационной безопасности можно разделить на *технические* (различные аппаратные и программные средства и технологии защиты от вредоносных программ, внешних сетевых атак и пр.; к ним, в частности, относятся все антивирусные программы) и *правовые*, под которыми понимается совокупность нормативных и правовых актов, регулирующих вопросы защиты информации.

Правовые основы информационной безопасности

Основным законом, касающимся вопросов использования и защиты информации, в нашей стране является *Федеральный закон № 149-ФЗ "Об информации, информационных технологиях и о защите информации"*, принятый 27 июля 2006 г. взамен ранее действовавшего Федерального закона № 24-ФЗ "Об информации, информатизации и защите информации" от 20 февраля 1995 г.

В частности, статья 1 Закона "Об информации, информационных технологиях и о защите информации" гласит, что данный закон "регулирует отношения, возникающие при:

1) осуществлении права на поиск, получение, передачу, производство и распространение информации;

2) применении информационных технологий;

3) обеспечении защиты информации".

Статья 3 формулирует понятия "безопасность" и "защита информации" в рамках принципов правового регулирования отношений в сфере информации, информационных технологий и защиты информации. Причем в качестве одного из основных принципов постулируется "обеспечение безопасности Российской Федерации при создании информационных систем, их эксплуатации и защите содержащейся в них информации".

А вот статья 10, носящая название "Распространение информации или предоставление информации", уже напрямую касается проблемы спам-рассылок:

- пункт 2: "Информация, распространяемая без использования средств массовой информации, должна включать в себя достоверные сведения о ее обладателе или об ином лице, распространяющем информацию, в форме и в объеме, которые достаточны для идентификации такого лица";
- пункт 3: "При использовании для распространения информации средств, позволяющих определять получателей информации, в том числе почтовых отправлений и электронных сообщений, лицо, распространяющее информацию, обязано обеспечить получателю информации возможность отказа от такой информации";
- пункт 4: "Предоставление информации осуществляется в порядке, который устанавливается соглашением лиц, участвующих в обмене информацией".

Следовательно, противозаконной является рассылка любой информации, в которой отсутствуют точные и достоверные сведения о ее распространителе, причем даже при наличии таких сведений пользователь должен иметь возможность отказа от получения этой информации. Однако если вы сами где-либо подписались на электронную рассылку, то ее отправка вам становится законной в соответствии с любыми теми условиями, которые указаны в правилах этой рассылки или в соответствующем "публичном договоре". Поэтому, прежде чем нажать кнопку "подписка" на каком-нибудь сайте, обязательно прочтите внимательно соответствующие правила!

Далее, статья 16 Закона "Об информации, информационных технологиях и о защите информации" рассматривает уже основные понятия и положения, касающиеся защиты информации:

- пункт 1: "Защита информации представляет собой принятие правовых, организационных и технических мер, направленных на:

1) обеспечение защиты информации от неправомерного доступа, уничтожения, модифицирования, блокирования, копирования, предоставления, распространения, а также от иных неправомерных действий в отношении такой информации,

2) соблюдение конфиденциальности информации ограниченного доступа,

3) реализацию права на доступ к информации";

- пункт 2: "Государственное регулирование отношений в сфере защиты информации осуществляется путем установления требований о защите информации, а также ответственности за нарушение законодательства Российской Федерации об информации, информационных технологиях и о защите информации";
- пункт 4: "Обладатель информации, оператор информационной системы в случаях, установленных законодательством Российской Федерации, обязаны обеспечить:

1) предотвращение несанкционированного доступа к информации и (или) передачи ее лицам, не имеющим права на доступ к информации,

2) своевременное обнаружение фактов несанкционированного доступа к информации,

3) предупреждение возможности неблагоприятных последствий нарушения порядка доступа к информации,

4) недопущение воздействия на технические средства обработки информации, в результате которого нарушается их функционирование,

5) возможность незамедлительного восстановления информации, модифицированной или уничтоженной вследствие несанкционированного доступа к ней,

6) постоянный контроль за обеспечением уровня защищенности информации".

Эти положения закона относятся, например, к администрации и техническим службам организаций, поддерживающих размещенные в Интернете базы данных, к владельцам серверов для размещения пользовательских сайтов и т.д.

Наконец, статья 17 рассматриваемого Закона определяет ответственность за правонарушения в сфере информации, информационных технологий и защиты информации:

• пункт 1: "Нарушение требований настоящего Федерального закона влечет за собой дисциплинарную, гражданско-правовую, административную или уголовную ответственность в соответствии с законодательством Российской Федерации".

То есть любые деяния, приводящие к повреждению или уничтожению информации, к ее намеренному искажению, равно как и стремление получить неправомерный доступ к чужой конфиденциальной информации, — то, чем, собственно, и занимаются авторы вирусов, "троянов" и других вредоносных программ, — являются ***преступлением,*** со всеми вытекающими отсюда последствиями. О том, какими могут быть эти последствия, мы поговорим чуть позже.

Еще один законодательный акт — *Федеральный закон № 152-ФЗ "О персональных данных",* вступивший в действие 27 июля 2006 г., определяет основные понятия и положения о защите персональной, т.е. личной информации каждого человека. Так, статья 3 гласит, что персональные данные — это "любая информация, относящаяся к определенному или определяемому на основании такой информации физическому лицу (субъекту персональных данных), в том числе его фамилия, имя, отчество, год, месяц, дата и место рождения, адрес, семейное, социальное, имущественное положение, образование, профессия, доходы, другая информация". А согласно пункту 1 статьи 6, обработка таких персональных данных может осуществляться оператором только с согласия субъектов этих персональных данных, за исключением лишь особо оговоренных пунктом 2 этой статьи случаев. Тем самым еще раз подчеркивается противозаконность действий, приводящих к несанкционированному доступу и хищению личной информации пользователей, даже если такое хищение не привело к непосредственному материальному ущербу.

Что же грозит тем, кто, несмотря на предупреждения двух рассмотренных выше законов (причем незнание закона, как известно, не освобождает от ответственности!), будет создавать вредоносные программы или производить какие-либо вредоносные действия либо пытаться получить несанкционированный доступ к чужой информации? В России соответствующие меры наказания определены в *Уголовном кодексе Российской Федерации (УК РФ),* носящем номер 63-ФЗ и вступившем в действие 13.06.1996 г. Преступления, связанные с компьютерной информацией, рассмотрены в нем в главе 28, где содержатся три следующие статьи:

• статья 272. Неправомерный доступ к компьютерной информации;

• статья 273. Создание, использование и распространение вредоносных программ для ЭВМ;

• статья 274. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети.

Приведем здесь полностью их текст.

Статья 272

Неправомерный доступ к компьютерной информации

1. Неправомерный доступ к охраняемой законом компьютерной информации, то есть информации на машинном носителе, в электронно-вычислительной машине (ЭВМ), системе ЭВМ или их сети, если это деяние повлекло уничтожение, блокирование, модификацию либо копирование информации, нарушение работы ЭВМ, системы ЭВМ или их сети, — наказывается **штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев, либо исправительными работами на срок от шести месяцев до одного года, либо лишением свободы на срок до двух лет.**

2. То же деяние, совершенное группой лиц по предварительному сговору или организованной группой либо лицом с использованием своего служебного положения, а равно имеющим доступ к ЭВМ, системе ЭВМ или их сети, — наказывается **штрафом в размере от ста тысяч до трехсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период от одного года до двух лет, либо исправительными работами на срок от одного года до двух лет, либо арестом на срок от трех до шести месяцев, либо лишением свободы на срок до пяти лет.**

Статья 273

Создание, использование и распространение вредоносных программ для ЭВМ

1. Создание программ для ЭВМ или внесение изменений в существующие программы, заведомо при-

водящих к несанкционированному уничтожению, блокированию, модификации либо копированию информации, нарушению работы ЭВМ, системы ЭВМ или их сети, а равно использование либо распространение таких программ или машинных носителей с такими программами — наказываются **лишением свободы на срок до трех лет со штрафом в размере до двухсот тысяч рублей или в размере заработной платы или иного дохода осужденного за период до восемнадцати месяцев**.

2. Те же деяния, повлекшие по неосторожности тяжкие последствия, — наказываются **лишением свободы на срок от трех до семи лет**.

Статья 274

Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети

1. Нарушение правил эксплуатации ЭВМ, системы ЭВМ или их сети лицом, имеющим доступ к ЭВМ, системе ЭВМ или их сети, повлекшее уничтожение, блокирование или модификацию охраняемой законом информации ЭВМ, если это деяние причинило существенный вред, — наказывается **лишением права занимать определенные должности или заниматься определенной деятельностью на срок до пяти лет, либо обязательными работами на срок от ста восьмидесяти до двухсот сорока часов, либо ограничением свободы на срок до двух лет**.

2. То же деяние, повлекшее по неосторожности тяжкие последствия, — наказывается **лишением свободы на срок до четырех лет**.

Таким образом, создание вредоносных программ, несмотря на "виртуальный" характер, является реальным преступлением и карается реальными штрафами (чаще всего — с конфискацией соответствующего компьютерного оборудования), а в отдельных случаях — реальными тюремными сроками (и далеко не всегда условными).

Впрочем, в других странах наказание за информационные преступления бывают еще строже. Например, на Филиппинах в настоящее время принят закон, предусматривающий за разработку вирусов смертную казнь.

Заметим, кстати, что высокий уровень знаний принципов работы компьютера и операционных систем и программистских умений, которыми любят хвастаться хакеры, не спасает их от возмездия. Вот перечень фамилий целого ряда знаменитых хакеров, попавших, несмотря на все свои ухищрения, в руки правосудия:

- Кевин Митник — был арестован 15 февраля 1995 г. в городе Рэлей (Северная Каролина), когда его сумел выследить компьютерный эксперт Цитому Шимомура. На суде Митник признал свою вину по большинству предъявленных обвинений и был приговорен к 46 месяцам реального и трем годам условного заключения, а также к выплате значительного штрафа (Митник вышел на свободу из тюрьмы 21 января 2000 г.);
- Пьер-Ги Лавуа, 22-летний канадский хакер, согласно канадским же законам, был приговорен к 12 месяцам общественных работ и к условному заключению на 12 месяцев за подбор паролей с целью проникновения в чужие компьютеры;
- наш соотечественник, 26-летний Василий Горшков из Челябинска, 10 октября 2001 г. был приговорен по 20 пунктам обвинения за многочисленные компьютерные преступления, совершенные против сети Speakeasy в Сиэтле (Вашингтон), банка Nara в Лос-Анджелесе (Калифорния), Центрального национального банка Вако в Техасе и онлайновой платежной компании PayPal из Пало-Альто (Калифорния);
- Олег Зезев, гражданин Казахстана, 1 июля 2003 г. был приговорен Манхэттенским федеральным судом к 51 месяцу заключения за компьютерное вымогательство.

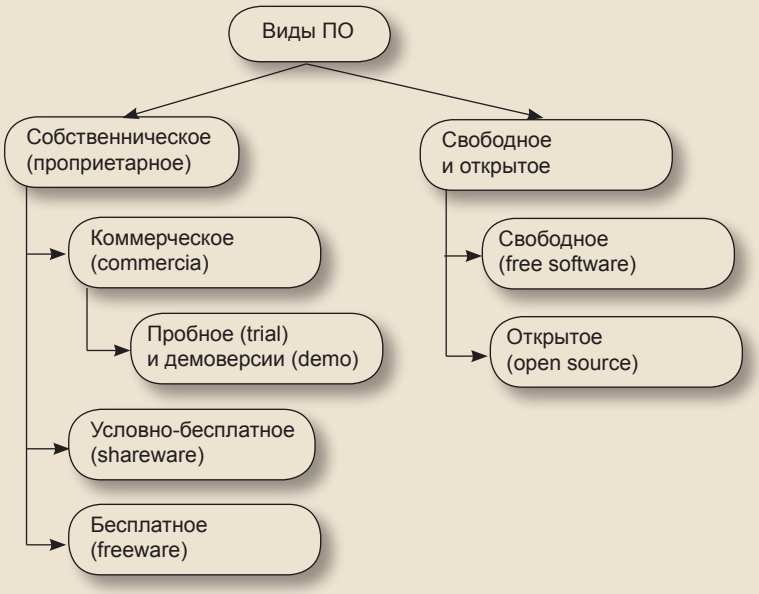
Основы лицензионной политики в сфере распространения программ и данных

Итак, согласно закону, распространение, доступ и использование информации строго регламентируется, поэтому легальное (т.е. законное) распространение и использование любых программ, а также данных (например, информации в составе баз данных) должно осуществляться согласно определенным правилам. Набор таких правил, устанавливаемый (в рамках, допустимых законом) собственником программы — ее создателем или продавцом, называют *лицензией (пользовательской лицензией)*. Соответственно, *лицензионная политика* рассматривается как комплексный механизм, определяющий всю совокупность условий предоставления той или иной программы пользователям, включая систему ценовых скидок и оговоренные в лицензии ограничения и специальные условия ее использования.

Виды ПО

По принципам распространения и использования, согласно соответствующим лицензиям, программное обеспечение делится на следующие виды:

Коммерческое программное обеспечение (commercial software) — это программы и программные пакеты, распространяемые только путем продажи. Возможен также вариант, когда сама программа предоставляется бесплатно, а продаже подлежит, например, подписка на осуществляемые при помощи этой программы онлайн или иные услуги. Возможен и случай, когда плата взимается как за саму программу (однократно при ее покупке), так и в виде абонентской платы за некоторый период времени. Примером является "Антивирус Касперского": купив саму программу, далее нужно оплачивать только подписку на услугу по регулярному обновлению антивирусных баз, что необходимо для полноценной работы антивируса.



Разновидностями коммерческого ПО являются также *пробные версии программ.* Как правило, это разновидности соответствующей коммерческой программы, распространяемые бесплатно в рекламных целях, но ограниченные по функциям по сравнению с полноценной версией либо полнофункциональные, но работающие только в течение ограниченного периода времени (оговоренное количество дней либо количество запусков) после установки, после чего пробная программа либо перестает работать вовсе, либо переходит в режим с ограниченным функционалом. Примером такого вида ПО (или, точнее, способа распространения и рекламы соответствующих коммерческих программ) является все тот же "Антивирус Касперского": можно скачать с сайта саму программу, получить к ней "пробный" ключ доступа к обновлениям вирусных баз на один месяц, а по завершении этого периода (если не оплатить подписку на дальнейшее обновление вирусных баз) программа перейдет в ограниченный режим: ее функции по проверке и лечению ранее известных программе вирусов сохранятся, но функции обновления вирусных баз будут отключены.

Условно-бесплатное ПО (shareware) — это программы (как правило, созданные индивидуальными программистами-любителями), которые можно получить и использовать бесплатно, но с определенными ограничениями, указанными в лицензии (например, разрешается бесплатное использование только в некоммерческих, личных целях или в образовательных учреждениях), либо имеется обращение к пользователю с просьбой о небольшом добровольном денежном пожертвовании в качестве вознаграждения автору в качестве стимула к дальнейшему совершенствованию программы, если она понравилась данному пользователю. Примером этого класса ПО является антивирус Avira AntiVir Personal-Edition Classic, который можно бесплатно использовать на личном компьютере или в школе.

Бесплатное ПО (freeware) — это программы, которые, согласно лицензии, вообще не требуют никаких денежных выплат автору (ограничиваются лишь возможные действия пользователя по отношению к этой программе, — например, допускается только ее личное использование "как есть", но не разрешено что-либо менять в исполняемом коде, переписывать программу другим лицам и т.д.). Как правило, это программы, созданные отдельными энтузиастами "для себя и для друзей" либо написанные начинающими программистами, желающими "создать себе имя". В некоторых случаях бесплатное ПО также создают крупные фирмы, если этого требует какая-либо важная цель: например, "Лаборатория Касперского" предлагает посетителям своего сайта целый ряд бесплатных антивирусных утилит, нацеленных на борьбу с отдельными вредоносными программами, — ради того, чтобы как можно быстрее "погасить" ту или иную вирусную эпидемию.

Заметим, что все указанные программы относятся к категории *проприетарного*, или *"собственнического", программного обеспечения,* поскольку во всех описанных разновидностях предполагается, что программа (даже бесплатная, freeware) кому-то принадлежит: пользователь покупает или получает на иных условиях право использовать такую программу, но не имеет права что-либо менять в ней или копировать программу другим лицам.

Противоположностью проприетарного является *свободное,* или *открытое, программное обеспечение.* Как правило, оно тоже является бесплатным (а оплата может браться только за сборку комплекта дистрибутивов с учетом их тестирования на возможные взаимные конфликты, выполнения русификации программ и пр.). Сами категории свободного и открытого ПО в основном аналогичны и различаются небольшими нюансами.

Так, *свободное ПО (free software)* предполагает, что пользователю (согласно особой, свободной

лицензии) предоставляются права ("свободы") на неограниченные установку, запуск, свободное использование, изучение, распространение и изменение (совершенствование) таких программ.

Сторонники же *открытого ПО (open source software)* делают основной акцент на предоставлении вместе с исполняемой программой ее исходного программного кода (листинга), открытого для просмотра, изучения и внесения изменений. Поэтому открытой лицензией не запрещается ни дорабатывать открытую программу (вносить изменения и исправления в ее исходный код с последующей компиляцией в новую версию исполняемой программы), ни использовать фрагменты такого исходного кода для создания собственных программ (хотя при этом может ставиться условие, что программы, содержащие такой заимствованный исходный код, могут далее распространяться тоже только как открытые). А вот свободность и даже бесплатность открытых программ необязательны, хотя большинство открытых программ также одновременно являются и бесплатными. Обратным примером является, скажем, утилита UnRAR — распаковщик RAR-архивов: ее исходный код имеется в открытом доступе для распространения и изменения, но, согласно лицензии, его нельзя использовать для создания RAR-совместимых архиваторов, т.е. эта утилита не является полностью свободной.

Коммерческая лицензия

Типовые условия, указанные в лицензии (лицензионном соглашении) на коммерческую программу, обычно включают в себя следующее:

- поскольку исключительные имущественные права на все компоненты поставляемого программного пакета принадлежат правообладателю (фирме-изготовителю программного обеспечения), покупка антивирусной программы не передает пользователю никаких прав собственности на нее, а передает только перечисленные ниже в лицензии права на использование этой программы при соблюдении ряда оговоренных условий, а также права на получение обновлений и указанных в лицензии сервисных услуг;
- для юридической фиксации соглашения между правообладателем и пользователем обычно достаточно, чтобы пользователь выразил свое согласие, нажав на кнопку **Согласен** (или аналогичную) и продолжив установку программы, а не прервав этот процесс, — никаких подписей под тем или иным документом (как это обычно делается при заключении договоров) от обеих сторон соглашения не требуется;
- четко оговорено, что пользователь может делать с программой (или с ее помощью) и какие еще услуги (и при каких условиях) он может получать от правообладателя;
- определяется возможность для пользователя изготавливать для себя "страховочную" (архивную) копию программы, а также полностью передавать свои права, оговоренные лицензией, другому пользователю;
- указаны действия над программой, которые запрещено производить пользователю;
- присутствует ряд указаний на ситуации, в случае которых правообладатель (как изготовитель программы) снимает с себя ответственность за неправильную работу либо полную неработоспособность самой программы, а также за возможный прямой или косвенный ущерб от ее использования.

В лицензии также могут оговариваться условия использования программы в каких-либо конкретных ситуациях (например, только некоммерческое, только личное, только в образовательных учреждениях), количество компьютеров, на которые может быть установлена данная программа с одного приобретенного дистрибутива, и т.д.

Кроме лицензии, в некоторых случаях для подтверждения лицензионных прав пользователя предусматривается *сертификат подлинности (лицензионный сертификат),* на котором указаны наименование программного продукта и *"ключ продукта" (Product Key)* в виде некоторого *серийного номера.*

Открытая лицензия GNU GPL

В отличие от коммерческой *открытая лицензия (GNU General Public License, GNU GPL)* удостоверяет, что данное ПО является свободным для всех его пользователей и гарантирует им свободу использовать и изменять это ПО. Она была создана в рамках проекта GNU в 1988 г., а позже были созданы вторая (в 1991 г.) и третья ее версии (в 2007 г.). Цель лицензии GNU GPL — предоставить пользователю следующие права ("свободы"):

Ричард Столлман — основатель движения свободного ПО и проекта GNU. Фото на выступлении в МГУ 3 марта 2008 г. (автор: Леонид Дмитриев, leozub)

• свободу запуска программы с любой целью;

• свободу изучения исходного кода программы, принципов ее работы и ее улучшения (модификации);

• свободу распространения копий программы;

• свободу улучшений и доработок программы и выпуска новых ее версий в публичный доступ.

Кроме того, лицензия GNU GPL гарантирует, что пользователи всех программ, созданных на основе упомянутого исходного кода, тоже получат все вышеперечисленные права. Например, запрещается создавать на основе свободной программы с лицензией GPL какое-либо другое ПО, в комплекте которого не будет предоставлен открытый исходный листинг.

Более подробно с текстом открытой лицензии GNU GPL интересующиеся могут ознакомиться самостоятельно, например, на сайтах:

• *http://www.gnu.org/copyleft/gpl.html* — исходный английский текст;

• *http://www.citforum.ru/operating_systems/articles/gpl_rus.shtml* — неофициальный русский перевод.

Угрозы при использовании нелицензионного ПО

Нелицензионным (пиратским) называют программное обеспечение, полученное и/или используемое незаконным способом, т.е. в нарушение правил пользовательской лицензии (как правило, от третьих лиц или через третьих лиц[1], не имеющих прав на распространение такой программы).

Что же грозит вам и вашему компьютеру при использовании нелицензионного программного обеспечения?

Во-первых, распространение и использование нелицензионного ПО является нарушением *Закона об охране авторских прав,* за что виновный несет гражданскую, административную или уголовную ответственность. Подробно этот аспект мы рассматривать не будем; желающие могут обратиться к тексту Закона РФ “Об авторском праве и смежных правах”.

Во-вторых, практически любая современная программа, распространяемая на коммерческой основе, содержит встроенные средства контроля нелицензионного использования (защита от несанкционированного копирования): это может быть индивидуальный серийный номер либо специальная *процедура “активации”*, специальный ключевой файл, аппаратный ключ, вставляемый в один из портов компьютера, и т.д. Конечно, абсолютной защиты, которую невозможно было бы “взломать”, практически не существует, — однако, “взламывая” подобную защиту, хакер может по незнанию повредить остальной программный код. В этом случае работа “взломанной” программы становится нестабильной, она может время от времени аварийно прекращать работу, в ней могут не работать некоторые функции[2] либо даже такая программа может портить информацию, имеющуюся на диске. Более того, хакеры нередко используют “взломанные” программы в качестве своеобразной “приманки”, встраивая в них “троянские программы” для своих преступных целей. Разумеется, ни в том, ни в другом случае изготовитель программы не несет никакой ответственности за нанесенный ею ущерб.

Наконец, в-третьих, сайты (а нередко — и диски), посредством которых распространяются “взломанные” программы либо различные средства для их “взлома” пользователями (серийные номера, программы — “генераторы серийных номеров”, утилиты[3] для отключения средств контроля нелицензионного использования и т.п.), нередко бывают заражены вредоносными программами различных видов: опять-таки возможность бесплатно заполучить дорогостоящую программу используется как “приманка”.

Таким образом, пользуясь нелицензионным программным обеспечением, вы подвергаете опасности данные, которые хранятся на вашем компьютере. А в результате вам в лучшем случае потребуется заново переустанавливать операционную систему и все прикладное ПО, а в худшем — это может привести к невосстановимой потере результатов вашей работы за длительный период времени либо даже к тому, что ваша конфиденциальная информация (а может быть, даже и деньги) окажется в руках злоумышленников. Подобный ущерб может оказаться существенно большим, чем стоимость полноценной коммерческой версии программы.

[1] В юридических документах “третьим лицом” называют кого-либо, кто не является одним из участников заключенного договора, документа и т.д. (т.е. кто-то посторонний). Если нелицензионное ПО было переписано из Интернета, то это означает, что существует кто-то, кто в нарушение Закона об охране авторских прав разместил эту программу в Интернете (на сайте, в файловой коллекции, сети файлообмена и пр.). Поэтому и в этом случае речь идет о незаконном получении программы от третьих лиц.

[2] Например, при нелицензионном использовании антивирусной программы становится невозможным своевременное автоматическое обновление ее вирусных баз, из-за чего программа теряет существенную часть своей эффективности.

[3] *Утилита* — компьютерная программа, расширяющая стандартные возможности оборудования и операционных систем, выполняющая узкий круг специфических задач.

Система автоматизации тестирования

MimioVote™

обеспечивает проведение тестирования учащихся, делает проверку работ и выставление оценок легким и быстрым.

Любые типы тестов и ведение статистики.

Получайте результаты мгновенно и избавьтесь от проверки тестов вручную!

Система тестирования **MimioVote™** обеспечивает подготовку и проведение тестирования учащихся, автоматически производит проверку работ и упрощает выставление оценок. Журнал ответов и оценок по отдельным ученикам и в целом по классам позволяет анализировать статистику и формировать отчеты.

MimioVote™ позволяет подходить к подготовке тестовых заданий гибко и творчески. Система поставляется с комплектом готовых шаблонов для всех типов тестирования. Вы можете создавать тесты с помощью **MimioStudio™**, использовать Microsoft PowerPoint или импортировать уже имеющиеся у Вас тесты в различных форматах. Использование готовых тестовых наборов, удобные инструменты создания и импорта тестов облегчают подготовку к уроку и существенно экономят ваше время.

Пульты для ответов спроектированы таким образом, что работать с ними ученикам не только удобно и понятно, но и увлекательно. Они автоматически заряжаются и нумеруются при хранении. Доступность кнопок ответа на пульте определяется типами вопросов и прекрасно индицируется, что упрощает работу учеников и уменьшает вероятность случайных ошибок. В комплект входит 24 или 32 беспроводных перезаряжаемых пульта. Для хранения и транспортировки предусмотрен удобный прочный футляр.

MimioVote™ является частью комплексного решения **MimioClassroom™**, интегрированной интерактивной системы, включающей также интерактивную приставку, документ-камеру и многое другое.

Узнайте больше, посетите наш сайт или позвоните:

http://www.mimioclass.ru

8 (800) 5555-33-0

Звонок по России бесплатный

ООО «Рене» — генеральный дистрибьютор Mimio

DYMO | Mimio

О чем это?
Авторы ЕГЭ и их соавторы, готовящие тренировочные варианты, не перестают радовать нас новыми заданиями ☺. За оставшееся до экзамена время надо постараться попробовать все эти радости на вкус.

И это все?!
На диске имеется презентация, которая поможет при разборе материалов статьи в классе.

Сжимать иль не сжимать — вот в чем вопрос

О.Б. Богомолова, д. п. н., учитель информатики и математики, Восточный округ г. Москвы

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

Задачи о передаче некоторого массива данных последовательно по двум каналам с разной скоростью (см. статью «“Задача о передаче”: решение задачи ЕГЭ при помощи сетевых диаграмм Ганта» в № 7 за 2011 г.) в ЕГЭ по информатике и в ЕГЭ-подобных контрольно-тренировочных работах стали уже, можно сказать, традиционными. Однако в диагностической работе № 1 по информатике (декабрь 2011 г.), подготовленной МИОО, учащимся предлагалась задача хотя и тоже про передачу данных, но с другим условием, которое затрагивает теперь тему “Архивирование информации”.

B10, вариант 1

Документ объемом 10 Мб можно передать с одного компьютера на другой двумя способами:

А) сжать архиватором, передать архив по каналу связи, распаковать;

Б) передать по каналу связи без использования архиватора.

Какой способ быстрее и насколько, если:

— средняя скорость передачи данных по каналу связи составляет 2^{18} бит в секунду,

— объем сжатого архиватором документа равен 30% от исходного,

— время, требуемое на сжатие документа, — 5 секунд, на распаковку — 1 секунда?

В ответе напишите букву А, если способ А быстрее, или Б, если быстрее способ Б. Сразу после буквы напишите количество секунд, насколько один способ быстрее другого. Например, если способ Б быстрее способа А на 23 секунды, в ответе нужно написать “Б23”. Слова “секунд”, “сек.”, “с.” к ответу добавлять не нужно.

Решение

В чем-то данная задача даже проще, чем традиционные задачи о передаче данных, когда учащемуся приходилось строить сетевую диаграмму, чтобы увидеть наглядно ход всего процесса передачи массива информации. Только расчеты нужно выполнить дважды — для обоих предложенных вариантов (А и Б).

Вариант А:

Предполагается, что исходный массив информации сначала архивируется (5 секунд). Затем он в уже упакованном виде (30% от исходного объема) передается по каналу связи с заданной средней скоростью, а после этого распаковывается (1 секунда).

Запишем соответствующую цепочку вычислений:

$$5 + (0{,}3 \cdot \underbrace{10}_{\text{мегабайты}} \cdot \underbrace{2^{20}}_{\text{байты}} \cdot \underbrace{2^{3}}_{\text{биты}} / 2^{18}) + 1$$

Далее можно после некоторого количества арифметических операций вычислить длительность всего процесса в секундах, —

но мы пока подождем это делать (почему — будет ясно чуть позже).

Вариант Б:

Аналогичным образом можно записать вычисления времени передачи неупакованного массива информации по каналу связи:

$$10 \cdot 2^{20} \cdot 2^3 / 2^{18}$$

мегабайты
байты
биты

Выполнив вычисления и сравнив результат с полученным для варианта А, можно найти ответ к задаче.

Однако можно решить эту задачу более экономно — выполнив вычисления не дважды, а только один раз.

Сравним предлагаемые варианты (А и Б). Очевидно, что 30% от исходного объема массива информации приходится передавать по каналу связи в обоих случаях. А в чем различие?

В варианте А, кроме этих 30% объема файла, по каналу связи больше ничего передавать не требуется, но зато имеются "накладные расходы" времени на упаковку/распаковку в количестве 6 секунд.

В варианте Б этих дополнительных расходов времени нет, но зато приходится передавать по каналу связи оставшиеся 70% объема файла.

Получается, что для решения задачи достаточно сравнить между собой затраты времени именно на эти различающиеся в вариантах А и Б операции:

6 секунд на упаковку/распаковку VS передача 70% объема файла

Другими словами, нам достаточно вычислить время передачи по каналу связи с заданной скоростью 70% исходного объема информации и вычесть из этого времени 6 секунд:

- если результат отрицателен, то быстрее способ Б;
- если результат положителен, то быстрее способ А;
- если результат равен нулю, то оба способа равнозначны по скорости;
- абсолютное значение (модуль) разности — это и есть искомое время, на которое один способ быстрее другого.

В нашем случае:

$0{,}7 \cdot 10 \cdot 2^{20} \cdot 2^3 / 2^{18} - 6 = 0{,}7 \cdot 5 \cdot 2^{24} / 2^{18} - 6 =$

$= 3{,}5 \cdot 2^6 - 6 = 3{,}5 \cdot 64 - 6 = 218$ с.

Ответ: **А218** (способ А быстрее способа Б на 218 секунд).

Следует заметить, что этот ответ достаточно близок к реальности, если речь идет о быстром компьютере (процессор которого обладает высоким быстродействием) и достаточно медленном канале связи; например, данный эффект особенно заметен для файлов большого объема при их передаче через dial-up-доступ в Интернет (там, где он еще остался и не вытеснен Ethernet-доступом ☺) или при копировании информации на внешний накопитель через USB 1.0 (например, если на современном компьютере "собьется" драйвер, обеспечивающий более быстрый протокол передачи данных USB 2.0). В подобных случаях действительно выгоднее упаковывать файл, а после передачи (или переноса на внешнем накопителе на другой компьютер) распаковывать его.

Кстати, аналогичный "побочный" эффект нередко наблюдали и пользователи 286-х, а особенно 386-х компьютеров, на которых был установлен драйвер упаковки/распаковки файлов "на лету" типа Stacker. Основное его назначение — экономия места на жестком диске (емкости которых тогда были довольно малы): файл при его записи из ОЗУ на жесткий диск автоматически сжимался, а при чтении или копировании с жесткого диска на внешний носитель — автоматически распаковывался. А поскольку скорость чтения/записи информации с/на эти первые модели жестких дисков была не очень высока, пользователи нередко замечали, что с драйвером Stacker работать с файлами большого объема получалось несколько быстрее. И вы, дорогой читатель, теперь знаете, отчего…

B10, вариант 2

Документ объемом 10 Мб можно передать с одного компьютера на другой двумя способами:

А) сжать архиватором, передать архив по каналу связи, распаковать;

Б) передать по каналу связи без использования архиватора.

Какой способ быстрее и насколько, если:

— средняя скорость передачи данных по каналу связи составляет 2^{18} бит в секунду,

— объем сжатого архиватором документа равен 30% от исходного,

— время, требуемое на сжатие документа, — 7 секунд, на распаковку — 1 секунда?

В ответе напишите букву А, если способ А быстрее, или Б, если быстрее способ Б. Сразу после буквы напишите количество секунд, насколько один способ быстрее другого. Например, если способ Б быстрее способа А на 23 секунды, в ответе нужно написать "Б23". Слова "секунд", "сек.", "с." к ответу добавлять не нужно.

Решение задачи B10, вариант 2

В случае А на упаковку/распаковку файла затрачивается 8 секунд, но зато требуется передать по каналу связи только 30% от исходного объема файла.

В случае Б затрат времени на упаковку и распаковку файла нет, но требуется передать, кроме тех же самых 30%, остальные 70% объема файла.

Получаем, что в сравнении способов А и Б:

8 секунд на упаковку/распаковку VS передача 70% объема файла

Выполняем расчеты:

$$0{,}7 \cdot 10 \cdot 2^{20} \cdot 2^3 / 2^{18} - 8 =$$

мегабайты
байты
биты

$= 0{,}7 \cdot 5 \cdot 2^{24} / 2^{18} - 8 = 3{,}5 \cdot 2^6 - 8 = 3{,}5 \cdot 64 - 8 = 216$ с.

Ответ: **А216** (способ А быстрее способа Б на 216 секунд).

Кстати, эти задачи рождают тему для интересного исследования: предлагаем читателям (составив соответствующую программу, используя Excel и т.д.) самостоятельно выяснить — начиная с какого значения объема файла (для заданных скоростей передачи данных, упаковки и распаковки) архивирование файла перед его передачей становится выгодным, либо наоборот, какая наименьшая величина скорости передачи данных делает выгодным архивирование передаваемого файла при заданных его объеме и скорости упаковки/распаковки.

О чем это?
Это статья о языке программирования Python. Язык этот далеко не новый, но именно в последнее время он все шире начинает использоваться в российских школах. Статья может быть использована для первого знакомства с языком.

И это все?!
На диске имеются свободно распространяемый дистрибутив Python'а (его можно также, разумеется, скачать из Интернета) и исходные файлы для всех примеров.

Python, проглатывающий слона

И.А. Сукин

От редакции

Очень трудно определить жанр статьи, предлагаемой вашему вниманию. Редакция поставила перед автором задачу написать предельно краткое введение в язык Python, проведя в тех случаях, где это возможно, параллели с Паскалем. По факту получилось нечто среднее между кратким введением, обзором и предельно кратким справочником.

В статье совсем нет методической составляющей, но ее и нельзя было требовать от автора, который является специалистом по языкам программирования. Зато степень концентрации информации именно по языку на страницу текста очень высока — "только добавь воды" — из статьи вся "вода" отжата.

Мы надеемся, что для желающих статья позволит понять, что такое Python, на что он похож, на что не похож, на что похож код на этом языке, как его читать, как его писать, как его запускать. А методические материалы, уверены, последуют — Python все шире используется в российских школах, и все большее число наших коллег имеет свой личный опыт преподавания алгоритмизации и программирования с опорой на Python. Опыт, кстати, в отличие от мнения, высказанного автором, по большей части положительный.

Введение

Язык программирования Python, о котором мы будем вести речь, является одним из самых знаковых языков современной эпохи. Этот язык можно порекомендовать как новичку (но не "чайнику") в сфере компьютерной инженерии, так и матерому профессионалу. Области применения языка Python поистине безграничны, он используется в тех или иных целях подавляющим большинством современных крупных корпораций, управляет бизнес-потоками и полетами космических кораблей, обеспечивает логику в компьютерных играх или даже является основным языком написания пользовательских приложений.

Чтобы начать знакомство с этим невероятно многогранным инструментом, как и с другим языком программирования, мы ненадолго окунемся в историю и посмотрим, какие парадигмы унаследовал Python и ка-

кие проблемы он должен был решить по замыслу своего создателя.

Немного истории

В начале девяностых годов область скриптовых[1] языков была развита значительно слабее, чем сегодня, а об унифицированности в ней можно было только мечтать. Многие разработчики программных систем предпочитали решения ad-hoc, проектируя для каждого конкретного случая свой небольшой язык. Основной причиной этого были довольно небольшие вычислительные мощности, делавшие невозможной или очень трудной эффективную реализацию интерпретируемого языка общего назначения. В операционных системах UNIX место основного скриптового языка прочно занимал молодой в ту эпоху Perl, не отличавшийся особой скоростью выполнения, но успешно справлявшийся со своими задачами.

В то же время сотруднику голландского института CWI Гвидо ван Россуму понадобился скриптовый язык для его собственной операционной системы Amoeba. Основными целями разработки нового инструмента Гвидо ставил легкость освоения (уже тогда время программистов ценилось больше времени машин), хорошую скорость исполнения программ, выразительную мощность и средства объектно ориентированного программирования. Потом язык приобрел некоторые другие отличительные особенности, считающиеся сейчас основополагающими в нем, но об этом позднее.

Чем же ван Россуму не понравился Perl, успевший стать неким стандартом де-факто среди профессиональных программистов? Perl был достаточно производительным в своей области и позволял выражать довольно сложные идиомы простым программным кодом, однако его дизайн был (и является) слишком своеобразным (в шутку можно заметить, что автор языка Perl выигрывал соревнования по написанию самого запутанного кода на C), чтобы соответствовать требованию “легкости освоения”, к тому же в языке нет и не было встроенной объектной системы.

За основу нового языка был взят язык ABC, в разработке которого Гвидо ван Россум принимал участие несколькими годами ранее. Поскольку читателям название “ABC” скорее всего ничего не говорит, упомяну, что сам ABC является наследником таких языков, как SETL и Algol-68. Разумеется, некоторая оглядка, в частности в сфере применения в скриптах командных оболочек, была сделана на Perl. Остальными языками, повлиявшими на Python, были Icon, Lisp, Smalltalk, Fortran, Modula-3, Miranda. В результате получилась динамическая объектно ориентированная среда, обладающая чертами как императивного, так и функционального программирования и позволяющая писать довольно сложные программы общего назначения.

Рассмотрим ключевые особенности языка Python, позаимствованные им из разработок прошлых лет:

- **Пакеты (модули)** — важная инкапсуляционная (обеспечивающая сокрытие данных) особенность, пришедшая из Modula-2/3. Любой файл с программным кодом в Python является модулем с собственным пространством имен (namespace).
- **Объектная модель**, позаимствованная из Smalltalk. Об объектно ориентированном программировании в Python я расскажу отдельно.
- **Функционалы, лямбда-выражения** и другие черты функционального программирования взяты в основном из языка Lisp. Функциональному программированию на Python также будет посвящен отдельный подраздел.
- **Срезы массивов, компле́ксная арифметика** взяты из Фортрана. Взятие срезов позволяет очень гибко организовывать работу с массивами.
- **Списковые выражения**, которые должны быть знакомы тем читателям, которые имели дело с языком Haskell, взяты из прародителя Haskell — языка Miranda. Списочные выражения позволяют гибко организовать работу с множествами и коллекциями.
- **Генераторы**, известные любителям функционального программирования в виде инструментов ленивых вычислений. Позволяют работать с “бесконечными” наборами данных и “ленивыми” коллекциями данных.

Не беспокойтесь, если некоторые слова из предыдущего абзаца показались вам незнакомыми или даже страшными, я обязательно вернусь к подробному описанию этих возможностей позже и покажу, насколько просто и интуитивно понятно они реализованы в языке Python.

Основная мысль, которую я бы хотел вынести в этом подразделе: Python был создан в переходную для информатики эпоху, на стыке многих существовавших в то время и успевших положительно зарекомендовать себя технологий, имея целью создание простого, но мощного языка. Двадцать лет развития языка Python показали жизнеспособность и конкурентоспособность этого начинания. В качестве признака успеха для “продвинутых” читателей могу привести тот факт, что Python вытеснил Scheme в качестве языка для обучения основам информатики и алгоритмизации в Массачусетсском технологическом институте.

Почему Python?

Важным вопросом в использовании любого языка программирования X является вопрос: “А почему X?”. Вы скорее всего тоже задавали себе этот вопрос еще до знакомства с этой статьей. Наиболее полный ответ на него вы можете найти у Марка Лутца в [1], я же отвечу лишь обзорно, но достаточно убедительно.

[1] То есть, в узком смысле, языков, предназначенных для написания сценариев для различных программных оболочек.

Предпосылки к использованию:

• *Простота и высокая читаемость кода.* Обеспечивается поддержкой и некоторым слежением за стилем со стороны интерпретатора, а также простотой синтаксиса. Даже в чужих программах на Python легко может разобраться новичок.

• *Выразительность и мощность встроенных средств.* Python поддерживает множество парадигм программирования. Вы можете писать одновременно и функциональные, и объектно ориентированные программы, используя самые последние достижения обоих областей. С помощью хороших возможностей метапрограммирования (однако не таких хороших, как в Lisp'е) любая другая парадигма, как то логическое программирование или аспектно ориентированное, может быть легко реализована в Python.

• *Богатая стандартная библиотека.* Стандартная библиотека в Python содержит переносимые модули для самых разных задач: веб-программирования, работы с данными в разнообразных форматах, математических вычислений, обработки текстов и строк, работы с графическим интерфейсом пользователя. Существуют реализации языка Python для платформ .NET и JVM (Java), имеющие доступ ко всему многообразию стандартных библиотек этих платформ.

• *Огромное количество сторонних библиотек.* Ни для одного другого языка, кроме C, не существует такого количества сторонних библиотек и фреймворков, как для языка Python. Сторонние модули существуют уже для поддержки практически любых инструментов и вычислений. Так, например, значительный объем накопившихся Фортран-библиотек уже переведен на Python с помощью автоматических трансляторов. Стоит также сказать, что подавляющее большинство сторонних модулей для Python распространяется свободно и бесплатно.

Причины, по которым вы можете выбрать Python в качестве своего основного языка программирования, опираются на указанные мной предпосылки:

• *Высокое качество программного обеспечения* поддерживается простотой программного кода и легкостью его сопровождения, а также довольно единообразными правилами его написания. Большое количество программистов, уже пишущих на языке Python, обеспечивает нахождение множества ошибок в программах (согласно "закону Линуса"[2]).

• *Высокая скорость разработки.* Те годы, когда время машины стоило гораздо дороже времени программиста, давно канули в Лету и уступили место эпохе языков, ориентированных на разработчика. Python в этом плане не исключение и обеспечивает хорошую производительность разработки за счет мощной выразительности кода и большого количества уже написанных программ и модулей. Не последнюю роль в высокой скорости программирования играет интерактивный интерпретатор с богатыми возможностями отладки и, собственно, сама концепция интерпретируемого языка, избавляющая разработчика от дополнительных циклов компиляции и компоновки (для этих же целей в CERN создали интерпретаторы C и C++).

• *Переносимость и кроссплатформенность.* Интерпретаторы Python существуют на огромном числе программных и аппаратных платформ, а стандартная библиотека поддерживает создание переносимых графических интерфейсов, интерфейсов к файловой системе и веб-приложениям. Если и этого вам не хватит, то существует, к примеру, интерпретатор Jython, работающий под управлением Java-машины и запускающийся везде, где и последняя. В этом пункте стоит добавить, что, несмотря на хорошую кроссплатформенность, в стандартной и сторонних библиотеках есть множество модулей для поддержки функций каждой конкретной платформы.

• *Высокая степень интеграции.* Из языка Python можно вызывать функции, написанные на других языках, таких, как C, C++, Lisp, Prolog, Java, OCaml, что обеспечивает очень тесную интеграцию с уже существующими системами и позволяет не переписывать свой код лишний раз.

• *Удовольствие от программирования.* В современной компьютерной инженерии, когда объемы программных систем достигают десятков, сотен тысяч и даже миллионов строк кода, никто не хочет разрабатывать их с помощью устаревших инструментов, в которых значительная часть усилий уходит на не важные для задачи действия. Благодаря простоте синтаксиса и динамической природе Python позволяет не отвлекаться на бесполезную работу.

Кроме всего этого, немаловажным фактором является то, что Python при всей своей выразительной мощи прост для изучения тем, кто знаком с базовыми языками школьной и университетской программ: Basic, Pascal, C, C++, — в отличие от тех же языков чисто функционального программирования типа Haskell и Ocaml. Описанию языка Python как наследника языка Pascal будет посвящен второй подраздел.

Почему не Python?

Разумеется, как и любой другой инструмент, Python неидеален и имеет ряд недостатков, которые могут сыграть решающую роль в выборе вами Python как языка для изучения, разработки, преподавания. Я опишу два важнейших:

• *Python — медленный язык.* Как и все остальные интерпретируемые языки. Скорость выполнения программ на нем может быть до 200 раз меньше скорости выполнения программ, написанных на C

[2] "При достаточном количестве программистов любая проблема будет быстро обнаружена", эмпирическое правило, названное в честь Линуса Торвальдса, автора ядра операционной системы Linux.

(при текстовой интерпретации и использовании неэффективных конструкций), но в целом программы на Python примерно в 2–4 раза медленнее, чем аналогичные программы на C (особенно при использовании компиляции в байт-код). Поэтому следует помнить, что все критичные к производительности участки придется переписывать на C или Фортране. Большинство математических библиотек для Python уже написано на C. Существуют также попытки создания оптимизирующих компиляторов для Python, которые, однако, пока работают только с некоторым подмножеством языка. В общем, если ваше приложение требует исключительной скорости или будет работать в условиях жестко ограниченных ресурсов, Python — не ваш выбор.

• *Программирование на Python — не искусство*, а технологический процесс [1]. В "философии" языка явно оговаривается тот факт, что должен быть один и только один способ решить задачу (в идеале этот тезис противопоставляется тезису языка Perl "всегда должно быть более одного способа решить задачу"). Если вы представляете программирование как своего рода искусство или выражение собственных мыслей в программном коде, я также не могу порекомендовать вам Python.

Для обучения программированию эти недостатки не являются слишком критичными. Как мы дальше увидим, в Python есть место и скорости, и творчеству.

Беглый обзор и как начать работу

Подробное описание внутренностей интерпретатора и нюансов работы с ним вы можете найти в соответствующей литературе [1]–[4], я же расскажу здесь лишь о том, что необходимо для прочтения и понимания следующих подразделов.

Основной интерпретатор Python (или, как еще говорят, референсный) называется CPython и доступен на официальном сайте *http://www.python.org*, интерпретатор для Java-машины — Jython доступен с сайта *http://www.jython.org*. Для желающих посмотреть на внутренности Python, не вникая в код на C, существует интерпретатор Python, написанный на самом Python, — PyPy, *http://www.pypy.org*. В этой статье я буду использовать интерпретатор CPython.

Важным моментом является то, что у языка есть два направления развития, два "стандарта" языка и два интерпретатора — Python 2 и Python 3, причем код на Python 3 не обязательно совместим с Python 2. В статье будет использован Python 2 по следующим причинам: другие интерпретаторы (Jython, PyPy, IronPython) поддерживают только Python 2; бóльшая часть сторонних библиотек пока поддерживает только Python 2; ветка Python 2 является наиболее стабильной. Тем не менее весь код в этой статье будет написан с максимальной совместимостью с Python 3.

Скачав дистрибутив и установив с официального сайта (последней версией Python 2 на момент написания статьи является 2.7.2) или установив пакет (для пользователей GNU/Linux), вы можете запустить интерпретатор из командной строки, набрав `python`, или с помощью соответствующего пункта меню. После запуска откроется консоль интерпретатора с небольшой информацией и приглашением вроде такого:

```
  Python 2.6.6 (r266:84292, Dec 27 2010,
00:02:40)
  [GCC 4.4.5] on linux2
  Type "help", "copyright", "credits"
  or "license" for more information.
  >>>
```

После чего вы можете интерактивно общаться с интерпретатором — вводить команды и смотреть на ответы Python.

Однако интерактивная строка интерпретатора не подходит для написания и исполнения серьезных программ и даже примеров, поэтому мы будем пользоваться средой разработки, входящей в стандартную поставку и называющейся IDLE (пользователям GNU/Linux, возможно, придется установить IDLE отдельно). В принципе программа может быть написана в любом текстовом редакторе, сохранена в файле с расширением *.py и затем выполнена командой `python имя_файла.py`.

Как только вы откроете IDLE, вашему взгляду предстанет уже знакомая командная строка интерпретатора — сюда будет направляться вывод нашей программы. Чтобы создать программу, необходимо нажать Ctrl–N или выбрать пункт меню **File** → **New Window**. Любая программа должна быть сохранена, после чего ее можно будет запустить клавишей F5 или пунктом меню **Run** → **Run Module** (в окне программы, а не интерпретатора!).

Теперь вы окончательно готовы перейти к работе с самим языком.

Python как "Паскаль на стероидах"

Фраза "на стероидах" (англ. *on drugs*) в информатике применяется по отношению к языку или программному инструменту, который может служить более мощной и удобной заменой другого инструмента и быть понятным пользователям последнего. В данном случае я попытаюсь сравнить Python с известным многим и широко используемым в школе языком Pascal и показать способность Python вытеснить Pascal из этой ниши (обучения программированию). В своем повествовании я буду в основном ориентироваться на уже ставшую классической книгу Валерия Васильевича Фаронова [6], поэтому параграфы будут иметь похожие заглавия.

Алфавит

По умолчанию в Python 2 используется семибитная ASCII-кодировка, то есть воспринимаются только латинские символы. Если вы хотите использовать символы вашего родного языка или Юнико-

да, вы должны явно специфицировать это в начале файла программы:

```
# Для КОИ-8
# -*- coding: koi8-r -*-
# Для Юникода
# -*- coding: utf-8 -*-
# Для пользователей Windows
# -*- coding: cp1251 -*-
```

Обратите внимание также на то, что символ "#" является еще и символом однострочного комментария.

Все идентификаторы и ключевые слова в Python набираются исключительно латиницей. В Python 3 этой проблемы нет, поскольку он полностью переведен на Юникод, можно даже называть переменные, функции и классы на своем родном языке, что, однако, считается дурным тоном.

Python чувствителен к регистру, `Test` и `test` — разные имена.

Ключевые слова

По сравнению с языком Pascal в Python не так уж и много ключевых слов, вот большинство (если не все) из них: `if else elif in is while for pass and not or def return yield lambda class raise except finally import from as global print` (в Python 3 `print` больше не является ключевым словом).

Константы

В качестве констант могут выступать: целые, вещественные, двоичные, восьмеричные, шестнадцатеричные и компле́ксные числа, логические константы, символы, строки, Юникод-строки, конструкторы списков, кортежей и словарей, лямбда-выражения, "пустой" объект, объект-эллипсис.

Целые числа могут быть со знаком и без, при этом они могут иметь любое значение, а не ограниченный диапазон значений, как в языке Pascal (!!!). Числа, большие максимального аппаратного значения Integer (Pascal), будут храниться (внешне незаметно для программиста) как особый "длинный" тип чисел, однако операции над ними будут теми же. Двоичные константы начинаются с `0b`, восьмеричные — с `0`, шестнадцатеричные — с `0x`. Все они могут быть как со знаком, так и без знака. Символ "`L`" или "`l`" сразу после целого числа определяет его как принадлежащее типу `long` (см. дальше).

Вещественные числа записываются точно так же, как в Pascal, и имеют те же самые границы. Компле́ксные числа записываются в виде *Re*+*Im*`j`, где Re и Im — вещественные числа, или целые, приведенные к вещественным. Например: `5-3e-3j` или `1+4j`. Мнимая единица записывается как `1j`.

Логическими константами являются `True` и `False`, однако стоит быть осторожным, потому что они не являются ключевыми словами и их можно переопределить значением любого другого типа (о типизации я расскажу дальше). Также в качестве ложного значения могут выступать `0`, `None`, пустой список и некоторые другие "пустые" выражения, а в качестве истинного — любые другие.

Символы в Python не отличаются от строк и, по сути, просто являются строками длины 1. Строки записываются в апострофах или кавычках и могут включать любые символы (если вы правильно задали кодировку). Для ввода непечатаемых символов используются так называемые "escape-последовательности" `'\хшестнадцатеричный_код'`, например, `'\x20'` — это символ пробела, `'\x07'` — символ "звонка". Код символа всегда должен состоять из двух цифр. Строки, ограниченные тремя кавычками, передаются "как есть", то есть со всеми переносами строки, и, кроме этого, могут являться многострочными комментариями (если они ничему не присваиваются, то и действия не производят, см. Выражения).

Юникод-строки в Python записываются точно так же, как и обычные, однако перед ними ставится символ "u", например, `u"test"`. Внутри Юникод-строк можно использовать Юникод-escape-последовательности для записи символов Юникода — `'\икод'`, например, `'\u0007'` — Юникод-символ "звонка". Обратите внимание, что теперь код состоит из четырех цифр, потому что Python 2 поддерживает Юникод вплоть до UCS-2.

Конструктор списка создает списки, аналоги массивов и выглядит как конструктор множеств в Pascal: `[1, 2, 4]`. Объекты в списке могут быть разнородными (знающие ООП понимают, что это появление полиморфизма вкупе с наследованием). Кортеж представляет собой "неизменяемый список" и применяется в тех случаях, когда функция должна вернуть несколько результатов.

Пустым объектом, аналогом NIL или NULL, в Python является `None`.

Все остальные типы констант выходят за рамки сравнения с Паскалем и будут описаны позже.

Выражения

Практически все в Python является выражением: любая переменная, любая константа. Операторы (условные, циклы) выражениями не являются. В интерактивном режиме интерпретатора можно ввести любое выражение и посмотреть его значение. В программе значение выражения теряется, если только оно не присваивается какой-нибудь переменной. Это позволяет использовать строки в качестве комментариев.

```
"""Это комментарий! """
```

Операции

Операции в Python аналогичны операциям в большинстве других языков:

• Арифметические: `+`, `-`, `*`, `/` (это целочисленное деление! Чтобы деление было вещественным, один из его аргументов должен быть явно приведен к вещественному типу), `%` (нахождение остатка), `<<`, `>>` (сдвиги), `**` (степень).

```
print(10/3)
# выведет 3, а не 3.33333333333
```

В Python 3 целочисленное и обычное деление имеют различные обозначения. Первое обозначается так: `//`. То есть `10//3 = 3`, а `10/3 = 3.33333333333`

• Логические: `not`, `and`, `or` (работают не как побитовые операции, а как логические).

• Побитовые: `~` (не), `|` (или), `&` (и), `^` (исключающее или).

• Отношения: `==` (равенство), `!=` (неравенство), `<=`, `>=`, `<`, `>`, `in` (проверка на вхождение в множество или список/коллекцию), `is` (проверка на тождественность, говоря языком C или Паскаля, сравнение указателей на объекты).

Структура программы

Простейшая программа на Python — это пустая строка. Выполняются все строки, написанные в файле с программой и находящиеся вне определений классов и функций. Блоки кода в языке выделяются с помощью отступов. Здесь важно понять, что отступы в Python обязательны и весь код одного уровня вложенности должен быть написан с одинаковым отступом! Отступы делаются исключительно пробелами, а не табуляцией, и стандарт рекомендует отступы в четыре пробела. Блоки кода создаются только внутри операторов или определений, но не внутри других блоков кода. Если вы пока не поняли идею обязательных отступов, то увидите ее в действии позднее. Символом, разделяющим выражения в Python, является перевод строки.

Операторы

Простейший оператор — `pass`, пустой оператор, который не делает ничего. Он обычно используется в теле условного оператора (которое не может быть пустым).

Другой простой оператор — оператор присваивания (=), он присваивает некоторой переменной значение.

```
a = 5
```

Оператор присваивания в языке Python может иметь варианты с накоплением, как в C: `+=`, `-=`, `*=`, `/=` и т.д.

Условные операторы в языке Python представлены простым и составным `if`. После ключевого слова `if` идет условие, за ним двоеточие и блок кода (с отступом!), выполняющийся, если условие истинно. Дальше (со старым отступом) идет опциональное (не обязательное) ключевое слово `else`, двоеточие и блок кода, выполняющийся, если условие ложно.

```
a = 5
if a < 3:
    print("a<3") #обязательный отступ
else:
    print("a>=3")
#отступ должен быть таким же!
```

[3] В Python 2 `print` является как функцией, так и оператором (можно записать `print a, b, c`), однако я буду использовать ее в виде функции для совместимости с Python 3.

Составной оператор `if` служит заменой `case` и позволяет использовать более сложные условия.

```
a = 4
if 1 < a < 4:
# допустимы и такие условия
    print("1<a<4")
elif 4 <= a < 10:
    print("4<=a<10")
else:
# ветвь else здесь также опциональна
    print("a>=10 или a<=1")
```

Циклов в Python существует два типа: с предусловием (`while`) и с фиксированным числом итераций (`for`).

```
a = 4
while a > 0:
    print(a)
    # здесь также нужен отступ
    a = a - 1
```

Приведенный выше цикл исполняется четыре раза.

Циклы с фиксированным числом итераций имеют несколько непривычную для программиста на Pascal семантику: `for переменная in коллекция`. Коллекцией обычно является список.

```
for i in [4, 5, 1, 3]:
# выведет последовательно 4, 5, 1, 3
    print(i)
```

Для создания обычных циклов, в которых переменная изменяется от некоторого начального значения до некоторого конечного (не включительно) с некоторым шагом, существует встроенная функция `range(нач_значение, кон_значение, шаг)`, в которой начальное значение и шаг необязательны.

```
for i in range(5):
    print(i)
    # выведет 0, 1, 2, 3, 4
for i in range(1, 5):
    print(i)
    # выведет 1, 2, 3, 4
for i in range(1, 5, 2):
    print(i)
    # выведет 1, 3
```

Типизация и типы данных

В отличие от Pascal типизация в Python *динамическая*. Это значит, что типы переменных не должны быть объявлены заранее, они определяются во время выполнения. Кроме того, во время выполнения переменная может и поменять свой тип. Во время исполнения программы тип переменной можно узнать с помощью функции `type`.

```
print(type(5)) # выведет <type 'int'>[4]
```

Переменные начинают существовать сразу после того, как им присвоят значение, соответствен-

[4] В Python 3 отдельного понятия “тип” нет, есть только классы, и то же выражение выведет `<class 'int'>`. В Python 2 тип `type` можно считать обособленным вариантом класса.

но, неинициализированных переменных в Python нет. Все значения переменных, которые становятся недоступными из-за смены типа переменной, удаляются из памяти автоматически так называемым “сборщиком мусора”.

Прежде чем перейти к описанию непосредственно типов, хотелось бы заметить, что Python — язык, что называется, “управляемый” (англ. *managed*), и прямой доступ к памяти средствами языка невозможен, а внутреннее представление составных типов скрыто от программиста. При желании можно заглянуть внутрь всех конструкций и получить доступ к памяти, однако это считается небезопасным и практически не применяется. В дальнейшем вы увидите, что это ограничение не является на самом деле существенным.

Для целых чисел[5] предназначены типы `int` (занимающий машинное слово) и `long` (целое любой длины, вплоть до всего объема оперативной памяти). Для вещественных и комплексных — `float` и `complex` соответственно. Логические значения имеют тип `bool`, строки — `str`, Юникод-строки — `unicode`. Типов перечислений (они заменяются словарями или списками), типов-диапазонов и типов-объединений в Python нет, а типы-записи представляются классами и объектами.

Стандартные функции для чисел в Python в основном унаследованы из C (math.h). Многие математические функции находятся в модуле `math`, который должен быть импортирован для их использования.

```
import math
# необходима спецификация функции
# через точку
print(math.sqrt(4))
from math import *
# импортировать все функции из math
# в глобальное пространство имен
print(sqrt(4))
# теперь специфицировать функцию
# не надо (и нельзя!)
```

Примеры функций и констант (в скобках — аналог в Паскале): `abs (abs)`, `math.sqrt (sqrt)`, `math.sin (sin)`, `math.cos (cos)`, `math.tan (tan)`, `math.atan (ArcTan)`, `math.atan2`, `round (round)`, `int (int)`, `ceil`, `floor`, `math.pi (pi)`, `random.seed (Randomize)`, за полным списком можно обратиться в [5]. Для простого преобразования числа в строку служит встроенная функция-конструктор `str`. Обратное преобразование (из строки в число) производится соответствующими функциями-конструкторами `int`, `long`, `float`, `complex`, при этом если целое число не укладывается в диапазон `int`, соответствующая функция автоматически сделает из него `long`.

“Массива” в низкоуровневом понимании Паскаля или C в Python нет, вместо него используются связные списки, имеющие тип `list`. Для абсолютного большинства задач разницы между этими двумя сущностями нет. Создаются списки при помощи уже упоминавшегося конструктора списков, а индексируются как обычные массивы (индексация всегда начинается с нуля, списки в Python реализованы так, что время доступа постоянно для любого элемента — O(1)).

```
a = [1, 2, 3, 4]
print(a[1])  # выведет 2
print(a[-1]) # выведет 4, отрицательные
             # индексы отсчитываются
             # с конца списка
```

В Python также поддерживается пришедшее из Фортрана взятие срезов списков. Синтаксис у него следующий: `список[инд_нач_элемента:инд_кон_элемента]`, конечный элемент не включается в список, любой из индексов может быть опущен, тогда список отрезается с самого начала или до самого конца.

```
a = [1, 2, 3, 4]
a[:3]              # [1, 2, 3]
a[1:3]             # [2, 3]
a[2:]              # [3, 4]
a[2:-1]            # [3]
a[2:-3]            # [] срезы в обратном
направлении дают пустой список
```

Списки в Python — объекты динамические, их можно расширять или ужимать во время выполнения, память для них выделяется автоматически. Чтобы узнать длину списка, можно использовать функцию `len`.

Присваивать значения можно как одиночным элементам, так и целым срезам, при этом список, присваиваемый срезу, может быть как короче, так и длиннее среза.

```
a = [1, 2, 3, 4]
a[0:2] = 0         # a = [0, 3, 4]
a[0:2] = [7, 8]    # a = [7, 8, 4]
a[0:2] = [1, 2, 3] # a = [1, 2, 3, 4]
```

Здесь примеры выполняются не последовательно, а каждый сам по себе для одного и того же исходного списка.

Необязательным аргументом среза является шаг.

```
a = [1, 2, 3, 4]
a[0:3:2]           # [1, 3]
```

Срезам с шагом можно присваивать только списки, равные по длине.

```
a = [1, 2, 3, 4]
a[0:3:2] = [7, 8]    # a = [7, 2, 8, 4]
a[0:3:2] = [1, 2, 3] # Ошибка!
```

Списки также могут состоять из разнородных элементов, например, `[1, "test", True]`.

Важно помнить, что переменные, хранящие списки, на самом деле являются ссылками на них (указателями, как в C).

```
a = [1, 2]
b = a
print(a is b)  # выведет True, так как
# a и b - один и тот же список
```

[5] Функция `ord` в Python работает не как в Паскале. Тут она возвращает ASCII-код переданного ей символа.

Для создания копии существующего списка следует использовать функцию copy из модуля copy.

```
from copy import copy
# импортировать copy из модуля copy
# в глобальное пространство имен
a = [1, 2]
b = copy(a)
print(a is b)
# выведет False, a и b - разные списки!
print(a == b) # выведет True, так как
# a и b равны поэлементно
```

К строкам применимы те же самые операции, что и к спискам, однако строки в Python — неизменяемые объекты и часть существующей строки изменить нельзя.

```
a = "Hello, World!"
a[4:-1:3]          # "oWl"
a[-3:]             # "ld!"
a[10:]             # "ld!"
a[65536:70000]     # "", это не ошибка
a[1] = 'a'         # Ошибка!
```

Еще одной отличительной особенностью строк в Python является то, что *все* экземпляры совпадающих побуквенно строк являются одной и той же строкой (для экономии памяти).

```
from copy import copy
a = "Hello!"
b = "Hello!"
print(a is b)  # выведет True
c = copy(b)
print(b is c)  # все равно выведет True
```

К счастью, это не является серьезной проблемой, так как гибкий механизм взятия срезов позволяет легко реализовать строковые операции практически любой сложности. К спискам и строкам применима операция конкатенации +. Строки можно также умножать на натуральное число:

```
print(2*"ma")  # выведет mama
```

Для символов (как я уже говорил, в Python это строки длины 1) применима функция ord, возвращающая ASCII-код символа, обратное действие совершается функцией chr.

```
a = 'p'
print(a == chr(ord(a))) # выведет True
```

Множества в Python совсем недавно вошли в сам язык из модуля стандартной библиотеки sets. Они имеют тип set и создаются из любых других объектов-коллекций, например списков, с помощью одноименной функции-конструктора. К множествам применимы побитовые и арифметические операции и отношения, имеющие смысл в исчислении множеств. Размер множества в Python ограничен только доступной оперативной памятью, в отличие от Паскаля.

```
a = set([1, 2, 3, 4])
b = set([3, 4, 5, 6])
a | b        # set([1, 2, 3, 4, 5, 6])
a & b        # set([3, 4])
a - b        # set([1, 2])
b - a  # set([5, 6])
a ^ b  # set([1, 2, 5, 6])
a <= b # a - подмножество b или само b?
a < b  # a - подмножество b?
```

Приведение типов

Вопрос приведения типов в Python тесно связан с объектно ориентированной природой языка (каждая сущность в Python является объектом), и подробнее я опишу его в соответствующем подразделе. Пока что лишь кратко рассмотрю основные его принципы.

Целые числа в случае чего приводятся к длинным целым, так что переполнения при целочисленных арифметических операциях никогда не происходит. Приведение целых к вещественным производится в том случае, если один из аргументов операции — вещественное число. Если длинное целое выходит за рамки диапазона вещественных чисел, возникает ошибка.

Вещественные и целые числа приводятся к комплексным, если одним из аргументов операции является комплексное число.

Для вывода с помощью функции print любые сущности не обязательно явно приводить к строкам, однако в данном случае такое поведение не является примером неявного приведения (почему это так, я расскажу в подразделе, посвященном объектно ориентированному программированию). На самом же деле неявного преобразования к строкам не происходит никогда.

Функции (они же процедуры)

Разницы между процедурой и функцией в Python нет: если функция не возвращает никакого значения, считается, что она возвращает None.

Принципы локализации имен примерно такие же, как и в Паскале, однако глобальные переменные простых типов (но не функции!) по умолчанию не видны на запись[6] внутри функций и должны быть явно описаны в ее теле с помощью спецификатора global (см. пример далее). Глобальные переменные составных типов видны внутри функций и на чтение, если при этом не происходит изменение типа переменной (если мы присвоим глобальной переменной спискового типа число, она при этом не изменится, а создастся локальная переменная с таким же именем. Однако индексация с присваиванием изменит глобальную переменную[7]). Рекомендуется не

[6] Детальный механизм этого заключается в том, что при присваивании глобальной переменной простого типа внутри функции просто создается локальная переменная с таким же именем. Можно сказать, что глобальные переменные простого типа передаются в функции “по значению”. Слово “global” заставляет переменные передаваться “по ссылке” в любом случае.

[7] Глобальные переменные составных типов передаются “по ссылке” только в том случае, если операции с ними внутри функции не изменяют их тип.

полагаться на эти правила и использовать глобальные переменные внутри функций как можно реже. Также, само собой разумеется, что имена формальных параметров перекрывают имена глобальных переменных.

Определяется функция с помощью ключевого слова `def`, после которого следуют имя функции, ее формальные параметры в скобках и двоеточие. В силу динамической типизации формальные параметры описываются просто своими именами. Тело функции описывается с отступом (который, как вы помните, должен быть равен всем другим отступам того же уровня). Для возврата значения используется ключевое слово `return`.

Опережающее описание (описание прототипа) функции в Python невозможно и не нужно, потому что функция, описанная в любом месте модуля, видна из любого другого его места.

```
def add(a, b):
    return a+b
print(add(4, 5))    # выведет 9
a = 4
def add_to_a(b):
    return a+b
print(add_to_a(5)) # выведет 9
a = 4
def add_to_a(b):
    a += b   # Ошибка!
    # Переменная a недоступна для записи
a = 4
def add_to_a(b):
    global a
    a += b
    # теперь все сработает правильно
a = [1, 2]
def change_a():
    # эта строка изменит глобальную a
    a[0] = 2
    a = 2 # а эта уже вызовет ошибку,
    # потому что a передана по
    # ссылке с условием сохранения
    # ее типа
a = [1, 2]
def change_a():
    global a
    a[0] = 2
    # теперь глобальная a поменяет тип
    a = 2
```

Передача фактических параметров в функции подчиняется практически тем же правилам, что и “передача” глобальных переменных:

- Переменные простых типов передаются “по значению”, способа передать их “по ссылке” нет, так как это считается плохим тоном.
- Переменные составных типов передаются “по ссылке” и могут быть изменены (без изменения типа) внутри функции. Если же внутри функции произойдет смена типа параметра, переданного “по ссылке”, просто будет создана локальная переменная с тем же именем, которая будет перекрывать параметр.

Параметры в Python могут иметь значения по умолчанию.

```
def calc(number, price=100):
    overall = number*price
    number = 5 # фактический параметр
    # не изменится, теперь
    # number - это локальная переменная
    return overall
print(calc(4, 30))  # выведет 120
print(calc(4)) # выведет 400, price=100
               # по умолчанию
def change_list(some_list):
    # здесь фактический параметр
    # изменится
    some_list[0] = 1
    some_list = 2     # а здесь нет
```

Поскольку типы фактических параметров становятся известны только во время выполнения, могут произойти различные ошибки. Например, в описанную выше функцию `change_list` можно передать число вместо списка. В таком случае возникнет ошибка времени выполнения при попытке индексации этого числа, и программа остановится.

Рекурсивные вызовы функций в Python осуществляются точно так же, как в Паскале.

Функции также являются и самостоятельным типом данных, при этом, в силу динамической природы Python, существует только один тип `function`, а не множество специфицированных типов, как в Pascal. Функции можно передавать в функции в качестве параметров и возвращать в качестве результатов.

```
def scaler(mul, div, a, b, c):
# функция scaler вернет значение a*b/c,
# при этом функции деления и умножения
# задаются пользователем
    return div(mul(a, b), c)
def make_adder(value):
    def adder(value2):
        return value + value2
    return adder
add_to_2 = make_adder(2)
print(add_to_2(3)) # выведет 5
```

Одним из минусов языка Python является то, что перегрузка функций не поддерживается, вместо этого рекомендуется прямо в теле функции явно проверять типы аргументов. Для проверки принадлежности переменной или результата вычисления выражения к некоторому типу используется встроенная функция `isinstance`.

```
print(isinstance(4, int))
# выведет True
print(isinstance(4, float))
# выведет False
```

Работа с файлами

Операции для работы с файлами в Python во многом позаимствованы из C. К сожалению, ра-

бота с ними в этом языке мне кажется довольно неудобной, в чем я виню вовсе не динамическую типизацию (как пытаются это делать многие критики языка Python), а неудачный дизайн файловых операций и его соотношение со стандартными типами. Файлы в Python не являются типизированными и представляются в виде текста, а все операции чтения и записи оперируют строками, поэтому на программиста ложится бремя ручного приведения нужных ему переменных и значений к требуемым типам, что, в свою очередь, также уменьшает скорость выполнения программ. Медленный ввод-вывод — бич Python и является одной из причин его недооцененности в прошлом. К счастью, разработчики вовремя осознали ошибку, и в Python 3 уже добавлен двоичный режим работы с файлами (то, что в Паскале называется "нетипизированными файлами").

В этом разделе я уже не смогу обойтись без использования специфических особенностей объектно ориентированного программирования, поэтому мне придется упомянуть о нотации через точку: методы (собственные функции каждого объекта) вызываются как *имя_объекта.имя_метода()* — для людей уже знакомых с ООП в этом нет ничего необычного, новичкам же, знающим об объектах только понаслышке, придется обратиться к соответствующей литературе.

В языке Python существует три основных режима работы с файлами:

• *r* — режим чтения. В данном режиме существующий файл открывается на чтение. Если файл не существует, то происходит ошибка;

• *w* — режим записи. Открывает новый или существующий файл на запись. Если файл существует, то его содержимое стирается;

• *a* — режим добавления. В этом режиме новый или существующий файл открывается для записи, однако в том случае, если файл существует, его содержимое не стирается, а указатель перемещается в самый конец.

Существуют также смешанные режимы работы с файлами:

• *r+w* — в этом режиме существующий файл открывается как на чтение, так и на запись, при этом его содержимое не стирается. Следует помнить, что файловый указатель в этом случае по умолчанию находится в самом начале файла, и операции записи будут затирать имеющееся содержимое;

• *r+a* — то же самое, что и *r+w*;

• *w+a* — то же самое, что и просто *w*.

В Python 3 существует еще один режим работы — *b* (от слова *binary* — "двоичный"), который позволяет работать с файлом как с нетипизированным в терминологии Pascal. Для представления нетипизированных данных в Python 3 введен новый стандартный тип `bytes`. Двоичный режим работы с файлами может быть скомбинирован с любым другим ранее описанным режимом. Замечу также, что простого доступа к внешним устройствам (как в TurboPascal под DOS) в Python нет, однако многочисленные сторонние библиотеки реализуют этот доступ.

Для открытия файла в Python служит стандартная функция `open`, имеющая несколько аргументов, большинство из которых необязательны. Первый и обязательный ее аргумент (строка) — это имя открываемого или создаваемого (в зависимости от режима) файла. Второй аргумент (тоже строка) — режим работы с файлом, по умолчанию равный "*r*", то есть по умолчанию все файлы открываются на чтение. Остальные аргументы касаются режима буферизации и кодировки (Python 3) и рассматриваться мной не будут. Желающие могут обратиться в [5] для более подробных сведений.

```
# открыть some_file.txt на чтение
a = open("some_file.txt")
# то же самое
a = open("some_file.txt", "r")
# создать или открыть и очистить
# файл для записи
b = open("any_file.txt", "w")
# открыть файл для чтения и записи
b = open("any_file.txt", "r+w")
```

Для чтения из файла имеются два основных метода: `read` и `readline`. Оба метода имеют необязательный параметр, ограничивающий число байтов, которые должны быть прочитаны. Метод `read` читает заданное число байт или все байты до конца файла и возвращает их в виде строки. Метод `readline` читает заданное число байт или же все байты до символа конца строки включительно. Допустим, у нас есть файл `"my_text.txt"`, содержащий две строки `Hello` и `World!`.

```
a = open("my_text.txt")
print(a.read()) # выведет "Hello",
# перевод строки, "World!", перевод строки
a = open("my_text.txt")
# открыть файл заново
print(a.readline())
# выведет "Hello", перевод строки
```

Для изменения работы с файловым указателем существуют методы `seek` и `tell`. Первый из них изменяет позицию файлового указателя, а второй говорит текущую позицию. Позиция файлового указателя представляется целым числом типа `long`. Обязательным аргументом `seek` является смещение, а необязательным — то, от чего отсчитывается это смещение. Этот необязательный аргумент может принимать три значения:

• 0 — значение этого параметра по умолчанию. Все смещения при этом отсчитываются от начала файла и, следовательно, могут быть строго положительными.

• 1. При этом значении смещения отсчитываются от текущей позиции файлового указателя и могут быть как положительными, так и отрицательными.

• 2. При этом значении смещения отсчитываются с конца файла и обычно являются отрицательными, хотя некоторые программные платформы (например, Linux) позволяют перемещение файлового указателя за конец файла. При этом либо файл расширяется, а пропущенное место заполняется нулями, либо происходит создание специфического объекта файловой системы — файла с “дыркой”.

Необходимо сказать, что не все файловые объекты имеют файловый указатель, примером может служить терминал ввода-вывода.

Для записи в файл существуют методы `write` и `writelines`. Первый метод принимает строку и записывает ее в файл, а второй принимает последовательность строк, например список, и тоже записывает ее в файл. Оба метода не добавляют символы конца строки, об этом должен заботиться программист.

Последним, но не менее важным методом для работы с файлами является метод `close`, не имеющий аргументов, закрывающий файл и записывающий в него все несохраненные изменения. Для записи изменений без закрытия файла используется метод `flush`. При выходе из программы все незакрытые файлы закрываются автоматически.

По аналогии со стандартными файлами `input` и `output` в Паскале, в Python имеются стандартные файлы `stdin` и `stdout` соответственно, описанные в модуле `sys`.

```
import sys
sys.stdout.write("Test\n")
# выведет "Test" и перевод строки
# на стандартный вывод
```

До сих пор я не использовал никакой функции для ввода данных из консоли. К сожалению, это еще одно слабое место Python. Для ввода существуют две встроенных функции — `input` и `raw_input`. Функция `input` — одна из центральных функций интерпретатора Python, она читает строку из консоли и выполняет ее как Python-выражение с помощью встроенной функции `eval`. При этом выполняться могут только выражения, но не операторы. Функция `raw_input` читает строку из консоли и возвращает ее “как есть”, обрезая лишь символ перевода строки в конце. Вся забота о преобразовании этой строки или разбиении ее на несколько и последующем преобразовании ложится на программиста.

```
a = int(raw_input())
# пытаемся преобразовать введенную
# строку в число
```

Множество встроенных функций для работы с файловой системой, таких, как создание и удаление каталогов, чтение атрибутов файла, создание ссылок на файл, кроссплатформенная работа с путями в Python, находится в стандартном модуле `os`. Любопытные и интересующиеся могут прочитать о нем в документации [5].

Модули

Этот параграф будет последним в моем сравнении Python с Паскалем, поскольку я уже осветил все основные сходства этих языков и хочу в следующем разделе перейти непосредственно к описанию тех новшеств, что отличают эти две технологии с довольно солидной разницей в возрасте, и рассказать о собственном неповторимом стиле написания программ на Python.

Как я сказал ранее, все файлы с исходным кодом в Python являются модулями. Для загрузки модуля в программу используется ключевое слово (оператор) `import`. При этом выражение `import имя_модуля` импортирует функции из модуля в соответствующее пространство имен, то есть функция должна быть вызвана как `имя_модуля.имя_функции()`. Можно переименовать пространство имен импортируемых функций следующим образом: `import имя_модуля as новое_имя`, после чего функции можно будет вызывать как `новое_имя.имя_функции()`. Для импортирования всех фукнций из модуля в общее (глобальное) пространство имен применяется следующий вариант `import`: `from имя_модуля import *`. Таким же образом можно импортировать лишь отдельные функции из модуля: `from имя_модуля import имена функций через запятую`. При этом можно переименовывать каждую импортируемую функцию с помощью ключевого слова `as`.

```
# Запутаем всех и импортируем синус
# как косинус, а косинус как синус!
from math import sin as cos, cos as sin
```

При загрузке модуля все выражения в нем, не находящиеся внутри определений, выполняются — это можно использовать для необходимой инициализации. Во время работы программы всегда определена специальная переменная `__name__`, в которой хранится имя текущего модуля. Для главной программы это имя равно `__main__`.

Кроме простых модулей, в Python поддерживаются модули составные, состоящие из нескольких подмодулей. Для этого файлы подмодулей располагаются в одном каталоге, имя которого и будет именем большого модуля. Если составному модулю нужна предварительная инициализация, ее код помещается в файл *__init__.py*. На этом я закончу разговор о модулях, поскольку их реализация в Python довольно проста и изящна; если вы хотите узнать больше, можете обратиться в [1], [3], [4], [5].

Python — не Паскаль!

В прошлом разделе я хотел показать, как можно решать задачи на Python с помощью методов и терминологии языка Паскаль. Однако Pascal был достаточно зрелым языком в то время, когда Гвидо ван Россум писал свою операционную систему, и многие способы программирования на нем в Python могут казаться устаревшими. В данной главе

я постараюсь показать, как же все-таки решать задачи методами Python, и добавлю целостности к вашему восприятию этого языка.

Объектно ориентированное программирование

Одной из ключевых концепций языка Python является понятие объекта. Я подразумеваю, что большая часть читателей знакома с объектным программированием, потому что объем и тема статьи не позволяют мне углубляться в этот предмет слишком сильно. Любая сущность в Python: числа, строки, списки, функции, модули — является объектом[8] и имеет набор свойств (полей) и методов (функций). Объектная система унаследована из языка Smalltalk и благодаря этому проста, но в то же время мощна и выразительна. Практически все свойства и методы являются публичными (public) в терминах Объектного Паскаля, все методы к тому же являются виртуальными. Член класса можно сделать приватным или "внутренним", если написать его имя начинающимся с двух символов подчеркивания (это не касается тех особых членов, чье имя и начинается, и заканчивается двумя подчеркиваниями). При этом он будет доступным публично с именем *_имякласса__имячлена*. Члены класса, чье имя начинается с одного символа подчеркивания, не считаются внутренними, однако говорится, что они "не входят в интерфейс класса", то есть являются служебными и не указываются в спецификации и документации. При этом забота о защите данных ложится на программиста и, как показывает практика, это не ухудшает качество кода.

Пару слов стоит сказать о распознавании типов-классов во время выполнения. Например, как функция понимает, что ей передали список, а не число? Сложный и неэффективный путь предполагает одновременную передачу переменной и спецификации ее типа. В Python же, как и в большинстве других динамических языков, используется так называемая "утиная типизация" (такое название произошло от шутливой фразы "*Если что-то выглядит как утка, плавает как утка и крякает как утка, то, вероятно, это утка*"). То есть, если объект поддерживает методы, присущие списку, то можно считать, что это список. Таким образом любой другой объект может имитировать поведение того же списка, на этом основана концепция итераторов и коллекций в Python, о чем я расскажу позднее.

Класс объявляется с помощью ключевого слова `class`, после которого идут имя класса, возможные родители в скобках (в Python поддерживается множественное наследование) и двоеточие. Тело класса описывается с отступом. Все методы имеют точно такой же вид, как и функции, но их первым обязательным аргументом всегда является `self`. В любом из методов могут создаваться новые свойства объекта (это свойство появится в данном конкретном экземпляре класса, но не во всех других).

```
class TestClass(object):
# наследуем от базового класса object.
# Хороший тон
  def some_method(self):
  # self - обязательный параметр
  self.a = 4
  # создаем новое свойство в объекте.
b = TestClass()
b.some_method()
```

Специальные функции, отвечающие за внутреннюю работу с объектом или отнесение его к какому-либо типу, имеют имена, окруженные двумя символами подчеркивания с обеих сторон, например, конструктор всегда имеет имя `__init__`.

```
Class AnotherClass(object):
    def __init__(self, value):
        # Создадим два свойства
        self.value = value
        self.a     = 4
b = AnotherClass(6)
print(b.value)          # выведет 6
print(b.a)              # выведет 4
```

Функции, реализующие операции (например, арифметические или индексацию), имеют специальные имена, такие, как `__add__`, `__sub__`, `__getitem__`, `__setitem__`.

```
class ListLike(object):
   def __getitem__(self, index):
   # реализация операции индексации
   # для чтения
         return 5
   def __setitem__(self, index, value):
   # реализация индексации для записи
         pass
a = ListLike()
print(a[6])      # выведет 5
print(a[10000])  # тоже выведет 5
```

Вышеописанный объект будет вести себя подобно бесконечному списку пятерок при индексации с чтением и ничего не будет делать при индексации с записью.

Представлением объекта в виде строки (для использования в функции `print`) управляет метод `__repr__`.

```
class Representable(object):
    def __repr__(self):
        return "I'm looking good!"
a = Representable()
print(a)  # выведет "I'm looking good!"
```

Полный список таких "специальных" методов можно узнать в документации или в учебниках [1], [4].

[8] Разумеется, для обеспечения надлежащей скорости выполнения программ числа на самом деле не являются объектами в своем внутреннем представлении, однако семантика языка поддерживает работу с ними как с полноправными объектами.

Разумеется, в Python существует такое понятие, как “статический” метод — метод, работающий не для каждого экземпляра класса, а для всего класса в целом. Кроме того, в качестве наследия Smalltalk существует особое понятие о “методе экземпляра класса” или просто “методе класса”. Поскольку все в Python является объектом, классы — не исключение, и методы класса как раз работают с классом как с неким “экземпляром”. Главным отличием его от статического метода является возможность доступа к внутренним членам класса. На способы описания этих методов вы можете посмотреть, например, в [2].

Весь список методов и свойств объекта можно посмотреть встроенной функцией `dir`. Узнать, имеет ли заданный объект нужное свойство или метод, можно функцией `hasattr`.

```
a = []
print(hasattr(a, "__getitem__"))
# выведет True, этот объект будет
# вести себя
# как список в операциях индексации
# для чтения
```

На этом я заканчиваю краткое описание классов и перехожу к рассказу о способах хранения и обработки наборов данных в Python.

Итераторы, генераторы и коллекции

Любой объект, предназначенный для хранения множества других объектов, принято называть *коллекцией*. В Python встроенные типы-коллекции многообразны, гибки и имеют мощные средства обработки. Коллекции делятся на:

• *Изменяемые* — такие коллекции, элементы которых могут быть изменены на ходу без создания новой коллекции. Среди стандартных типов Python изменяемыми коллекциями являются следующие:

◦ *Список*. Мы с вами уже довольно детально рассмотрели этот тип данных. Его элементы можно менять, можно добавлять новые элементы и удалять старые.

◦ *Словарь, или ассоциативный массив,* является вариантом массива, в котором индексами могут быть объекты любого типа (и даже сразу нескольких). С помощью такой структуры данных очень легко решить, например, задачу, какое слово встречается в тексте чаще всего.

◦ *Множество*. В него тоже можно добавлять объекты или удалять их.

• *Неизменяемые* — такие коллекции, которые не могут быть непосредственно изменены (однако сами объекты в них могут быть изменены с помощью своих методов). В Python к стандартным неизменяемым коллекциям относятся:

◦ *Строки* в Python являются неизменяемыми. Для изменения строки необходимо создавать новую. Отчасти это плата за очень эффективный механизм хранения строк.

◦ *Кортежи* являются неизменяемым вариантом списков и предназначены для эффективного выполнения программ в тех местах, где не требуются списки, поддерживающие изменение, например, при возвращении из функции нескольких значений.

◦ *Замороженные множества*, стандартный тип `frozenset`, являются полными аналогами множеств за исключением того, что объекты в них нельзя добавить или удалить.

Хранить данные в виде явных коллекций бывает неэффективно или даже не нужно, поэтому основным способом работы с большими объемами данных в Python являются итераторы. Итераторы — это объекты, предоставляющие последовательный доступ к объектам коллекции. Говоря строго техническим языком Python, итераторы — это объекты, реализующие метод `next`. Коллекция называется итерируемой, если она имеет метод `__iter__`, возвращающий итератор для данной коллекции (коллекция может быть и сама себе итератором, в таком случае `__iter__` просто возвращает `self`). Для превращения любой итерируемой коллекции в итератор используется стандартная функция `iter`. Цикл `for` неявно вызывает эту функцию для выражения, стоящего после `in`. Второй вариант применения функции `iter` — создание итератора, вызывающего заданную функцию до тех пор, пока она не возвратит заданное значение.

Для удобной работы с итераторами в Python имеется целый модуль `itertools`, в котором находятся функции, позволяющие комбинировать итераторы всевозможными способами, например, повторять итератор заданное число раз, склеивать несколько итераторов, применять функции к каждому элементу итератора, создавать и клонировать параллельные итераторы. Работа с этим модулем подробно описана в [2].

Важным новшеством, пришедшим в свое время из языка Icon, является концепция генераторов. Генераторы являются обычными функциями с одним лишь исключением: они не возвращают некоторое значение, а возвращают серию значений по необходимости, приостанавливая свое выполнение после выработки очередного значения. В Python генераторы просто создают объект с интерфейсом итератора. Генераторы являются примером *сопрограмм*.

```
def factorial(N):
# генерирует последовательность
# факториалов чисел от 1 до N
    f = 1
    for i in range(1,N+1):
        f = f*i
        yield f # вернуть очередное
  # значение и приостановить выполнение
fac_10 = factorial(10)
for value in fac_10:
    print(value)
# выведет последовательно факториалы
# от 1! до 10!
```

С помощью генераторов можно имитировать бесконечные последовательности.

Из функционального языка Miranda в Python пришли списочные выражения, позволяющие с помощью довольно простого синтаксиса автоматически создавать списки, не генерируя их последовательно в цикле. Синтаксис списочного выражения такой: `[выражение_с_переменными for i1 in итератор1 … for N-ная_переменная in итераторN if необязательные_условия]`. Это выглядит довольно сложно, поэтому я покажу пару простых примеров.

```
from math import *
sines = [sin(pi/2*x/90) for x in range(91)]
# генерируем таблицу синусов для углов
# от 0 до 90 включительно с шагом 1
muls  = [x*y for x in range(1,11) for
y in range(1,11) if x%2 == 0 if y%2 == 0]
# генерируем список произведений
# четных чисел от 1 до 10.
```

По аналогии со списочным выражением в Python существует генераторное выражение, результатом которого является не список, а генератор. Отличие в синтаксисе минимально — вместо квадратных скобок используются круглые. Преимущество генераторных выражений в том, что они не вычисляют все значения сразу, чем сильно экономят память при создании больших коллекций, к элементам которых необходим последовательный доступ. Аналогично встроенной функции `range`, создающей список, существует функция `xrange` с аналогичным синтаксисом, возвращающая генератор.

Другие “вкусности”

К счастью для пользователей, но к несчастью для автора краткого обзора, в Python существует еще много как встроенных, так и сторонних средств поддержки разных инструментов практического программирования. Отдельно хотелось бы выделить обработку строк с помощью регулярных выражений, модули для численной и символьной математики, модули для обработки изображений и создания мультимедийных приложений и компьютерных игр, поддержку многопоточности, поддержку сокетов, сетевых протоколов и баз данных, встроенный графический интерфейс, интерфейсы к библиотекам на других языках программирования и средства интроспекции интерпретатора. Большую часть из перечисленного я постараюсь осветить в примерах к статье.

Выводы и рекомендации

Как вы могли видеть на протяжении этой статьи, Python — довольно неоднозначный язык. С одной стороны, он очень гибок и имеет мощные и выразительные средства программирования, с другой — в нем до сих пор присутствуют довольно серьезные недочеты на уровне дизайна, это значит, что изменить их будет довольно сложно. Неоднозначна также ситуация с разделением языка на две ветви: второй и третий Python, поскольку значительное число сторонних модулей, составляющих залог успеха Python, пока что работают только с второй версией, позиционирующейся как устаревшая, а их разработчики не горят желанием переделывать всю свою работу, сталкиваясь с новыми, порой труднонаходимыми ошибками (и по-человечески я их понимаю). Может быть, через 4–5 лет эта ситуация хоть немного улучшится.

Что можно сказать касательно применения Python в школе в качестве первого языка программирования? Я очень скептически отношусь к этой идее. Да, всевозможные интерпретаторы языка Python являются свободным ПО и без всяких проблем работают под операционной системой Linux, а Pascal устарел уже довольно давно, и всякие попытки его воскресить в лице Object Pascal и Delphi выглядят не очень красиво. Однако Python — язык, который с виду кажется простым, требует некоторого опыта для четкого понимания принципов работы некоторых конструкций. Кроме того, являясь апологетом обучения программированию с помощью низкоуровневых языков, я считаю, что чрезмерная высокоуровневость Python может отрицательно повлиять на школьников: они не до конца будут понимать базовые принципы работы ЭВМ, они не смогут полностью понять формализмы программирования (Python все-таки не чистый язык, подобно Scheme). В конце концов после изучения Python у них элементарно может не быть желания использовать, если необходимо, другие языки, ставящие программиста в жесткие рамки, но дающие дополнительные преимущества, например, в эффективности. Вы спросите: что же тогда я бы мог порекомендовать для изучения в школе? На этот вопрос уже давно ответил Никлаус Вирт продолжением своих исследований в направлении Паскаля: такие языки, как Модула или тем более Оберон, идеально подходят для этих целей, и если найти кроссплатформенную и поддерживаемую среду программирования на Обероне довольно тяжело, то для Модулы-2 и Модулы-3 существуют переносимые и свободные реализации. По моему мнению, Python должен изучаться в рамках вузовской программы, а не школьной.

Каковы перспективы языка Python в сфере профессиональной разработки ПО? Здесь мое мнение будет прямо противоположно мнению об использовании Python в школе. Язык идеально подходит для решения многих реальных задач и обладает для этого всеми необходимыми средствами. Как правило, разработчики стараются следовать какому-либо определенному стилю программирования и использовать лишь то подмножество Python, которое

им необходимо, избавляясь от двусмысленных конструкций. Ключевую роль в успехе языка Python в индустрии играют простота использования в нем программ и библиотек на других языках и наличие большого количества уже имеющихся качественных модулей. Этот язык в своей работе используют такие гиганты, как Google, NASA, Industrial Light and Magic. Особенно Python зарекомендовал себя в области биоинформатики и биоинженерных исследований.

В заключение хочу сказать пару слов о своем опыте использования языка Python. Я успешно пользовался им в самых различных задачах: от прототипирования при разработке графических приложений и создания компьютерных игр-однодневок ради удовольствия до исследований в области как численного анализа, так и символьной математики. Неоднократно мне приходилось писать интерфейсы к библиотекам на языке C. Все это время мне удавалось избегать сложностей языка имеющихся в нем “by design”, а простой синтаксис и быстрый цикл проектирования-разработки-тестирования позволяли писать достаточно надежные программы. В целом этого достаточно, чтобы я мог порекомендовать Python в качестве основного языка для практикующего программиста.

Примеры

В этом разделе собраны примеры программ на языке Python. Логически все задачи разбиты на две группы: те, что охватывают стандартные языковые средства и семантику, и те, что используют возможности встроенных или сторонних модулей в основном для выполнения полезных практических действий, как то: рисование графических интерфейсов, работа с программами на других языках программирования, многопоточность и т.д.

Все примеры будут работать как в IDLE, так и при запуске из консоли. Список ссылок на необходимые дополнительные модули для некоторых примеров будет в конце.

1. Множественное присваивание и присваивание итераторов.

В Python можно сразу присвоить нескольким переменным несколько значений, для этого необходимо либо указать их через запятую, либо упаковать в любой объект с интерфейсом итератора.

```
a, b = 4, 5   # Значения через запятую
a, b = [4, 5] # Список
a, b = (4, 5) # Кортеж
a, b = "45"  # Строка, a = "4", b = "5"
a, b = range(4, 6)  # Список
a, b = iter(range(4, 6))
# Явный итератор
a, b = xrange(4, 6)  # Генератор
# Обмен двух переменных местами!!!
a, b = b, a
```

2. Условные операторы.

Простой пример двух возможных условных операторов.

```
a, b = 4, 5
if a < b:
    print("A is less than B")
else:
    print("A is not less than B")
a, b = 4, 5
if a < b:
    print("A is less than B")
elif a == b:
    print("A is equal B")
elif a > b:
    print("A is greater than B")
else:
    print("Something strange has
          happened!")
```

3. Циклы.

Пример двух возможных циклов в Python.

```
a = 10
while a > 0:
    print(a)
    a -= 1
a = ["just", "try", "python", "test"]
for s in a:
    print(s)
```

4. Функции: опциональные и ключевые аргументы.

Опциональным, или необязательным, аргументом функции называется такой аргумент, который может не передаваться в нее. В Python опциональные аргументы описываются с помощью обозначения `*args` *в заголовке функции. Ссылаться на них в теле можно с помощью кортежа* `args`*. Ключевыми аргументами называются такие аргументы, которые могут не передаваться в функцию, но при передаче для каждого такого аргумента указывается его имя. В Python ключевые аргументы в заголовке описываются обозначением* `**kwargs`*, а в теле доступны через словарь* `kwargs`*. Имена ключевых аргументов не должны пересекаться с именами других формальных аргументов. Также с помощью звездочек можно передать заранее сгенерированные списки или словари аргументов.*

```
# Функция с тремя обязательными
# аргументами, один из которых имеет
# значение по умолчанию, опциональными
# аргументами и ключевыми аргументами
def f(a, b, c = 0, *args, **kwargs):
    print(a, b, c, args, kwargs)
    # выведет кортеж этих значений
f(1)          # Ошибка!
f(1, 2)       # (1, 2, 0, (), {})
f(1, 2, 3)    # (1, 2, 3, (), {})
f(1, 2, 3, 4) # (1, 2, 3, (4,), {})
f(1, 2, 3, 4, 5) # (1, 2, 3, (4, 5), {})
```

```
f(1, 2, 3, 4, 5, q=6, w=7)     # (1, 2, 3, (4, 5), {'q': 6, 'w': 7})
f(1, 2, q=6, w=7)              # (1, 2, 0, (), {'q': 6, 'w': 7})
a = [4, 5, 6]
b = {'q': 7, 'w': 8, 'e': 9}
f(*a)                          # (4, 5, 6, (), {})
f(1, 2, *a)                    # (1, 2, 4, (5, 6), {})
f(1, 2, 3, *a)                 # (1, 2, 3, (4, 5, 6), {})
f(1, 2, 3, *a, **b)            # (1, 2, 3, (4, 5, 6), {'q': 7, 'e': 9, 'w': 8})
```

5. Функции: анонимные функции (лямбда-выражения).

Анонимные функции в Python описываются с помощью ключевого слова "lambda", после которого идет перечисление фактических параметров через запятую (справедливы все утверждения из предыдущего примера), двоеточие и тело функции. Тело анонимной функции должно быть выражением, следовательно, использование операторов в нем запрещено. Это поощряет использование функционального стиля при применении лямбда-выражений.

```
print( (lambda x, y=3: x+y)(4, 5) )  # 9
a = lambda x: x+1
print(a(6))                          # 7
```

6. Функции: коарность.

Коарностью называется возможность возврата из функции нескольких значений. Коарность в Python обеспечивается уже рассмотренным присваиванием итераторов. Обычно для возврата нескольких значений явно или неявно используется кортеж.

```
def some_func(a, b):
    return a, b
x, y = some_func(4, 5)       # x = 4, b = 5
print("%s, %s" % (x, y))
```

7. Функции: ООП.

Функции в Python являются объектами. Строго говоря, такие сущности называются функторами. Для определения собственного функтора необходимо описать в нем метод `__call__`.

```
class Adder(object):
    def __init__(self, a):
        self.a = a
    def __call__(self, x):
        return x+self.a
add_to_2 = Adder(2)
print(add_to_2(3))          # 5
```

8. Исключения.

Исключения в Python являются потомками класса Exception. Вызываются они ключевым словом `raise` *и могут иметь параметры. Для выполнения участков кода, в которых могут возникнуть исключения, существует конструкция* `try - except`*, где для каждого исключения используется свой* `except`*. Ветвь* `finally` *позволяет гарантированно выполнить какие-либо действия. Стоит сказать, что в Python в отличие от многих других языков исключения являются "дешевыми" и широко применяются.*

```
class MyError(Exception):
    def __init__(self, value=6):
        self.value = value
    def __str__(self):
        return str(self.value)
try:
    raise MyError(5)
except MyError as e:
    print("There is an error #" + str(e.value))
# Будет выведено "There is an error #5"
finally:
    print("This will be printed!")
raise MyError("test")
# Будет выведено:
# Traceback (most recent call last):
# File "<stdin>", line 1, in ?
# __main__.MyError: 'test'
```

9. Работа с изменяемыми последовательностями.

Изменяемые последовательности (например, списки) в Python имеют методы и функции, значительно упрощающие работу с ними.

```
a = [1, 2, 3]
len(a)                  # 3
1 in a                  # True
1 not in a              # False
a + a                   # [1, 2, 3, 1, 2, 3]
3*a                     # [1, 2, 3, 1, 2, 3, 1, 2, 3]
-3*a                    # []
min(a)                  # 1
max(a)                  # 3
a.append(-3)            # a = [1, 2, 3, -3]
a.count(3)              # 1, считает число элементов, равных заданному
a.append([4, 5])        # a = [1, 2, 3, -3, [4, 5]]
a.pop()                 # a = [1, 2, 3, -3], возвращает [4, 5]
a.pop(2)                # a = [1, 2, -3], возвращает 3
a.extend([4, 5]         # a = [1, 2, -3, 4, 5]
a.reverse()             # a = [5, 4, -3, 2, 1]
a.insert(1, 8)          # a = [5, 8, 4, -3, 2, 1]
a.index(8)              # 1, возвращает наименьший индекс элемента с заданным значением
a.sort()                        # a = [-3, 1, 2, 4, 5, 8]
a.sort(lambda a, b: b-a)        # a = [8, 5, 4, 2, 1, -3]
```

10. Стандартные функции высших порядков.

```
a = [1, 2, 3, 4]
b = [-1, -2, 3, 6]
map(lambda x, y: x+y, a, b)     # [0, 0, 6, 10]
map(None, a, b)                 # [(1, -1), (2, -2), (3, 3), (4, 6)]
filter(lambda x: x>0, b)        # [3, 6]
sum(a)                          # 10
reduce(lambda x, y: x*y, a, 1) # 24
zip(range(4), a)                # [(0, 1), (1, 2), (2, 3), (3, 4)]
zip(range(43545), a)            # [(0, 1), (1, 2), (2, 3), (3, 4)]
```

Подробнее об этих функциях можно прочитать в [2, 5] и литературе по функциональному программированию. С появлением списковых и генераторных выражений многие вещи можно сделать без этих функций.

11. Использование математических модулей: `numpy` и `scipy`.

Сторонние модули `numpy` *и* `scipy` *объединяют в себе большое количество подпрограмм, написанных на C и Фортране, и предназначены для высокопроизводительных математических вычислений.*

```
import numpy
import numpy.linalg as LinearAlgebra
a = numpy.matrix([[2, -3], [4, 1]])
b = numpy.matrix([[1, 6], [-3, -2]])
# Матричное произведение
r1 = a*b
# Определитель
r2 = LinearAlgebra.det(a)
# Обратная матрица
r3 = LinearAlgebra.inv(a)
# Собственные числа и собственные векторы
r4 = LinearAlgebra.eig(a)
# QR-разложение
r5 = LinearAlgebra.qr(a)
from math import *
import scipy.integrate as integrate
import scipy.optimize as optimize
# Определенный интеграл
r1 = integrate.quad(sin, 0, 1)
# или
r2 = integrate.romberg(sin, 0, 1)
# Безусловная минимизация
r3 = optimize.fmin(sin, 0)
# Условная минимизация
```

```
# Функция sin(x+y) при x^2 + y^2 < 1
r4 = optimize.fmin_cobyla(lambda x: sin(x[0]+x[1]), [0.0, 0.0],
                          \[lambda x: 1-(x[0]**2 + x[1]**2)])
# Неподвижная точка функции
r5 = optimize.fixed_point(cos, 0)
# Решение нелинейного уравнения sin^2(x) + ln(x) = 0
r6 = optimize.fsolve(lambda x: sin(x)**2 + log(x), 1)
```

12. Использование графических модулей: `matplotlib`.

Сторонний модуль `matplotlib` *предназначен для высококачественной отрисовки графиков функций и диаграмм.*

```
from math import *
import numpy
import matplotlib.pyplot as pyplot
x = numpy.arange(0, 2*pi, 0.1)     # Вещественный аналог range
y = numpy.sin(x)                   # Поэлементное применение синуса
pyplot.plot(x, y)
pyplot.show()
pyplot.clf()                       # Стереть график
import pylab
import matplotlib.pyplot as pyplot
x = pylab.randn(10000)             # Нормальное распределение
pylab.hist(x, 100)                 # Построить гистограмму
```

13. Работа с базами данных: SQLite.

Python поддерживает работу с базами данных по протоколу DB-API. Встроенный модуль умеет работать с библиотекой SQLite, сторонние модули дают возможность обращаться к MySQL, PostgreSQL, Oracle, Firebird и другим СУБД. API для всех СУБД стандартизовано и выглядит примерно так же, как и для SQLite.

```
import sqlite

db  = sqlite.connect(database="test")  # База создастся автоматически
dbc = db.cursor()
dbc.execute("""CREATE TABLE t1 ( name VARCHAR[40], age INTEGER );""")
dbc.executemany("""INSERT INTO t1 VALUES (%s, %s);""", \
                (["Ivan", 34], ["Andrew", 40]))
db.commit()
dbc.execute("""SELECT name, age FROM t1 ORDER BY name;""")
for name, age in dbc.fetchall():
    print(name, age)
```

14. Работа со строками: регулярные выражения.

Регулярные выражения, активно использующиеся в Perl, есть и в Python, во встроенном модуле `re`.

```
import re
pattern = r"[01]+"
c       = re.compile(pattern)
print(c.findall("541104310"))        # ['110', '10']
```

Подробно о регулярных выражениях можно прочитать в соответствующей литературе.

15. Работа со строками: модуль `Spipat`.

Регулярные выражения имеют довольно специфический синтаксис, который не всем нравится, поэтому я рассмотрю еще один сторонний модуль для работы со строками, использующий концепции, пришедшие из языков Snobol, Spitbol и Icon.

```
import spipat
pattern = spipat.span("01")
s = "541104310"
b = pattern.search(s)
print(b.match())            # '110'
print(b.start())            # 2
print(b.end())              # 5
```

16. Работа с изображениями.

*Сторонний модуль PIL (**P**ython **I**maging **L**ibrary) позволяет работать с изображениями в различных форматах как с массивами (линейными или двухмерными). При этом в двухмерный массив можно вносить изменения напрямую.*

```
import Image
a = Image.open("lena.jpg")
b = a.getdata()                 # Как линейный массив
c = b.pixel_access()            # Как двухмерный массив
print(b[0])                     # (227, 134, 117)
print(c[(0, 0)])                # (227, 134, 117)
r1 = a.histogram()              # Составляет гистограмму цветов
a.size                          # (204, 204)
# Небольшой скос и перемещение изображения – аффинное преобразование
r2 = a.transform(a.size, Image.AFFINE, (1, 0, -10, 0.1, 1, -10))
a.save()                        # Сохранить изменения в файл (однако, у нас их нет)
```

17. Мультимедийные приложения.

Один из сторонних модулей для создания мультимедийных приложений и компьютерных игр — Pygame, основанный на библиотеках SDL.

```
import pygame
import time
pygame.init()
screen = pygame.display.set_mode((640, 480)) # Создать окно для отображения
im     = pygame.image.load("lena.jpg")       # Загрузить изображение
# Вывести изображение в точке 100, 100
screen.blit(im, (100, 100))
# Нарисовать красный круг с центром в 100, 100 и радиусом 50
pygame.draw.circle(screen, (255, 0, 0), (100, 100), 50)
# Вывести все на экран
pygame.display.flip()
# Загрузить музыку
pygame.mixer.music.load("test.ogg")
# Начать проигрывание
pygame.mixer.music.play()
# Приостановить проигрывание
time.sleep(1)
pygame.mixer.music.pause()
raw_input()
```

18. Работа с данными: XML и YAML.

Формат данных XML поддерживается в стандартной библиотеке. YAML поддерживается с помощью модуля PyYAML. Однако для XML удобнее всего использовать модуль LXML.

```
import lxml.etree as ET
root = ET.Element("html")
body = ET.SubElement(root, "body")
body.set("bgcolor", "#ffffff")
body.text = "Hello!"
tree = ET.ElementTree(root)
tree.write("some_page.xhtml")
import lxml.etree as ET
tree = ET.parse("some_page.xhtml")
# XPath
print(tree.findtext("body"))  # Hello!
root = tree.getroot()
for element in root:
    print element             # Выведет один элемент body
import yaml
document = """
    a: 1
    b:
        c: 3
        d: 4
"""

doc = yaml.load(document)     # {'a': 1, 'b': {'c': 3, 'd': 4}}
print(doc)
doc['e'] = 7
print(yaml.dump(doc))         # 'a: 1\nb: {c: 3, d: 4}\ne: 7\n'
```

19. Многопоточность.

Python имеет встроенные модули поддержки многопоточности и собственную (не зависящую от операционной системы) реализацию потоков.

```
import threading
def hello(n):
    print("Hello from thread #%s" % n)
t1 = threading.Thread(target=hello, name="t1", kwargs={"n": 1})
t2 = threading.Thread(target=hello, name="t2", kwargs={"n": 2})
t1.start()
t2.start()
```

Другие инструменты модуля threading, например блокировки и семафоры, подробно рассмотрены в [2].

20. Многопроцессность.

Python имеет модуль поддержки многопроцессности, аналогичный модулю поддержки многопоточности. Разница лишь в том, что многопроцессность использует процессы операционной системы.

```
import multiprocessing as mp
def hello(n):
    print("Hello from process #%s" % n)
p1 = mp.Process(target=hello, name="p1", kwargs={"n": 1})
p2 = mp.Process(target=hello, name="p2", kwargs={"n": 2})
p1.start()
p2.start()
```

21. Графический интерфейс пользователя.

Python позволяет с легкостью разрабатывать GUI. Здесь я рассмотрю простейший встроенный GUI — Tkinter, основанный на Tcl/Tk. К слову, IDLE использует именно Tkinter.

```
from Tkinter import *           # tkinter для Python 3
class App:
    def __init__(self, master):
        frame = Frame(master)
        frame.pack()
        Label(frame, text="Hello, Tkinter!").pack()
        self.pressme = Button(frame, text="Press me!", \
                              command=self.pressed)
        self.pressme.pack()
    def pressed(self):
        print("Oh, hi!")
root = Tk()
app  = App(root)
root.mainloop()
```

Пример нарисует окошко с надписью и кнопкой, при нажатии на которую в консоль будет выводиться текст.

Примеры алгоритмов на Python

1. Быстрая сортировка.

Для простоты в качестве опорной точки берется элемент с индексом 0.

```
def quicksort(data):
    if len(data) <= 1:
        return data
    else:
        return (quicksort([y for y in data[1:] if y < data[0]]) +
                data[:1] +
                quicksort([y for y in data[1:] if y >= data[0]]))
```

2. Сортировка слиянием.

Еще один быстрый алгоритм сортировки.

```
def merge(left, right):
    result = []
    while left and right:
        if left[0] <= right[0]:
            result.append(left.pop(0))
        else:
            result.append(right.pop(0))
    if left:
        result.extend(left)
    if right:
        result.extend(right)
```

```
    return result
def mergesort(data):
    if len(data) <= 1:
        return data
    else:
        left   = mergesort(data[: len(data)/2])
        right  = mergesort(data[len(data)/2 :])
        return list(merge(left, right))
```

3. Решето Эратосфена.

Алгоритм генерации простых чисел. Реализован в виде генератора.

```
def sieve(n):
    is_prime = [False] * 2 + [True] * (n-1)
    for p in range(n+1):
        if is_prime[p]:
            yield p
            for i in range(p*p, n+1, p):
                is_prime[i] = False
```

4. Факторизация числа.

Разбитие числа на простые сомножители. Использует процедуру `sieve` *из предыдущего примера.*

```
import sys
def sieve(n):
    is_prime = [False] * 2 + [True] * (n-1)
    for p in range(n+1):
        if is_prime[p]:
            yield p
            for i in range(p*p, n+1, p):
                is_prime[i] = False
def factorize(n):
    for p in sieve(n):
        if p*p > n: break
        while n % p == 0:
            yield p
            n /= p
    if n > 1:
        yield n
```

5. Задача о Ханойских башнях.

```
def hanoi(n, a, b, c):
    if n != 0:
        hanoi(n-1, a, c, b)
        print("Moving from %s to %s" % (a, b))
        hanoi(n-1, c, b, a)
```

6. Развертывание списка.

Превращение списка с вложенными списками в линейный список. Для экономии памяти списковое выражение лучше заменить генераторным.

```
def flatten(L):
    return sum([[x] if not isinstance(x, list) else flatten(x) for x in L], [])
```

Список литературы

1. *Лутц М.* Изучаем Python, 4-е издание. Пер. с англ. СПб.: Символ-Плюс, 2011, 1280 с.: ил., ISBN 978-5-93286-159-2.

2. *Сузи Р.А.* Язык программирования Python: учебное пособие. М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2007, 326 с.: ил., табл., ISBN 978-5-94774-711-9, 978-5-9556-0109-0.

3. *Ван Россум Г., Дрейк Ф.Л.Дж., Откидач Д.С.* Язык программирования Python. Черновик [Электронный ресурс], 2001, 454 с., URL: *http://www.python.ru/files/book-ods.pdf*.

4. *Pilgrim M.* Dive into Python [Электронный ресурс], 2004, URL: *http://www.diveintopython.net/toc/index.html*.

5. Python Documentation. — Официальная документация по языку Python [Электронный ресурс], URL: *http://docs.python.org*.

6. *Фаронов В.В.* Turbo Pascal: Учебное пособие. СПб.: Питер, 2010, 368 с.: ил. ISBN 978-5-469-01295-5.

ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ г. МОСКВЫ
ИЗДАТЕЛЬСКИЙ ДОМ «ПЕРВОЕ СЕНТЯБРЯ»

МОСКОВСКИЙ ПЕДАГОГИЧЕСКИЙ МАРАФОН УЧЕБНЫХ ПРЕДМЕТОВ 2012

20 МАРТА – 13 АПРЕЛЯ

РАСПИСАНИЕ ДНЕЙ ПЕДАГОГИЧЕСКОГО МАРАФОНА

20 марта	**Открытие** День классного руководителя
21 марта	День школьного психолога
22 марта	День здоровья детей, коррекционной педагогики, логопеда, инклюзивного образования и лечебной физической культуры
	День учителя технологии (ЦО № 293)
23 марта	День учителя начальной школы (день первый)
24 марта	День учителя начальной школы (день второй)
25 марта	День дошкольного образования
27 марта	День учителя географии
28 марта	День учителя химии
	День учителя ОБЖ (Кадетская школа № 1784)
29 марта	День учителя биологии
30 марта	День учителя информатики
31 марта	День учителя физики
1 апреля	День учителя математики
3 апреля	День учителя истории и обществознания
4 апреля	День учителя МХК, музыки и ИЗО
5 апреля	День школьного и детского библиотекаря
6 апреля	День учителя литературы
7 апреля	День учителя русского языка
8 апреля	День учителя английского языка
10 апреля	День учителя французского языка
11 апреля	День учителя немецкого языка
12 апреля	День учителя физической культуры
13 апреля	День школьной администрации **Закрытие**

marathon.1september.ru

! Обязательная предварительная регистрация на все дни Марафона откроется 20 февраля 2012 года на сайте marathon.1september.ru.

! Каждый участник Марафона, посетивший три мероприятия одного дня, получает официальный именной сертификат (6 часов)

В дни Марафона ведущие издательства страны представляют книги для учителей

Начало работы – 9.00. Завершение работы – 15.00

УЧАСТИЕ БЕСПЛАТНОЕ. ВХОД ПО БИЛЕТАМ

РЕГИСТРИРУЙТЕСЬ, РАСПЕЧАТЫВАЙТЕ СВОЙ БИЛЕТ И ПРИХОДИТЕ!

Место проведения Марафона: лицей № 1535, ул. Усачева, дом 50 (в 3 минутах ходьбы от станции метро «Спортивная»)
Место проведения Дня учителя технологии: ЦО № 293, ул. Ярославская, д. 27 (ст. метро «ВДНХ»)
Место проведения Дня учителя ОБЖ: Кадетская школа № 1784, ул. Скаковая, д. 20 (ст. метро «Динамо», «Белорусская»)

По всем вопросам обращайтесь по телефону **8-499-249-3138** или по электронной почте: marathon@1september.ru

О чем это?
В статье имеется несколько заданий из серии “ломаем голову”. Их можно использовать как на внеклассных мероприятиях, так и на уроках, если хочется на несколько минут отвлечься от текучки.

И это все?!
На диске имеется презентация. Чтобы использовать материалы статьи, достаточно просто захотеть ☺.

Вопросы по информатике для проведения школьных конкурсов “Что? Где? Когда?” и “Брейн-ринг”

Д.М. Златопольский,
Москва

1. Вы, конечно, видели старинный русский счетный прибор — счеты. А знаете ли вы, что в их конструкции элементы, перемещающиеся по проволоке, называли “косточки”? Почему?

2. *Ведущий*: “Внимание — на экран (на мониторе компьютера или с помощью проектора показывается изображение счетов):

Вы, конечно, узнали старинный русский счетный прибор — счеты. Обратите внимание на то, что в каждом полном ряду из десяти косточек две косточки окрашены в более темный цвет, чем остальные 8. Для чего это сделано?”

Примечание. При наличии счетов можно показать участникам конкурса их.

3. В обязанности одного человека входило считывание с прибора данных о количестве информации в битах и запись в журнал соответствующего количества байтов. После нескольких измерений он увидел на индикаторе прибора следующее[1]:

и записал в журнал — 106. Почему?

Вариант вопроса (отличается изображением на индикаторе)

[1] Соответствующее изображение показывается участникам конкурса.

4. Может ли быть, что 23 + 2 = 1?
5. Может ли быть, что 59 + 2 = 1?
6. Что обозначают следующие изображения:

1)

3,5

2)

0,8

7. Американский ученый и изобретатель Уильям Кейстер в ноябре 1950 года подал патент на разработанное им автоматическое устройство, работающее на реле, с индикаторной панелью, состоящей из девяти элементов[2]. В какую игру мог играть этот автомат?

8. На этом поле нельзя играть в футбол и другие игры, сеять, проводить битвы, гулять. Что это за поле?

Вариант вопроса

На нем нельзя играть в футбол и другие игры, сеять, проводить битвы, гулять. О чем идет речь?

Ответы

1. Прообразом известных нам счетов был старинный русский счетный прибор — так называемый "дощаной счет". Он состоял из одного или двух неглубоких ящиков, поперек которых натянуты веревки или проволоки с надетыми на них фруктовыми косточками. До его создания для вычислений также применялись косточки, которые выкладывались по рядам (как бы мы сказали сейчас, "по разрядам").

2. В более темный цвет окрашены пятая и шестая костяшки. Это сделано для наглядности вычислений. Двухцветная окраска костяшек облегчала откладывание цифр, поскольку, согласитесь, что, например, четыре светлых костяшки и две темных на левой стороне быстрее определяются как цифра 6, чем отсчитывание шести одноцветных костяшек. Такая окраска позволяла также очень быстро определить, какое число набрано на счетах.

3. В приборе для указания тех или иных цифр используется так называемая "7-сегментная индикация". Этот человек знал (после нескольких измерений), что в первой цифре средний горизонтальный сегмент индикатора не загорается, и фактическое значение равно 848 битам. Поэтому он и записал в журнал значение 106 байт.

Ответ на вариант вопроса — тот же. Здесь дополнительная подсказка — начальная цифра показания равна нулю, чего быть не может, так как первый слева элемент индикатора ничего не показывает.

4. Такое может быть при измерении времени. Когда к 23 часам прибавляют 2 часа, то получается 1 час ночи.

5. Может (см. ответ на предыдущий вопрос). Когда к 59 минутам прибавляют 2 минуты, то получается 1 минута следующего часа. Аналогично, когда к 59 секундам прибавляют 2 секунды, то получается 1 секунда следующей минуты.

6. Первое изображение обозначает повышение какого-либо показателя на 3,5 пункта, процента и т.п., второе — понижение на 0,8.

Примечание. Определенной "подсказкой" в изображениях, кроме чисел, является цвет заливки — зеленый соответствует чему-то "положительному", улучшающему, разрешающему (например, цвет сигнала светофора), красный — отрицательному, запрещающему, ухудшающему.

7. В "крестики-нолики". (Источник информации указан в сноске 2.)

8. Речь идет о *поле* — одном из основных понятий баз данных. Так называют отдельную характеристику объектов, представленных в таблице (это может быть фамилия, имя, рост, номер класса и т.п.).

Что такое A4?

Задание может предлагаться двум или более командам — участникам викторин, конкурсов и других внеклассных мероприятий по информатике. Победителем считается команда, приведшая наибольшее число примеров.

Приведите примеры использования в различных предметных областях, в том числе и в информатике, следующих двух обозначений:

A4 a4

Возможные ответы

1. Адрес ячейки в электронной таблице.
2. Имя переменной, константы, процедуры или функции в программировании.
3. Имя файла или папки.
4. Обозначение клетки на шахматной доске.
5. Формат листа писчей бумаги или бумаги для черчения.
6. Модель легкового автомобиля фирмы Audi.

[2] Источник — *Шилов В.В.* Удивительная история информатики и автоматики. М.: ЭНАС, 2011.

Программа развития профессионально-личностных компетенций педагога

В программу входят следующие модульные курсы под общим названием «**Навыки профессиональной и личной эффективности**»:

- **Тайм-менеджмент,** или Как эффективно организовать свое время
- **Тайм-менеджмент для учеников,** или Как научить школьников организовывать свое время
- **Приемы конструктивного разрешения конфликтных ситуаций,** или Конфликты в нашей жизни: способы решения
- **Профессиональное выгорание,** или Как сохранить здоровье и не «сгореть» на работе
- **Стресс-менеджмент,** или Приемы профилактики и преодоления стресса
- **Управление имиджем,** или Как создать свой стиль
- **Искусство договариваться,** или Как понять других людей и донести свою точку зрения

6 часов –

нормативный срок освоения каждого модуля. Форма обучения – дистанционная. После окончания модуля выдается свидетельство установленного образца.

Стоимость одного модульного курса:

– 300 рублей – заявки на обучение на сайте edu.1september.ru

– бесплатно – для педагогических работников образовательных учреждений, участвующих в общероссийском проекте «Школа цифрового века – 2012/13». Подробности на сайте digital.1september.ru

годовая подшивка журнала «ИНФОРМАТИКА»

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

Цена с доставкой – 699 РУБ.

Вы можете оформить заказ:

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

Также можно приобрести:

ПОДШИВКИ БУМАЖНЫХ ВЕРСИЙ ИЗДАНИЙ НА КОМПАКТ-ДИСКАХ ЗА 2003–2010 ГОДЫ

Цена с доставкой:
2003–2007 год – 299 РУБ.
2008–2009 год – 399 РУБ.
2010 год – 499 РУБ.

ВИДЕОЗАПИСИ ЛЕКЦИЙ И МАСТЕР-КЛАССОВ ВЕДУЩИХ ПРЕПОДАВАТЕЛЕЙ

Цена с доставкой – от 299 РУБ.

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

GAMES.EXE

Математические фокусы — мистические способности или алгоритм?

С.Л. Решко,
учитель информатики школы № 773,
Алексей Жуков,
ученик 6-го класса школы № 773, Москва

В наше время существует много вариантов проведения досуга, начиная с самого простого и привычного — просмотра телевизионных передач, занятий спортом, посещения музеев, театров, других объектов культуры и т.д. и заканчивая наиболее "времяпоглощающим" занятием — Интернетом.

"Арифметика, сиречь наука числительная, с разных диалектов на славенский язык приведеная и во едино собрана и на две книги разделена… Сочинися сия книга чрез труды Леонтия Магницкого"

Алексей Жуков: "Я задался вопросом: "А как проводили время наши предки, когда у них не было ни телевизоров, ни радио, ни кино, ни тем более Интернета?". Что могло их развлечь? И мне хочется сегодня рассказать вам о математических фокусах, которые в те времена были более популярны, нежели сейчас. Так как образование в настоящее время находится на гораздо более высоком уровне и является обязательным для каждого гражданина нашей страны, то понятно, что во времена Петра I и даже М.Ю. Лермонтова, о которых речь пойдет дальше, математические фокусы производили большее впечатление, а людям, их демонстрирующим, приписывались магические способности".

Вначале нам хочется рассказать о Леонтии Филипповиче Магницком — авторе первого учебника по математике, вышедшего в России в 1703 г.

Л.Ф. Магницкий и его "Арифметика"

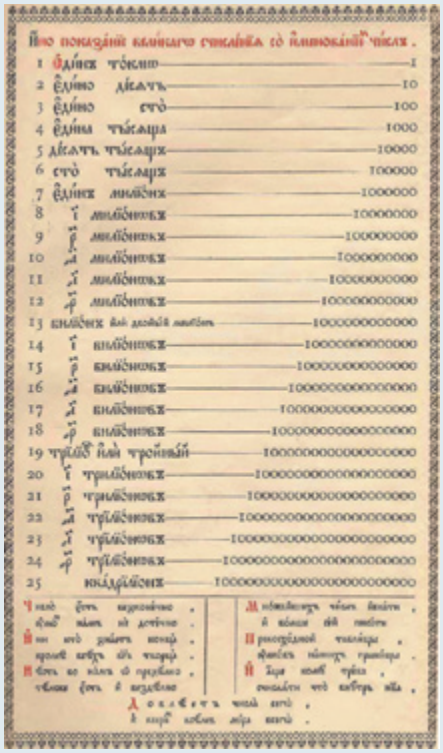
Книга эта содержит начала математических знаний того времени: арифметики, алгебры, геометрии и тригонометрии. В конце книги имеется снабженный большим числом таблиц раздел, посвященный морскому делу. Большую часть места, как указывает и заглавие книги, автор посвящает арифметике. В царствование Петра I, когда вышла в свет книга, в России происходил быстрый рост промышленности и торговли, переворот в военной технике. Стране потребовались образованные люди в значительно большем количестве, чем в предшествующие десятилетия. Был создан ряд технических учебных заведений, первым из которых была "школа навигацких и математических наук", открытая в Москве в Сухаревой башне в 1701 г. Учащимся в ней в первую очередь и предназначалась книга Магницкого. В течение полустолетия "Арифметика" с честью выполняла свою роль, став пособием для всех русских людей, которые стремились к математическому образованию.

Л.Ф. Магницкий

Об авторе этой замечательной книги мы знаем очень немного. Леонтий Филиппович Магницкий (Телятин) родился 9 июня 1669 года, умер в 1739 году. Надгробная надпись на могиле Магницкого, сделанная его сыном, рассказывает, что "Петр I многократно беседовал с ним о математических науках и был так восхищен глубокими познаниями его, что называл его магнитом и приказал писаться Магницким (имея в виду его способность притягивать знания подобно магниту)". "Какое он имел прозвище до этого, то даже ближним

его не известно”, — читаем в раннем его жизнеописании.

В “Арифметике” Магницкого есть особый раздел “О утешных неких действах, чрез арифметику употребляемых”, начинающихся с указания, что, следуя примеру арифметиков, автор помещает его в свою книгу для утехи и особенно для изощрения ума учащихся, хотя эти забавы, по его мнению, “и не зело нужные”. Мы уверены, что перевод последнего текста на современный русский язык не требуется. Заметим лишь, что Магницкий употребляет не очень привычное нам слово “забава” (развлечение, игра), которое будем использовать и мы.



Первая забава

Один из находящихся в компании восьми человек берет кольцо и надевает на один из пальцев на определенную фалангу. Требуется угадать, у кого, на каком пальце и на какой фаланге находится кольцо.

Показывающий фокус просит участников произвести следующие действия:

1) присвоить каждому человеку условный номер (от 1 до 8);

2) пронумеровать пальцы (от 1 до 10);

3) пронумеровать фаланги пальцев (от 1 до 3[1]);

4) номер человека, у которого находится кольцо, удвоить;

5) к полученному результату прибавить 5;

6) умножить полученный результат на 5;

7) прибавить номер пальца;

8) приписать 0 к полученному числу справа;

9) прибавить номер фаланги;

10) вычесть 250.

По названному участниками полученному результату отгадывающий может определить все, что нужно.

Секрет забавы

Пусть номер человека, у которого находится кольцо, — a, номер пальца — $в$, номер фаланги — c.

Выполним действия 4–10 в общем виде:

$a \times 2 = 2a$

$2a + 5$

$(2a + 5) \times 5 = 10a + 25$

$10a + 25 + в = 10a + в + 25$

$(10a + в + 25) \times 10 =$

$= 100a + 10в + 250$

$100a + 10в + 250 + c =$

$= 100a + 10в + c + 250$

$100a + 10в + c + 250 - 250 =$

$= 100a + 10в + c$

В полученном числе:

цифра сотен — номер человека;

цифра десятков — номер пальца;

цифра единиц — номер фаланги.

Предлагаем читателям проверить “работу” забавы на каком-либо конкретном примере.

Конечно, для произведения нужного эффекта необходимы актерские данные. Но все, что необходимо для отгадывания, называют сами зрители в конце фокуса.

Вторая забава

Пронумеруем дни недели, начиная с воскресенья: первый, второй и так далее до седьмого (до субботы). Играющий (или играющие) задумывает (задумывают) день. Нужно угадать, какой день задумали.

Отгадывающий предлагает выполнить следующие действия:

1) удвоить номер задуманного дня;

2) прибавить к полученному результату 5;

3) умножить новый результат на 5;

4) приписать 0 к полученному числу справа;

5) вычесть 250.

В конце отгадывающему остается только разделить на 100 полученный результат. Ответ — искомый номер дня недели.

Пусть, например, задумана пятница — шестой день. Играющий выполняет следующие действия:

1) $6 \times 2 = 12$;

2) $12 + 5 = 17$;

3) $17 \times 5 = 85$;

4) $85 \rightarrow 850$;

5) $850 - 250 = 600$,

— и объявляет результат. По нему отгадывающий легко вычисляет номер задуманного дня.

Предлагаем читателям доказать “секрет” забавы в общем виде.

Нетрудно увидеть, что “механизмы” двух рассмотренных забав очень похожи. Но если фокусы будут разнесены во времени, сходства никто не заметит…

Математическая забава Михаила Юрьевича Лермонтова

Известно, что великий русский поэт М.Ю. Лермонтов был большим любителем математики и в своих вольных и невольных переездах из одного

[1] У человека каждый палец, кроме большого, состоит из трех фаланг (основной, средней и ногтевой), а большой — из двух. Но для единообразия при демонстрации фокуса следует и большой палец условно разделить на три части. — **Прим. ред.**

места службы в другое всегда возил с собою учебник математики. Он также любил развлекать своих знакомых математическими фокусами. Суть его "забав" сводилась к тому, что задуманное число на каком-либо этапе вычислений он предлагал вычесть, а все математические действия сводились к действиям над названными им числами. В результате — вычисления не зависели от задуманного числа, он с легкостью угадывал полученный результат.

М.Ю. Лермонтов

Например:

1) к задуманному числу требовалось прибавить 150;

2) из полученного результата вычесть 36;

3) вычесть задуманное число;

4) умножить полученный результат на 5;

5) разделить на 2.

Итог вычислений можно "отгадать" — 285.

Эти действия можно описать с помощью следующих формул:

$a + 150$

$a + 150 - 36 = a + 114$

$a + 114 - a = 114$

$114 \times 5 = 570$

$570 : 2 = 285$.

Математический фокус с запоминанием больших чисел

Но не все поддается алгоритмизации. Существуют люди, у которых на самом деле есть некие математические способности.

Алексей Жуков: "Знакомясь с различными математическими фокусами, я открыл у себя такую способность. За несколько минут я способен запомнить большое количество 7- и даже 8-значных чисел.

Возьмем 30 больших чисел и присвоим каждому числу номер, как показано в таблице:

Номер числа	Число	Номер числа	Число
№ 1	1 123 583	№ 6	51 673 033
№ 11	2 246 066	№ 16	52 796 516
№ 21	3 369 549	№ 26	53 819 099
№ 2	11 235 831	№ 7	61 785 381
№ 12	12 358 314	№ 17	62 808 864
№ 22	13 471 897	№ 27	63 921 347
№ 3	21 347 189	№ 8	71 897 639
№ 13	22 460 662	№ 18	72 910 112
№ 23	23 583 145	№ 28	73 033 695
№ 4	31 459 437	№ 9	81 909 987
№ 14	32 572 910	№ 19	82 022 460
№ 24	33 695 493	№ 29	83 145 943
№ 5	41 561 785	№ 10	91 011 235
№ 15	42 684 268	№ 20	92 134 718
№ 25	43 707 741	№ 30	93 257 291

И если бы у меня была возможность продемонстрировать свою способность читателям, я попросил бы выбрать номер числа. Всего несколько секунд для визуального воспроизводства в памяти — и я назвал бы вам число, соответствующее данному номеру".

Но все это конечно же шутка. На самом деле число не запоминается, а вычисляется по следующему алгоритму.

Пусть номер числа равен 5.

1. К номеру числа прибавляем 9: $5 + 9 = 14$

2. Из числа 14 получим "обращенное" — 41. Это количество миллионов: 41000000.

Остается выполнить еще несколько шагов для поочередного заполнения других разрядов числа.

3. Складываем цифры полученного числа:

$4 + 1 = 5$. Вычислили разряд сотен тысяч.

4. Опять складываем числа. На этом шаге алгоритма второе слагаемое предыдущего шага становится первым слагаемым, а сумма, полученная на предыдущем шаге, становится вторым слагаемым:

$1 + 5 = 6$. Это десятки тысяч.

5. Складываем числа аналогично тому, что мы делали на предыдущем шаге: $5 + 6 = 11$.

На данном шаге алгоритма сумма — двузначное число. Для следующего шага алгоритма имеет значение цифра, стоящая в разряде единиц (цифра, стоящая в разряде десятков, отбрасывается). Итак, вычислили разряд тысяч — 1.

6. $6 + 1 = 7$ (сотни).

7. $1 + 7 = 8$ (десятки).

8. $7 + 8 = 15$ (последний разряд — единицы).

Выписываем последовательно результаты вычислений на каждом шаге:

41 561 785.

Полученный результат совпадает со значением в таблице.

Так же рассчитаны все остальные значения.

Запись алгоритма нахождения числа с номером N в общем виде:

1) $N + 9 = ba$;

2) $ba \rightarrow ab$;

3) $a + b = c$;

4) $b + c = d$;

5) $c + d = e$;

6) $d + e = f$;

7) $e + f = j$;

8) $f + j = h$.

Результат: *ab cde fjh*.

Ну вот — мы (вы) "запомнили" одно число. И теперь можете "запомнить" сколько угодно много. Время понадобится не для запоминания, а для подготовки демонстрации фокуса…

Все рассмотренные примеры убеждают нас в том, что математические фокусы не включают в себя элементы мистики, а являются результатом выполнения определенных алгоритмов.

Литература

1. *Депман И.* Рассказы о математике. Л.: Детгиз, 1954.

От редакции

1. Доказательство “секрета” второй забавы в общем виде, пожалуйста, пришлите в редакцию.

2. Статья о человеке, действительно обладавшем необычными “вычислительными” способностями, опубликована в статье А.И. Азевича “Человек-арифмометр” (“Информатика” № 22/2008).

3. Статьи в “Информатике”, посвященные несложным приемам быстрого счета:

— “Возведение двузначных чисел в квадрат” (№ 3/2005);

— *Исакова И.С.* “Четыре приема быстрого счета” (№ 10/2006);

— “Извлечение кубического корня в уме” (№ 22/2008);

— “Возведение в квадрат чисел пятого десятка” (№ 22/2009);

— “Умножение многозначных чисел в уме” (№ 16/2010).

ЗАДАЧНИК

Ответы, решения, разъяснения к заданиям, опубликованным в газете “В мир информатики” № 168 (“Информатика” № 14/2011)

Задача “Штирлиц и IP-адрес”

Напомним, что необходимо было по четырем разрозненным частям IP-адреса:

1) 137.72;
2) 1.2;
3) 35.;
4) 20

— восстановить исходный полный адрес.

Решение

Как известно, IP-адреса записываются в виде четырех неотрицательных целых чисел, меньших 256, разделенных точками.

Первую из имеющихся частей нельзя продолжить ни одним из остальных фрагментов (убедитесь в этом самостоятельно). Значит, искомый адрес заканчивается на 137.72. (Без последней точки.)

Перед числом 137 не могут быть записаны никакие цифры, т.е. перед ним должна стоять точка. Следовательно, предпоследний фрагмент — третий (35.), а вся найденная часть адреса выглядит так:

35.137.72

Далее, перед числом 35 не может быть записано (после точки) двузначное число. Значит, вторая часть адреса — 1.2, а весь искомый адрес такой:

201.235.137.72

Правильные ответы прислали:

— Антипин Никита, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

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

— Батракова Полина, Дукач Светлана, Максимова Ксения, Прокаев Слава и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дацук Карина и Зиновьева Анастасия, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

— Иванова Татьяна, Махмутов Роберт, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

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

— Матвеева Надежда, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

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

— Тайбарей Нина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**

Новая задача

Решите, пожалуйста, задачу о восстановлении полного IP-адреса по следующим четырем разрозненным частям:

1) 7.2;
2) 102.;
3) 47.;
4) 84.1.

Задание “Четыре столицы”

Напомним, что необходимо было, используя Интернет или другие источники информации, по нескольким известным фактам о четырех городах — столицах государств определить их название.

Ответы

1. Рига (столица Латвии).
2. Джакарта (столица Индонезии).
3. Лиссабон (столица Португалии).
4. Антананариву (столица Мадагаскара).

Правильные ответы представили:

— Архипова Александра, Бережная Ляна, Гребеньков Егор, Жданов Никита, Калиничева Анастасия, Надворный Александр, Нефедов Алексей, Никифоров Алексей, Праслова Анжела, Сизикова Полина и Шибков Максим, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

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

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

— Галабурда Анна, Захарова Юлия, Иванова Ирина, Кашпырев Александр, Козлова Наталья, Луганцов Андрей, Прохоров Юрий, Топорова Анастасия и Смирнов Максим, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Дацук Карина, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

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

— Матвеева Надежда, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

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

— Хенерина Валентина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**;

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

Большинство приславших ответы привели новые факты об указанных городах. Отметим ответы читателей из школы № 5 г. Сегежа, каждый из которых подготовил презентацию Microsoft PowerPoint с иллюстрациями об указанных четырех городах. Александра Архипова из этой же школы представила изображения с видами городов.

Задание “Азбука информатики в определениях”

Напомним, что необходимо было по приведенным определениям определить соответствующие термины, связанные с информатикой и компьютерами.

Ответы

А. Адрес, абак, антивирус.
Б. Бит, браузер, буфер.
В. Вирус, видеокарта, веб-камера.
Г. Графика, гипертекст, гиперссылка.
Д. Дисковод, дерево, диез.
Е. Ершов, Евклид, если.
Ж. Ждущий, жесткость, жесткий диск.
З. “Звезда”, значок, заставка.
И. Исполнитель, Интернет, информатика.
К. Курсор, клавиатура, ксерокс.
Л. Логика, логин, локальная сеть.
М. Магистраль, монитор, материнская.
Н. Носитель, ноутбук, ноль.
О. Оперативная память, окно, операнд.
П. Программа, пароль, процессор.
Р. Регистр, рейтинг, разряд.
С. Сервер, световое перо, сканер.
Т. Точка, таймер, транслятор.
У. Уплотненный, условный, умножение.
Ф. Файл, формула, фрагмент.
Х. Хаб, хакер, хостинг.
Ц. Цифра, цикл, циклический.
Ч. Чипсет, чат, чтение.
Ш. Шина, шифр, шрифт.
Щ. Щелчок, щи, щуп.
Э. Экран, эмулятор, энергозависимость.
Ю. Юзер, “юзать”, Ютьюб.
Я. Ячейки, ярлык, Яндекс.

Правильные ответы прислали:

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

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

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

— Лошак Антон и Турков Андрей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

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

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

Числовой ребус “ГРОМ ГРЕМИ”

Напомним, что необходимо было решить числовой ребус:

$$\begin{array}{rr} & \text{СМЕХ} \\ + & \text{ГРОМ} \\ \hline & \text{ГРЕМИ} \end{array}$$

Решение

1. Сразу видно, что **Г** = 1.

2. Рассмотрим разряд тысяч. С учетом того, что **Г** = 1, в нем может быть **С** + 1 = 10 + **Р** или **С** + 1 + + 1 = 10 + **Р** (во втором варианте еще одна едини-

ца переходит “в уме” из разряда сотен). Эти варианты могут быть в трех случаях:

1) **С** = 9, **Р** = 0;

2) **С** = 9, **Р** = 1;

3) **С** = 8, **Р** = 0.

Так как **Р** не может быть равно 1, значит, **Р** = 0, а **С** = 9 или **С** = 8.

3. В разряде сотен можно увидеть, что **М** < 9, так как при **М** = 9 в этом разряде будет **Е** = 0, а это невозможно, поскольку **Р** = 0. Следовательно, **М** < 9 и **Е** = **М** + 1.

4. Из двух последних выражений следует, что из разряда сотен в разряд тысяч ничего не переходит, значит, **С** = 9 (см. пункт 2).

5. Рассматривая сложение в разряде десятков, получим: **О** = 8, так как **Е** = **М** + 1 и **О** ≠ **С** (**С** = 9).

Итак, имеем:

```
   9МЕХ
+ 1 0 8М
-------
  10ЕМИ
```

6. Так как **Е** = **М** + 1, то из оставшихся нерасшифрованными цифр возможны пять вариантов значения **М**: 2, 3, 4, 5 и 6. Рассмотрев каждый из этих вариантов, можно обнаружить, что только **М** = 5 не приводит к противоречию, а дает следующее решение ребуса:

```
   9567
+  1085
-------
  10652
```

Правильные ответы представили:

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

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

— Есипова Мария, Круглякова Мария и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

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

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

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

Статья “Что за изображение?”

Ответ: изображение львенка (или котенка).

Ответы прислали:

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

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

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

— Авдеев Александр, Архипова Александра, Бережная Анна, Гребеньков Егор, Жданов Никита, Засядько Александр, Исаков Максим, Надворный Александр, Кочкин Даниил, Праслова Анжела, Сизикова Полина, Симкина Мария, Шибков Максим (Максим представил цветное изображение) и Яковенко Анна, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Хорьков Сергей, Москва, гимназия № 1530, учитель **Пастухов О.А.**;

— Цыганков Евгений, ученик 1-го (!) класса, Вадьковская средняя школа, Брянская обл., Погарский р-н, учитель **Цыганкова И.Ю.**

К сожалению, никто из приславших ответы не указал, что́ в задании не соответствует правилам построения изображения по пикселям на экране компьютера (на экране координаты пикселей не могут быть нецелыми). Конечно, в значительной степени это связано с возрастом соответствующих учащихся.

Задача “Пять карточек”

Напомним, что необходимо было из пяти карточек, на которых записана одна из букв: **А**, **Б**, **Е**, **Ж** или **И**, — составить последовательность из k карточек, соблюдая такие правила:

1) любая последовательность начинается буквой **А**;

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

3) буквы в последовательности не должны повторяться,

для трех случаев:

1) $k = 3$;

2) $k = 4$;

3) $k = 5$.

Ответ:

1) при $k = 3$: **АБЕ**, **АБИ**, **АЖЕ**, **АЖИ**;

2) при $k = 4$: **АБЕЖ**, **АБИЖ**, **АЖЕБ**, **АЖИБ**;

3) при $k = 5$: **АБЕЖИ**, **АБИЖЕ**, **АЖЕБИ**, **АЖИБЕ**.

Правильные ответы прислали:

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

— Батракова Полина, Дукач Светлана, Максимова Ксения, Прокаев Слава и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дацук Карина, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

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

— Иванова Татьяна, Махмутов Роберт, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Матвеева Надежда, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

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

— Тайбарей Нина и Хенерина Валентина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**

Задача “Из Ёлкино — в Палкино”

Решение

В Палкино прилетают три рейса: из аэропорта Булкино в 16:20, из аэропорта Веревкино в 16:30 и из аэропорта Ёлкино в 16:40. Можно сэкономить 10 минут по сравнению со временем прилета прямого рейса, если полететь через аэропорт Веревкино. Нужно вылететь из аэропорта Ёлкино в Веревкино в 7:30 и там сесть на самолет, отправляющийся в 14:20 в Палкино. В этом случае будет 3 часа 30 минут на пересадку. Через Булкино долететь из аэропорта Ёлкино в аэропорт Палкино нельзя, так как в момент посадки самолета в аэропорту Булкино (14:30) “стыковочный” самолет уже улетит.

Ответ: 16:30.

Правильные ответы представили:

— Антипин Никита, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

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

— Батракова Полина, Дукач Светлана, Максимова Ксения, Прокаев Слава и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дацук Карина и Зиновьева Анастасия, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

— Лошак Антон и Турков Андрей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

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

— Иванова Татьяна, Махмутов Роберт, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

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

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

Ребусы, посвященные Году космонавтики

Ответы: 1. Ракета. 2. Скафандр. 3. Астероид. 4. Шаттл. 5. Космонавт. 6. Космодром. 7. Спутник. 8. Старт. 9. Космос.

Правильные ответы прислали:

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

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

— Архипова Александра, Бережная Ляна, Гребеньков Егор, Засядько Александр, Исаков Максим, Калиничева Анастасия, Машкин Дмитрий, Нефедов Алексей, Никифоров Алексей, Праслова Анжела, Сизикова Полина и Шибков Максим, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

— Галабурда Анна, Закуленкова Виолетта, Захарова Юлия, Иванова Ирина, Кашпырев Александр, Козлова Наталья, Луганцов Андрей, Савченков Дмитрий, Смирнов Максим, Топорова Анастасия, Филимоненкова Анастасия и Яськина Валерия, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

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

— Докукина Снежана и Иванова Татьяна, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Есипова Мария, Круглякова Мария и Яснова Дарья, средняя школа поселка Осиновка, Алтайский край, учитель **Евдокимова А.И.**;

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

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

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

— Тайбарей Нина и Хенерина Валентина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**

Отметим ответы Алексея Нефедова, Дмитрия Машкина, Егора Гребенькова и Ляны Бережной,

подготовивших презентацию в Microsoft PowerPoint с иллюстрациями на космическую тему. Анастасия Калиничева включила иллюстрации в документ Microsoft Word. Александра Архипова, Александр Засядько, Максим Исаков, Анжела Праслова, Полина Сизикова и Максим Шибков представили изображения, соответствующие найденным терминам. Обратим также внимание на написание слова “Шаттл”.

Задача “На олимпиаде по информатике”

Напомним, что необходимо было определить, кто какое место занял на олимпиаде, если о пяти высказываниях:

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

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

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

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

5) Юра был четвертым, а Иван — вторым

— известно, что в каждом из них одно утверждение верное, а второе — нет.

Решение

Составим таблицу, записав в нее места, занятые учениками согласно высказываниям:

	Вася	**Гриша**	**Иван**	**Саша**	**Юра**
Высказывание 1	1				2
Высказывание 2	5			2	
Высказывание 3		3	2		
Высказывание 4		1			4
Высказывание 5			2		4

Анализ показывает, что ключевыми являются высказывания, характеризующие того, кто занял второе место (число 2 встречается в таблице чаще других). Иван не может занять второе место, потому что тогда все высказывания о Юре и Саше должны быть ложными, а истинными оказались бы два взаимоисключающих суждения о Васе в высказываниях 1 и 2, что ведет к противоречию. Поэтому суждения о втором месте Ивана ложны, а о четвертом месте Юры и третьем Гриши — истинны. Итак, Вася занял первое место, Саша — второе, так как в высказывании 2 суждение о пятом месте Васи ложно, значит, первое суждение истинно. Ивану досталось пятое место.

Можно также решить задачу методом допущений.

Допустим, что в высказывании 1 первое утверждение истинное (“первое место занял Вася”), а второе (“Юра занял второе место”) — ложное. Обозначив это схематически так:

Высказывание 1: Вася 1 + Юра 2 –,

можем вывести логическую цепочку:

в высказывании 1: Вася 1 + Юра 2 –,

из чего в высказывании 4: Гриша 1 –, а Юра 4 +,

отсюда в высказывании 5: Юра 4 –, то есть Иван 2 +,

из чего в высказывании 3: Иван 2 –, а Гриша 3 +,

и в высказывании 2: Иван 2 –, то есть Вася 5 +,

но последнее противоречит сделанному допущению.

Рассмотрение всех вариантов также приводит к ответу: 1-е место — Вася, 2-е место — Саша, 3-е место — Гриша, 4-е место — Юра, 5-е место — Иван.

Ответы представили:

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

— Дацук Карина и Зиновьева Анастасия, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

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

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

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

— Михайлов Иван, Гаязова Фатима и Хорькова Анна, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

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

— Яновский Виталий, Москва, гимназия № 1530, учитель **Козырева О.В.**

Задача “Сколько весит бидон?”

Напомним, что необходимо было определить, сколько весит бидон, если полный бидон с молоком весит 7 кг, а наполненный наполовину — 4 кг.

Решение

Из условия следует, что молоко, занимающее полбидона, весит $7 - 4 = 3$ кг. Значит, молоко в полностью заполненном бидоне весит $3 \times 2 = 6$ кг, из чего следует, что вес отдельно бидона составляет $7 - 6 = 1$ кг.

Ответ: 1 кг.

Правильные ответы прислали:

— Антипин Никита, Куминская средняя школа, Тюменская область, Ханты-Мансийский автономный округ — Югра, Кондинский р-н, учитель **Шишигина О.В.**;

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

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

— Иванова Ирина, Павлючкова Юлия и Топорова Анастасия, Смоленская обл., г. Демидов, школа № 1, учитель **Кордина Н.Е.**;

— Иванова Татьяна, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

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

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

Обратим внимание на то, что задача предназначалась ученикам начальной школы и учащимся 5–7-х классов.

Задача "Эликсир бессмертия"

Напомним, что необходимо было установить, как с помощью песочных часов на 3 минуты и на 8 минут за минимально возможное число операций (установок часов и их переворачиваний) отмерить *непрерывно* 7 минут для приготовления эликсира бессмертия.

Ответы представили:

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

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

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

— Матвеева Надежда, г. Ярославль, школа № 33, учитель **Цикина Е.Н.**;

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

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

— Ситникова Татьяна и Толмачева Ксения, Москва, гимназия № 1530, учитель **Козырева О.В.**;

— Хотеев Сергей, Москва, гимназия № 1530, учитель **Пастухов О.А.**

Большинство приславших ответы предложили такой вариант решения.

1–2. Одновременно запустить 8-минутные и 3-минутные часы, но пока не готовить.

3. Когда 3-минутные часы "остановятся" — запустить их еще раз (прошло 3 минуты).

4. Еще через три минуты опять запустить 3-минутные часы (с начала прошло 6 минут).

5. Когда 8-минутные часы "остановятся", начать готовить эликсир и делать это одну минуту — до истечения времени на 3-минутных часах.

6–7. Еще дважды готовить по 3 минуты — и эликсир готов!

В описанном алгоритме надо сделать 7 операций (установок часов и их переворачиваний). Есть алгоритм с меньшим числом операций (его нашли учащиеся из школы № 8 поселка Стрелка Красноярского края). Предлагаем читателям также найти более рациональный вариант и прислать его в редакцию.

Задача "Наименьшее число"

Напомним, что необходимо было, пользуясь только двумя одинаковыми цифрами и знаками действий, записать как можно меньшее число.

Ответ: $-(9)^9$.

Ответы прислали:

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

— Батракова Полина, Дукач Светлана, Максимова Ксения, Прокаев Слава и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка, школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

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

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

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

— Махмутов Роберт, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

— Хенерина Валентина, основная школа поселка Каратайка, Архангельская обл., учитель **Безумова В.А.**

Задача "Индюки и жеребята"

Напомним, что необходимо было решить задачу в стихах, но не составляя систему уравнений, а методом рассуждений.

Решение

Если всех индюков заменить жеребятами, то количество хвостов не изменится ☺. А вот ног станет больше. На сколько? Так как у жеребенка 4 ноги и жеребят получилось 11, то "новое" число ног будет $11 \times 4 = 44$, то есть их стало больше на $44 - 30 = 14$. Поскольку при "замене" одного индюка одним жеребенком становится больше на две ноги, то "новые" 14 ног появились за счет "замены" семи индюков.

Ответ: первоначально было 7 индюков и 4 жеребенка.

Можно также рассмотреть вариант "Если всех жеребят заменить индюками…".

Правильные ответы представили:

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

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

— Батракова Полина, Дукач Светлана, Максимова Ксения, Прокаев Слава и Соболев Иван, г. Лесосибирск Красноярского края, поселок Стрелка,

школа № 8 им. Константина Филиппова, учитель **Лопатин М.А.**;

— Дацук Карина и Зиновьева Анастасия, Республика Башкортостан, г. Уфа, лицей № 60, учитель **Гильзер Н.В.**;

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

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

— Парамонова Анастасия, Ставропольский край, Кочубеевский р-н, станица Барсуковская, школа № 6, учитель **Рябченко Н.Р.**;

— Махмутов Роберт, Сухоруков Антанас и Фазлыев Фанис, Республика Башкортостан, г. Стерлитамак, школа № 24, учитель **Орлова Е.В.**;

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

Отметим ответ Натальи Решетниковой, включившей в него (в документ Microsoft Word) изображения животных.

Задание “Четыре вопроса” (рубрика “Поиск информации”)

Ответы

1. Название модели автомобильной фирмы Renault, связанное с Интернетом, — Trafic. Юлия Шадрина предложила вариант — Рено Логан (название, близкое к термину *логин*).

2. Ряд фамилий Ней, Виктор, … можно продолжить следующими: Бернадот, Бертье, Виктор, Мюрат, Ней и др. (все эти люди — маршалы Наполеона I).

3. В результате объединения разных государств образовались следующие государства — Танзания (из отдельных государств Танганьика и Занзибар), Германия (ФРГ и ГДР) и др. В результате разъединения СССР образовались все страны СНГ (Россия, Казахстан, Беларусь и др.), Балтии (Латвия, Литва, Эстония), Грузия. Чехия и Словакия были образованы из единой страны Чехословакии. Можно привести и другие примеры.

4. За время наблюдений 26 октября в Москве самая низкая температура воздуха составляла –15,2°С (1880 г.), самая высокая — +15,6°С (1954 г.).

Ответы прислали:

— Ан Роман, Москва, гимназия № 1530, учитель **Козырева О.В.**;

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

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

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

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

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

— Яновский Виталий, Москва, гимназия № 1530, учитель **Козырева О.В.**

Задача “101 монета, одна фальшивая”

Напомним, что необходимо было среди 101 одинаковой по виду монеты с помощью чашечных весов без гирь за два взвешивания определить, легче или тяжелее настоящих единственная имеющаяся фальшивая монета (саму фальшивую монету находить не требуется).

Решение

Положим на каждую чашку весов по пятьдесят монет. Если чашки уравновесились, то оставшаяся, 101-я, монета — фальшивая. Сравнивая ее на весах с любой из настоящих монет, можно узнать, легче она или тяжелее их.

Если при первом взвешивании весы не находятся в равновесии, то возьмем 50 монет с более тяжелой чашки, разделим на две кучки по 25 монет и положим на чашки весов. Если весы уравновесятся, то фальшивая монета находилась на чашке, которая при первом взвешивании оказалась легче, т.е. она (монета) — легче настоящих. Если при втором взвешивании весы не уравновесятся, то среди сравниваемых 50 монет есть фальшивая, и она — тяжелее настоящих.

Возможны и другие варианты решения.

Правильные ответы представили:

— Ан Роман, Москва, гимназия № 1530, учитель **Козырева О.В.**;

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

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

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

— Михайлов Иван, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Шагинуров Эмиль, средняя школа села Урман, Республика Башкортостан, Иглинский р-н, учитель **Товмасян М.Г.**

Игра “Три кучки камней”

Напомним, что необходимо было определить, кто выиграет (начинающий игру или делающий ход вторым) в такую игру: “Имеются три кучки камней: в первой — 10, во второй — 15, в третьей — 20. Играют двое. За ход разрешается разбить любую кучку на две меньшие. Проигрывает тот, кто не сможет сделать ход”.

Решение

После каждого хода количество кучек увеличивается на 1. Сначала их было 3, в конце — 45. Таким образом, всего будет сделано 42 хода. Это значит, что последний, 42-й, сделает второй игрок, и именно он обязательно выиграет.

Ответы прислали:

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

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

— Михайлов Иван, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Федоренко Степан, средняя школа поселка Озеры Красноярского края, учитель **Филипченко И.С.**;

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

— Яновский Виталий, Москва, гимназия № 1530, учитель **Козырева О.В.**

Статья “Игры на клетчатой доске”

Ответы

1. Результат игр “Короли на клетчатой доске” и “Доминошки на клетчатой доске” зависит от размеров доски. Если общее число клеток — четное, то выигрывает делающий ход вторым (он всегда может сделать ход, симметричный ходу начинающего игру).

2. В игре, в которой двое по очереди ставят коней в клетки шахматной доски так, чтобы они не били друг друга (цвет коней значения не имеет), а проигрывает тот, кто не сможет сделать ход, — выиграет делающий ход вторым (можно использовать и центральную, и осевую симметрию).

Ответы представили:

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

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

— Лошак Антон и Турков Андрей, средняя школа села Сердар, Республика Марий Эл, учитель **Чернова Л.И.**;

— Михайлов Иван, средняя школа села Восточное Нижегородской обл., учитель **Долгова Г.А.**;

— Тарасюк Степан, средняя школа поселка Озеры Красноярского края, учитель **Филипченко И.С.**;

— Яценюк Станислав, Москва, гимназия № 1530, учитель **Козырева О.В.**

Программы, предложенные для самостоятельной работы в статье “Сколько квадратов в круге?”, представили:

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

— Цховребов Зураб, г. Пенза, школа № 512, учитель **Гаврилова М.И.**;

— Яценюк Станислав, Москва, гимназия № 1530, учитель **Козырева О.В.**

Правильные ответы и решения прислали также:

1) задача “Три брата”:

— Абросимов Денис, Алексеев Николай, Ашурко Анна, Байбабаева Анастасия, Богомолова Елизавета, Васильев Глеб, Гловацкая Виктория, Дуплий Дмитрий, Евченко Ульяна, Зунда Ксения, Иванов Глеб, Евдокимова Анна, Иванова Александра, Корхалёва Любовь, Костева Дарья, Лунина Валерия, Малышев Никита, Москвин Даниил, Пахоменко Елизавета, Пименова Светлана, Попова Мадина, Туху Виталий, Черткова Елизавета, Шаблыко Даниил и Юпилайнен Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

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

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

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

2) задача “Четыре сообщения”:

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

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

3) задача “Любительница мороженого”:

— Абросимов Денис, Алексеев Николай, Ашурко Анна, Байбабаева Анастасия, Богомолова Елизавета, Васильев Глеб, Гловацкая Виктория, Дуплий Дмитрий, Евченко Ульяна, Зунда Ксения, Иванов Глеб, Евдокимова Анна, Иванова Александра, Корхалёва Любовь, Костева Дарья, Лунина Валерия, Малышев Никита, Москвин Даниил, Пахоменко Елизавета, Пименова Светлана, Попова Мадина, Туху Виталий, Черткова Елизавета, Шаблыко Даниил и Юпилайнен Дарья, Республика Карелия, г. Сегежа, школа № 5, учитель **Меньшиков В.В.**;

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

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

Ответы на задания, опубликованные в сентябрьском номере, прислали также ученицы основной школы поселка Михинский, Воронежская обл., Таловский р-н (учитель **Удалова А.А.**):

— Петренко Анастасия (чайнворд “Соточка”);

— Манохина Татьяна (задание “Азбука информатики в определениях”).

Спасибо всем!

Списки читателей, приславших ответы “обычной” почтой, будут опубликованы в следующем выпуске.

Четыре задачи на системы счисления

Решите, пожалуйста, четыре задачи.

1. В какой системе счисления справедливо равенство $3 \times 4 = 10$?

2. Существует ли система счисления, в которой одновременно:

а) $3 + 4 = 10$ и $3 \times 4 = 15$;

б) $3 + 2 = 5$ и $3 \times 2 = 11$?

3. На доске сохранилась полустертая запись:

$$\begin{array}{cccccc} & 2 & 3 & & 5 & \\ + & 1 & & 6 & 4 & 2 \\ \hline & 4 & 2 & 4 & 2 & 3 \end{array}$$

Восстановите слагаемые.

4. Один школьный учитель информатики написал: “У меня в классе 100 детей, из них 24 мальчика и 32 девочки”. Возможно ли это?

Ответы присылайте в редакцию (можно решать не все задачи).

Бой часов

Имеются двое часов. Первые бьют через каждые 2 сек., вторые — через каждые 3 сек. Совпавшие по времени удары воспринимаются как один. Сколько времени часы били, если они начали бить одновременно и всего было насчитано 13 ударов? Ответ получите (желательно), используя электронную таблицу Microsoft Excel или другую либо разработав компьютерную программу.

Шесть деревьев[2]

Возле школы растут 6 деревьев: сосна, береза, липа, тополь, ель и клен. Какое из деревьев самое высокое и какое самое низкое, если известно, что береза ниже тополя, а липа выше клена, сосна ниже ели, липа ниже березы, сосна выше тополя?

Ученики и библиотеки

В классе 35 учеников. Все они являются читателями школьной и районной библиотек. Из них 25 берут книги в школьной библиотеке, 20 — в районной. Сколько из них:

а) не являются читателями школьной библиотеки;

б) не являются читателями районной библиотеки;

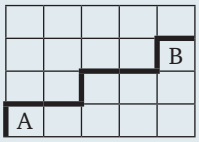
в) являются читателями только школьной библиотеки;

г) являются читателями только районной библиотеки;

д) являются читателями обеих библиотек?

Лестница

На координатной плоскости нужно нарисовать лестницу из трех ступенек из точки A в точку B. Точка A имеет координаты (0,0), а точка B — (5,3). Каждая ступенька должна быть высотой в одну единицу и иметь целое количество единиц в длину. Один из возможных вариантов показан ниже:



Каждая лестница может быть закодирована тройкой чисел, задающих длины первой, второй и третьей ступенек соответственно. Так, изображенная на рисунке кодируется тройкой 2, 2, 1.

Определите, сколько всего может быть таких лестниц, и перечислите все тройки чисел, соответствующие этим вариантам.

Три музыканта

В симфонический оркестр приняли на работу трех музыкантов: Брауна, Смита и Вессона, умеющих играть на скрипке, флейте, альте, кларнете, гобое и трубе. Известно, что:

1) Смит самый высокий;

2) играющий на скрипке меньше ростом играющего на флейте;

3) играющие на скрипке и флейте и Браун любят пиццу;

4) когда между альтистом и трубачом возникает ссора, Смит мирит их;

5) Браун не умеет играть ни на трубе, ни на гобое.

На каких инструментах играет каждый из музыкантов, если каждый владеет двумя инструментами?

ПОИСК ИНФОРМАЦИИ

Четыре вопроса

1. В каком городе родился чемпион мира по шахматам по имени Хосе?

2. Какой знак препинания ввел в российскую письменность писатель, имя которого — Николай?

3. В каком городе Франции имеется крупнейшая в стране трамвайная сеть?

4. Какой фрукт тайцы перед съедением посыпают перцем и солью?

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

[2] Задача предназначена для учащихся начальной школы и учеников 5–7-х классов.

Двоичная система у… древних египтян

Вы, конечно, знаете, что двоичная система счисления — основа работы компьютеров. А известны ли вам другие примеры использования этой системы? В нашем журнале приводилось несколько таких примеров — штриховое кодирование информации о товаре [1], кодирование марок фотопленок [2] и даже — приготовление пиццы ☺ [3]. Оказывается, двоичную систему применяли (конечно, сами того не понимая) древние египтяне.

Для умножения двух натуральных чисел они использовали методику, в которой для вычислений достаточно было уметь складывать, вычитать и умножать на 2. Делалось это так. В двух колонках записывались два числа: 1 и один из сомножителей. Затем в каждой следующей строке под ними записывались числа, в 2 раза большие. Так продолжалось до тех пор, когда в первой колонке получалось число, большее второго множителя. Пример для нахождения произведения 17 на 21:

1	21
2	42
4	84
8	168
16	336
32	

Результат в данном случае получался сложением чисел в правой колонке, которым в левой соответствуют числа 16 и 1: 336 + 21 = 357 (проверьте на калькуляторе, что 17 × 21 = 357). Почему?

Ответ такой.

1. Значения в правой колонке можем записать так:

21		21×1		21×2^0
42		21×2		21×2^1
84	или	21×4	или	21×2^2
168		21×8		21×2^3
336		21×16		21×2^4

2. Переведем число 17 в двоичную систему методом выделения максимальной степени двойки. 21 = 16 + 1, то есть $17_{10} = 10001_2$. Теперь умножим 17 на 21, но записав число 17 в так называемой “развернутой форме”:

$17 \times 21 = (1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) \times 21$

или

$17 \times 21 = (2^4 + 2^0) \times 21 = 21 \times 2^4 + 21 \times 2^0$.

3. Учитывая последнее равенство, для нахождения произведения 17 на 21 во второй колонке “основной ” таблицы необходимо сложить числа в строках, соответствующих числам 16 и 1, что мы и сделали при вычислениях ранее.

Похожий метод умножения использовали русские крестьяне (поэтому его называют “крестьянским”). Проиллюстрируем этот метод на примере нахождения произведения 13 на 18. Нарисуем таблицу, в первой колонке которой запишем число 13, а во второй — 18 (см. ниже). Далее каждое следующее число в первой колонке получаем из предыдущего делением нацело на 2, а во второй колонке — умножением предыдущего на 2:

13	18
6	36
3	72
1	144

После того как в первой колонке получим 1, деление и умножение прекратим.

Далее найдем в первой колонке четные числа и вычеркнем строки с ними:

13	18
~~6~~	~~36~~
3	72
1	144

Искомое произведение равно сумме оставшихся во втором столбце чисел: 18 + 72 + 144 = 234 (проверьте!).

Задания для самостоятельной работы

1. Как можно объяснить результат, получаемый при “крестьянском” методе умножения?

2. Покажите, что результат, получаемый при “древнеегипетском” и при “крестьянском” методах умножения, не зависит от порядка множителей.

3. Разработайте алгоритм умножения натуральных чисел, который основывался бы на троичной системе счисления (при этом использовались бы следующие операции: деление на 3 с остатком, умножение на 3 и сложение).

Литература

1. Кодирование информации о товаре. / “В мир информатики” № 55 (“Информатика” № 8/2005).

2. Двоичное кодирование информации в пленочных фотоаппаратах. / “В мир информатики” № 36 (“Информатика” № 37/2004).

3. Повар и пицца. / “В мир информатики” № 87 (“Информатика” № 5/2007).

“ЛОМАЕМ” ГОЛОВУ

Шесть “РЕГИСТРОВ”

Лейб Штейнгарц, Израиль

Читателям раздела “В мир информатики” конечно же известно, что такое “регистр”. Предлагаю решить несколько задач и числовых ребусов, связанных с этим словом. В них одинаковые цифры заменены одинаковыми буквами, а разные цифры — разными буквами.

1. Если выполняется следующее равенство:

РЕГ + **И** = **СТР**,

то чему равно произведение всех цифр числа **РЕГИСТР**?

2. Если выполняется следующее равенство:

$$\text{РЕ}^{\text{Г}} = \text{ИСТР},$$

то чему равно число **РЕГИСТР**?

3. Решите, пожалуйста, ребус:

РЕГИ : **СТР** = 4 : 3

4. Решите, пожалуйста, числовой ребус:

РЕГИ – **СТР** = **РЕГ** – **ИС**

5. Выполняется следующее равенство: в правой части которого расположено некоторое пятизначное число, цифры которого являются последовательными цифрами, расположенными в порядке убывания.

РЕГИС – **ТР** = □□□□□

Найдите значения букв-цифр числа **РЕГИСТР**.

6. При каких значениях букв справедливо равенство:

Р × **ЕГ** × **ИСТР** = 1 000 000?

Загадочное деление

В следующем примере на деление все цифры заменены звездочками, кроме семи семерок:

```
* * 7 * * * * * * * | * * * * 7 *
* * * * * *         | * * 7 * *
-----------
* * * * * 7 *
* * * * * * *
---------------
    * 7 * * * *
    * 7 * * * *
  -------------
  * * * * * * *
  * * * * 7 * *
  -------------
      * * * * * *
      * * * * * *
    -------------
                0
```

Определите делимое, делитель и частное.

Азбука информатики

Н.А. Владимирова,
учитель информатики гимназии № 2,
г. Заозерный Красноярского края

1. Азбука информатики в пословицах, поговорках и афоризмах

В приведенных ниже пословицах и поговорках отдельные слова или несколько последовательно идущих букв образуют термин, связанный с информатикой и ИКТ. Искомые термины начинаются на букву, оформленную жирным начертанием. Найдите их. Прокомментируйте найденные термины.

А Где хотите, там и бранитесь, а на табаке помиритесь.

Б Умел ошибиться, умей и поправиться.

В Рожь кормит сплошь, а пшеничка — по выбору.

Г Всякая птица свое гнездо любит.

Д Широка дорожка Ерошке.

Е И мед горчит, если его много.

Ж Ржавое железо не блестит.

З Если мало в привозе, то много в запросе.

И Истина выдерживает проверку опытом.

К Вода размывает канал в слабом месте.

Л Глупа та птица, которой гнездо свое не мило.

М На ходу подметки режет.

Н Где хвост — начало, там голова — мочало.

О Всякой молодец на свой образец.

П Закон — что паутина: шмель проскочит, а муха увязнет.

Р Дело сделал на копейку, а раструбил на всю деревню.

С Долгие сборы на короткий век.

Т Юность — время золотое: ест и пьет и спит в покое.

У Шито-крыто, а узелок-то тут.

Ф Два фонаря на пустой каланче.

Х Как не хитри, а правды не перехитришь.

Ц Цветет старость сединою.

Ч К мягкому воску печать, к молодому — ученье.

Ш Шумит дубровушка к погодушке.

Э Каково эхо, такова и конюшня.

Ю Верный вассал двум сюзеренам не служит.

Я Откладывать в долгий ящик.

Азбука информатики в анаграммах

Из букв приведенных ниже данных слов, переставив их, получите термин, связанный с информатикой и ИКТ. Обратим внимание на то, что, хотя в названии задания фигурирует слово “азбука”, искомые слова расположены не в алфавитном порядке.

1. ИГОЛКА. 2. МОЗЕР. 3. КОРВЕТ. 4. НИША. 5. КЕД. 6. ТУРНЕ. 7. ЗЕЛОЖЕ. 8. ТЕНОР. 9. ЗРЮЕ. 10. РАКЕТКА. 11. ПЕТАРДА. 12. АРБУЗЕР. 13. ФРИЦА. 14. ЗАБОР. 15. ЛАПТА. 16. ЭТАНОЛ. 17. КОРСЕТ. 18. РЕЗОН. 19. ИЛИТАТУ. 20 ЛАЙФ. 21. МАГМА. 22. КЛЕЩЧО. 23. ЧАСТИК. 24. КОКАИН. 25. ДОЯР. 26. ШЕВРО. 27. БАХ. 28. БАРОН.

Ответы присылайте в редакцию (можно находить не все термины).

Цифры вместо “?”

Какие цифры надо поставить вместо символов “?”, чтобы получить равенство?

1. (123 456 789 × ?) + ? = 987 654 321.
2. (12 345 678 × ?) + ? = 111 111 111.
3. 12 345 679 × ?? = 999 999 999.

Бочка с квасом

В бочке 16 ведер кваса. Надо поделить его пополам, имея две пустых емкости на 6 и 11 ведер. Как это сделать? Алгоритм решения задачи, пожалуйста, оформите в виде:

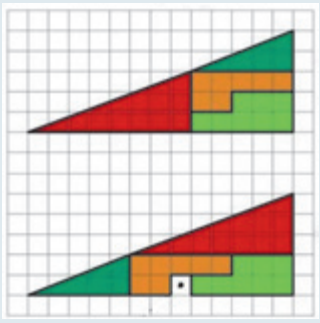
	Действие	Бочка	Емкость 6	Емкость 11
0	Исходное состояние	16	0	0
1	Перелить из бочки в 6	10	6	0
2	…			

Набор слов

Дан набор слов: AA, AAA, ABA, ABBA, BAC, BBA, BBAC. Расположите эти слова последовательно так, что каждое следующее слово получалось из предыдущего либо добавлением одной буквы, либо удалением одной буквы. Добавлять и удалять буквы можно в любом месте слова (в начале, в середине, в конце).

Куда пропал квадратик?

Посмотрите, пожалуйста, на верхний рисунок. На нем треугольник разрезан на несколько фигур, из которых собран треугольник на нижнем рисунке.



Размеры треугольников на обоих рисунках одни и те же, но на нижнем рисунке один квадратик не используется. Как это может быть, ведь на этом рисунке имеются все части верхнего рисунка?

ШКОЛА ПРОГРАММИРОВАНИЯ

Решаем задачу “Бой часов”

Условие задачи приведено в рубрике “Задачник”.

Идея решения достаточно ясная — “считать” секунды и в те моменты времени, когда хотя бы одни часы бьют, увеличивать счетчик общего числа ударов. В программе на школьном алгоритмическом языке последнюю величину назовем *всего_ударов*, а считаемые секунды — *сек*.

Так как мы не знаем, когда общее число ударов станет равно 13, то в программе следует применить оператор цикла с условием, например, с предусловием. Поскольку мы будем рассматривать каждую секунду, а значение *всего_ударов* будет увеличиваться на 1, то условие выполнения оператора цикла с предусловием такое:

всего_ударов < 13

Если вести отсчет времени от первого одновременного удара (*сек* = 0), то можно утверждать, что первые часы будут бить, когда значение *сек* кратно двум, а вторые — когда это значение кратно трем. В моменты времени, кратные шести (часы бьют одновременно), в значении *всего_ударов* должен учитываться один общий удар.

Вся программа имеет вид:

```
алг Бой часов
нач цел сек, всего_ударов
. сек := 0                |Начало замера
. всего_ударов  :=  1 |Первый удар
. нц пока всего_ударов < 13
. . сек := сек + 1 |Следующая секунда
. . |Если бьют хотя бы одни часы
. . если mod(сек, 2) = 0 или
          mod(сек, 3) = 0
. . . то |Увеличиваем счетчик числа
          | ударов
. . . . всего_ударов := всего_ударов + 1
. . все
. кц
. |Выводим ответ
. вывод нс, "Часы били ", сек,
            " секунд"
кон
```

Обратим внимание на то, что при приведенном оформлении оператора цикла искомое значение равно значению величины *сек* после окончания работы оператора.

Задания для самостоятельной работы

1. Разработайте вариант программы (на языке программирования, который вы изучаете) с оператором цикла с постусловием.

2. Использовав любой вариант программы, определите, через какое время общее число ударов часов станет равно 100.

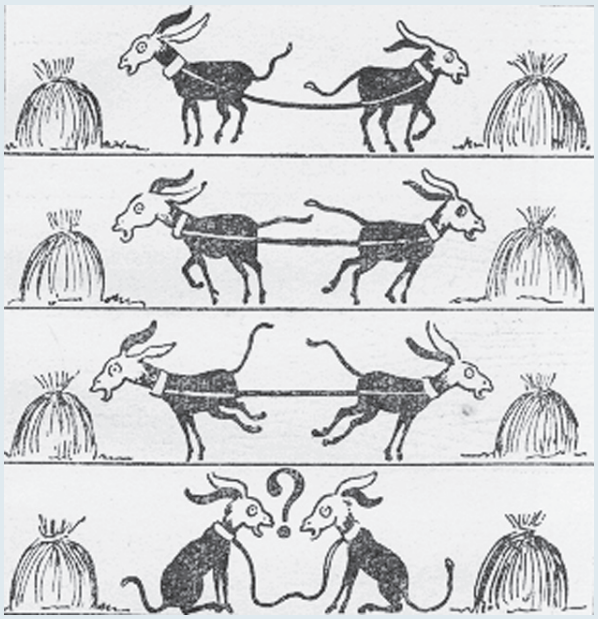
Результаты присылайте в редакцию.

ПРИ1Ы

Ослы и информатика[3] ☺

Подумайте, могут ли ослы выйти из “трудного” положения, показанного на рисунке. Если могут, то опишите соответствующий алгоритм.

[3] Задание предназначено для учащихся начальной школы.



Ответы, пожалуйста, присылайте в редакцию.

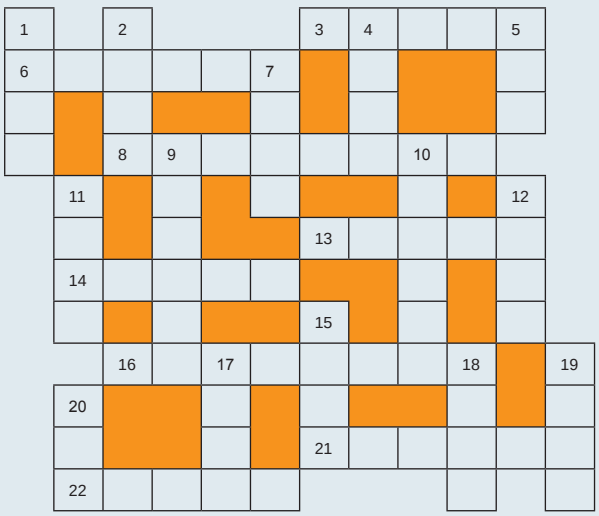
ВНИМАНИЕ! КОНКУРС

Конкурс № 92

В качестве задания этого конкурса предлагаем решить задачи в статье «Шесть “РЕГИСТРОВ”» в этом выпуске.

Ответы отправьте в редакцию до 10 марта по адресу: 121165, Москва, ул. Киевская, д. 24, “Первое сентября”, “Информатика” или по электронной почте: *vmi@1september.ru*. Пожалуйста, четко укажите в ответе свои фамилию и имя, населенный пункт, номер и адрес школы, фамилию, имя и отчество учителя информатики. Можно решать не все задачи.

Кроссворд



По горизонтали

3. Один из контактов транзистора типа **МОП** (**М**еталл — **О**ксид — **П**олупроводник).

6. Рассмотрение и при необходимости обработка всех элементов массива.

8. Изображение чего-нибудь, рассказ о ком-нибудь (о чем-нибудь) в письменной или устной форме.

13. Поименованная группа файлов, объединенных по какому-то признаку.

14. Процесс написания текста на компьютере.

16. Совокупность четко определенных правил для решения задачи за определенное число шагов.

21. Указатель места на экране монитора.

22. Так иногда называют столбец в таблице.

По вертикали

1. Непрошеное рекламное сообщение, сетевой “мусор”.

2. Язык программирования (как правило, для начинающих изучать последнее).

4. Жаргонное название результата обработки изображения на устройстве ввода информации в компьютер.

5. Число в системе условных обозначений символов.

7. Разновидность носителя информации.

9. Название клавиши.

10. Ввоз товаров из-за рубежа, а также вставка в документ приложений Windows объектов из других приложений.

11. Часть экрана, занимаемая приложением или документом Windows.

12. … данных.

15. Синоним слова “дорожка” (участка магнитного диска).

17. Конечное число точек на плоскости, соединенных отрезками кривых линий.

18. Устройство для соединения двух участков локальной сети.

19. Знак препинания.

20. Элемент языка разметки гипертекста.

Общероссийский проект «Школа цифрового века» по комплексному обеспечению образовательных учреждений методической интернет-поддержкой разработан в соответствии с программой модернизации системы общего образования России и направлен на повышение профессионального уровня педагогических работников

Общероссийский проект

Школа цифрового века

Интернет-сопровождение проекта – Издательский дом «ПЕРВОЕ СЕНТЯБРЯ»

2012/13
учебный год

Комплексное обеспечение образовательных учреждений предметно-методическими материалами по всем учебным дисциплинам и направлениям школьной жизни с адресной доставкой на современных носителях

С начала следующего учебного года в рамках проекта реализуется программа развития профессионально-личностных компетенций педагога

Прием заявок от школ – с 15 февраля 2012 года

Подробности на сайте digital.1september.ru

СПРАВКА О ХОДЕ РЕАЛИЗАЦИИ ОБЩЕРОССИЙСКОГО ПРОЕКТА «ШКОЛА ЦИФРОВОГО ВЕКА» В 2011/12 УЧЕБНОМ ГОДУ. По состоянию на 1 февраля 2012 года в проекте принимают участие 5892 образовательных учреждения. В том числе 411 лицеев и гимназий, 309 школ с углубленным изучением отдельных предметов, 119 школ-интернатов, 54 колледжа, включая 13 педагогических, 168 школ для детей с ограниченными возможностями здоровья, а также детей, попавших в трудную жизненную ситуацию, 48 вечерних (сменных) школ, 99 дошкольных образовательных учреждений, 48 учреждений дополнительного образования, 8 высших учебных заведений. Всего в проекте участвуют 181 505 педагогических работников.